École de technologie supérieure
Service des enseignements généraux
Local B-2500 514-396-8938
Site internet: http://www.etsmtl.ca/

MAT215

LOGIQUE ET MATHÉMATIQUES DISCRÈTES POUR L'OPTIMISATION

NOTES DE COURS

RÉDIGÉES PAR
GENEVIÈVE SAVARD,
ANOUK BERGERON-BRLEK
ET XAVIER PROVENÇAL

VERSION RÉVISÉE EN DÉCEMBRE 2023

Table des matières

Avant-propos **vii**

1 Logique et ensembles **1**
1.1 Logique du premier ordre . . . 1
1.1.1 Logique propositionnelle . . . 1
1.1.2 Équivalences propositionnelles . . . 14
1.1.3 Prédicats et quantificateurs . . . 19
1.1.4 Quantificateurs imbriqués . . . 28
1.2 Raisonnements . . . 34
1.2.1 Règles d'inférence . . . 34
1.2.2 Cohérence d'un ensemble de spécifications . . . 38
1.2.3 Types de preuve . . . 46
1.3 Théorie des ensembles . . . 48
1.3.1 Notions de base sur les ensembles . . . 48
1.3.2 Ensembles de nombres $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ . . . 49
1.3.3 Produit cartésien . . . 53
1.3.4 Opérations sur les ensembles $\cap, \cup \oplus, -$ . . . 55
1.3.5 Représentation de sous-ensembles par trains de bits . . . 57
1.4 Fonctions . . . 62
1.4.1 Fonctions plancher et plafond . . . 64

2 Modélisation **65**
2.1 Exemple 1 : plan de production de l'entreprise 3P . . . 65
2.1.1 Un premier modèle pour l'entreprise 3P . . . 66
2.1.2 Utilisation d'un tableur muni d'un solveur . . . 67
2.1.3 Présentation d'une solution optimale . . . 69
2.1.4 Bonne pratique : nommer des plages de cellules d'une feuille de calcul . . . 69
2.1.5 Un modèle algébrique : compact et adaptable . . . 72
2.2 Exemple 2 : répartition des points de service . . . 73
2.2.1 Un premier modèle, simple mais peu adaptable . . . 74
2.2.2 Un modèle algébrique : compact et adaptable . . . 75
2.2.3 Utilisation d'un tableur muni d'un solveur . . . 77
2.2.4 Présentation d'une solution optimale . . . 79
2.3 Exemple 3 : planification d'un horaire pour minimiser le coût des salaires. . . . 79
2.3.1 Une première tentative de répartition. . . . 80
2.3.2 Une seconde tentative de répartition. . . . 80
2.3.3 Un modèle mathématique avec coefficients numériques . . . 82
2.3.4 Un modèle algébrique : adaptable et compact, mais grosse soupe à l'alphabet ! . . . 82

2.3.5 Utilisation du solveur pour obtenir une solution optimale . . . 83
2.4 Exemple 4 : encore une planification d'un horaire pour minimiser le coût des salaires. . . . 84
2.4.1 Avec le solveur . . . 84
2.4.2 Sans le solveur . . . 85

3 Théorie des graphes 91
3.1 Terminologie et types de graphes . . . 91
3.2 Représentation des graphes . . . 96
3.2.1 Représentation par listes d'adjacence . . . 96
3.2.2 Représentation par matrice d'adjacence . . . 97
3.3 Chemins dans un graphe . . . 99
3.3.1 Chemins, circuits, cycles . . . 99
3.3.2 Dénombrement de chemins . . . 100
3.3.3 Chemins et circuits eulériens . . . 102
3.3.4 Chemins et circuits hamiltoniens . . . 109
3.4 Problème du plus court chemin (algorithme de Dijkstra) . . . 110
3.4.1 Exemple : plan d'approvisionnement . . . 116
3.5 Tri topologique (algorithme de Kahn) . . . 122
3.6 Chemin de poids maximal . . . 128
3.6.1 Chemin de poids maximal dans un graphe orienté acyclique . . . 128

4 Arbres 135
4.1 Définitions de base . . . 135
4.1.1 Exemple : arbres syntaxiques . . . 140
4.2 Quelques théorèmes sur les arbres . . . 141
4.3 Arbre couvrant d'un graphe . . . 146
4.3.1 Arbre couvrant de poids minimal (algorithme de Prim) . . . 151

5 Introduction à la complexité des algorithmes 157
5.1 Mesurer un temps de calcul à l'aide d'une fonction . . . 158
5.2 Notation grand-O et grand-Θ . . . 162
5.2.1 Grand-O d'une fonction composée . . . 170
5.3 Sommations . . . 175
5.4 Établir la fonction de complexité d'un algorithme . . . 177
5.5 Calculabilité et complexité . . . 183

6 Algorithmes récursifs 185
6.1 Définition et exemples d'algorithmes récursifs . . . 186
6.2 Fonctions récursives et relations de récurrence . . . 189
6.2.1 Résolution de relation de récurrence par la méthode itérative . . . 190
6.3 Algorithmes de type diviser pour régner . . . 192
6.3.1 Algorithmes et relations de récurrence de type diviser pour régner . . . 193
6.3.2 Résolution de relation de type diviser pour régner par la méthode itérative . . . 194

7 Preuve par récurrence 199
7.1 Preuve par récurrence simple . . . 199
7.2 Preuve par récurrence forte . . . 206
7.3 Preuve de validité d'un algorithme récursif . . . 210

8 Dénombrement 215

8.1 Notions de base 215
8.1.1 Principe du produit 215
8.1.2 Principe de la somme 216
8.1.3 Principe d'inclusion-exclusion 216
8.1.4 Principe des tiroirs 217
8.2 Permutations et arrangements 217
8.3 Combinaisons 219
8.4 Relations de récurrence et dénombrement 223

Réponses **227**
Chapitre 1 227
Chapitre 2 243
Chapitre 3 247
Chapitre 4 255
Chapitre 5 259
Chapitre 6 262
Chapitre 7 265
Chapitre 8 275

Bibliographie **281**

Index **283**

Avant-propos

Le cours MAT215-*Logique et mathématiques discrètes pour l'optimisation* a été créé à l'automne 2021 pour le programme de baccalauréat en génie des opérations et de la logistique. Les présentes notes de cours ont été adaptées de celles du cours MAT210-*Logique et mathématiques discrètes*, un cours offert au baccalauréat en génie logiciel ainsi qu'au baccalauréat en génie des technologies de l'information.

La rédaction des notes de MAT210 a débuté tranquillement en 2006. À l'origine, elles n'étaient destinées qu'à résumer certaines notions présentées en classe. Depuis, ces notes ont été peu à peu bonifiées par des exemples variés et des exercices accompagnés de solutions détaillées. Cependant, pour un discours complet sur les différents sujets abordés, ainsi que pour avoir accès à davantage d'exercices, nous vous invitons à consulter le livre de référence *Discrete Mathematics and Its Applications* de Rosen, cité par [**?**] dans ce document, ou encore *Méthodes d'optimisation pour la gestion* de Norbert, Ouellet et Parent, cité par [**?**].

Plusieurs des exemples présentés ne sont pas solutionnés : on propose à l'équipe enseignante de les compléter en classe. Il en va de même des preuves des théorèmes.

Nous remercions notre collègue André Bordeleau pour les graphiques illustrant la croissance des fonctions dans le chapitre sur la notation grand-O.

Nous remercions aussi chaleureusement notre collègue Marie Forest qui a scruté attentivement les notes de MAT210 à l'été 2020, nous signalant plusieurs coquilles et nous donnant des suggestions fort pertinentes !

Finalement, nous vous remercions à l'avance de bien vouloir nous signaler les éventuelles erreurs détectées dans ces notes et de nous donner vos suggestions par courriel, à l'adresse suivante :
genevieve.savard@etsmtl.ca.

Geneviève Savard,
Anouk Bergeron-Brlek
et Xavier Provençal,
professeurs enseignants à l'ÉTS

Remarques concernant la version d'août 2022

Cette nouvelle version propose quelques nouveaux exemples et exercices dans les chapitres suivants, ce qui entraîne une modification de la numérotation :

- Chapitre 1, Exercices 1.17, 1.42, 1.49
- Chapitre 7
- Chapitre 8, Section 8.4

Pour le reste, il s'agit essentiellement de corrections mineures, de reformulations ou de précisions.

Remarques concernant la version de décembre 2022

Cette nouvelle version ne comporte que des modifications mineures visant à corriger les coquilles détectées. La numérotation des exemples et exercices demeure la même.

Remarques concernant la version de août 2023

Cette nouvelle version ne comporte que des modifications mineures visant à corriger les coquilles détectées. La numérotation des exemples et exercices demeure la même.

Chapitre 1

Logique et ensembles

1.1 Logique du premier ordre

1.1.1 Logique propositionnelle

Définition 1.1 : Proposition

Un énoncé qui est soit vrai, soit faux est appelé une **proposition**. La **valeur de vérité** d'une proposition est donc vrai (**vrai**) ou faux (**faux**).

Un énoncé qui n'est pas une proposition (comme un paradoxe, une phrase impérative ou interrogative) sera qualifié d'inacceptable.

Exemple 1.1 (à compléter en classe)
Déterminez si la phrase suivante est une proposition ou non. Si oui, précisez s'il s'agit d'une proposition vraie ou d'une proposition fausse.

(a) Le nombre 7 est pair.

(b) $2+3=6$

(c) Où est Montréal ?

(d) Dépêchez-vous.

(e) Il pleut actuellement quelque part sur la ville de Montréal.

(f) $x<5$

(g) Tous les nombres positifs sont strictement inférieurs à leur carré.

(h) Cette phrase est fausse.

(i) L'entier 19 est un nombre premier.

(j) Il existe un seul nombre premier p tel que $p+1$ est également premier.

(k) L'entier 427 741 est un nombre premier.

(l) Il existe une infinité de nombres premiers p tels que $p+2$ est également premier.

Il n'est pas nécessaire de connaître la valeur de vérité d'une proposition pour savoir qu'il s'agit bien d'une proposition. Dans l'exemple précédent, la phrase (k) est une proposition, car soit le nombre 427 741 est premier (et alors la proposition serait vraie), soit il n'est pas premier (et la proposition serait fausse). En fait, ce nombre n'est pas premier, car $521 \times 821 = 427\,741$. Il en va de même avec la phrase (l) qui est forcément soit vraie, soit fausse, même si dans l'état actuel de la science, on ne sait pas ce qui en est. Cette question est connue sous le nom de la **conjecture des nombres premiers jumeaux**.

On peut construire de nouvelles propositions à partir de propositions existantes en utilisant des connecteurs logiques. Ces nouvelles propositions forment ce que l'on appelle des **propositions composées**, et les propositions qui ne sont pas formées à partir de connecteur sont appelées **propositions simples**.

Définition 1.2 : Connecteurs logiques $\neg \quad \wedge \quad \vee \quad \oplus \quad \rightarrow \quad \leftrightarrow$

Soient p et q des propositions.

$\neg p$ L'énoncé « *il n'est pas vrai que p* » est une nouvelle proposition. On l'appelle la **négation** de p et on la note $\neg p$ (lire « non p »). La proposition $\neg p$ est vraie quand p est fausse et elle est fausse quand p est vraie.

$p \wedge q$ L'énoncé « *p et q* » est une nouvelle proposition. On l'appelle la **conjonction** de p et de q et on la note $p \wedge q$. La proposition $p \wedge q$ est vraie uniquement quand p et q sont vraies.

$p \vee q$ L'énoncé « *p ou q* » est une nouvelle proposition. On l'appelle la **disjonction** de p et de q et on la note $p \vee q$. La proposition $p \vee q$ est fausse uniquement quand p et q sont fausses. Elle est vraie quand une ou l'autre ou les deux propositions sont vraies.

$p \oplus q$ L'énoncé « *p ou exclusif q* » est une nouvelle proposition. On l'appelle la **disjonction exclusive** de p et de q et on la note $p \oplus q$. La proposition $p \oplus q$ est vraie uniquement quand une seule des propositions p et q est vraie. Ce connecteur peut être défini à partir des précédents par $(p \wedge \neg q) \vee (\neg p \wedge q)$.

$p \rightarrow q$ L'énoncé « *p implique q* » est une nouvelle proposition. On dit que c'est une **implication** ou **un énoncé conditionnel**, et on le note $p \rightarrow q$. La proposition $p \rightarrow q$ est fausse uniquement quand p est vraie et q est fausse. Ce connecteur peut être défini à partir des précédents par $\neg p \vee q$.
Voici quelques formulations différentes pour l'implication $p \rightarrow q$:
- « si p alors q »
- « q si p »
- « p est une condition suffisante pour q »
- « q est une condition nécessaire pour p »
- « p seulement si q »

$p \leftrightarrow q$ L'énoncé « *p si et seulement si q* » est une nouvelle proposition. On la note $p \leftrightarrow q$ et on l'appelle **biconditionnelle**. La proposition $p \leftrightarrow q$ est vraie uniquement quand p et q ont les mêmes valeurs de vérité. Ce connecteur peut être défini à partir des précédents par $(p \rightarrow q) \wedge (q \rightarrow p)$.

Résumons toute l'information fournie par la définition 1.2 grâce à une **table de vérité**.

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \oplus q$	$p \rightarrow q$	$p \leftrightarrow q$
V	**V**						
V	**F**						
F	**V**						
F	**F**						

Exemple 1.2 (à compléter en classe)
Voici quatre propositions simples à partir desquelles on peut construire des propositions composées.

c : « Julie étudie en génie de la construction. »
l : « Julie étudie en génie logiciel. »
t : « Julie étudie en génie des TI. »
m : « Julie suit le cours MAT210 cet été. »

Traduisez les propositions suivantes en utilisant les propositions simples c, l, t et m ainsi que les connecteurs logiques. *On ne demande pas de dire si les propositions sont vraies ou fausses.*

(a) Julie n'étudie ni en génie logiciel, ni en génie des TI.

(b) Julie étudie en génie logiciel ou en génie de la construction, mais elle n'étudie pas dans les deux programmes.

(c) Si Julie suit le cours de MAT210 cet été, alors elle étudie en génie logiciel ou en génie des TI.

(d) Si Julie étudie en génie logiciel ou en génie des TI, alors elle suit le cours de MAT210 cet été.

(e) Si Julie étudie en génie de la construction, alors elle ne suit pas le cours MAT210 cet été.

(f) Si Julie ne suit pas le cours MAT210 cet été, alors elle étudie en génie de la construction.

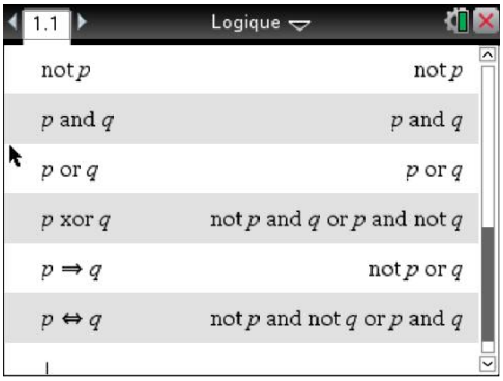
(g) Si Julie suit le cours de MAT210 cet été et n'étudie pas en génie des TI, alors elle étudie en génie logiciel.

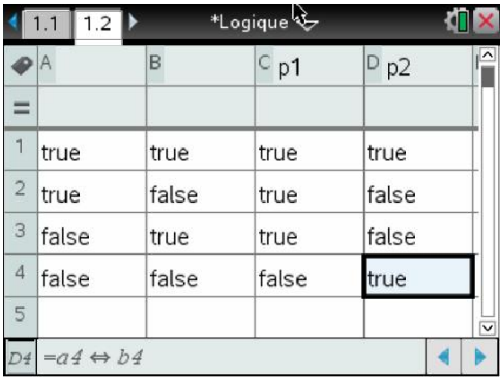
(h) Julie suit le cours de MAT210 cet été seulement si elle étudie en génie des TI ou en génie locigiel.

(i) Savoir que Julie étudie en génie de la construction est une information *suffisante* pour conclure que Julie ne suit pas le cours de MAT210 cet été.

(j) Il est nécessaire que Julie n'étudie pas en construction pour qu'elle suive le cours de MAT210 cet été.

Le symbole **V** (vrai) est parfois désigné par **T** (true) ou 1. Le symbole **F** (faux, false) est parfois désigné par 0. Les écrans suivants montrent la syntaxe utilisée par le logiciel et la calculatrice Nspire pour les connecteurs logiques.





Priorité des connecteurs logiques

La priorité des opérations arithmétiques est une convention qui dicte dans quel ordre effectuer les opérations. Elle permet de réduire le nombre de parenthèses d'une expression sans diminuer sa clarté. Par exemple, on peut omettre les quatre parenthèses dans l'expression $2+(3\times(2^2))$.

$$2+3\times 2^2 = 2+(3\times(2^2)) = 2+(3\times 4) = 2+12 = 14$$

Il en va de même avec la priorité des connecteurs logiques. Voici les listes de priorité des opérations arithmétiques (un petit rappel!) et des connecteurs logiques.

Priorité des opérations arithmétiques	
1	Parenthèses, de l'intérieur vers l'extérieur.
2	Exposants.
3	$\times$ $\div$
4	$+$ $-$

Priorité des connecteurs logiques	
1	Parenthèses, de l'intérieur vers l'extérieur.
2	$\neg$
3	$\wedge$
4	$\vee$ $\oplus$
5	$\rightarrow$
6	$\leftrightarrow$

Ainsi, $p\vee q\rightarrow\neg r$ est équivalent à $(p\vee q)\rightarrow(\neg r)$. Par ailleurs, bien que certaines parenthèses soient inutiles avec la priorité des connecteurs, il arrive souvent qu'on les ajoute pour un maximum de clarté.

Exemple 1.3 (à compléter en classe)

Ajoutez toutes les parenthèses possibles dans les expressions suivantes pour les clarifier sans en changer la valeur de vérité.

(a) $\neg p\vee q\wedge r$

(b) $p\vee q\wedge r$

(c) not p and not q or p and q

(d) not p and q or p and not q

(e) not a and b and not c or not d

Exemple 1.4 (à compléter en classe)
Ôtez les parenthèses inutiles dans les expressions suivantes.

(a) $\big(p \wedge q\big) \rightarrow (p \vee q)$

(b) $(q \rightarrow p) \vee \big(\big(\neg p\big) \wedge q\big)$

Exemple 1.5 (à compléter en classe)
Construisez la table de vérité de la proposition composée suivante. De plus, trouvez une expression plus simple ayant la même table de vérité.

(a) $(p \vee q) \oplus (p \wedge q)$

p	q	$p \vee q$	$p \wedge q$	$(p \vee q) \oplus (p \wedge q)$
V	**V**			
V	**F**			
F	**V**			
F	**F**			

(b) $(p \leftrightarrow q) \oplus (\neg p \leftrightarrow q)$

p	q	$\neg p$	$p \leftrightarrow q$	$\neg p \leftrightarrow q$	$(p \leftrightarrow q) \oplus (\neg p \leftrightarrow q)$
V	**V**				
V	**F**				
F	**V**				
F	**F**				

Distinctions entre le langage courant et le langage mathématique.

Dans le langage courant, le « ou » est souvent considéré exclusif, comme dans la phrase « vous avez droit à une soupe ou à une salade avec votre repas » qui signifie « vous avez droit à une soupe ou à une salade avec votre repas, *mais pas les deux.* »
Cependant, **en mathématique**, **on considère tous les « ou » comme des disjonctions inclusives** : « a ou b » sera vraie si a ou b ou (a et b) sont vraies.
De même, le « si » du langage courant est souvent utilisé au lieu du « si et seulement si », comme dans l'exemple « Tu auras un cadeau si tu es sage » qui signifie en fait « tu auras un cadeau si *et seulement* si tu es sage ». Ou encore « je vous embaucherai si vous faites votre stage ici », qui peut sous-entendre « je vous embaucherai si et seulement si vous faites votre stage ici ». Dans ce recueil de notes, le « si » correspondra toujours à une implication et non à une biconditionnelle.

Définition 1.3 : Réciproque, contraposée et inverse

La **réciproque** de la proposition $p \rightarrow q$ est la proposition $q \rightarrow p$.
La **contraposée** de la proposition $p \rightarrow q$ est la proposition $\neg q \rightarrow \neg p$.
L'**inverse** de la proposition $p \rightarrow q$ est la proposition $\neg p \rightarrow \neg q$.

Exemple 1.6 (à compléter en classe)
Soit les propositions

v : « Je viens à l'ÉTS à vélo. »
b : « Il fait beau. »

(a) Écrivez les propositions suivantes en utilisant les lettres v, b et les connecteurs logiques.

P1 : « Je viens à l'ÉTS à vélo s'il fait beau. »

P2 : « Si je ne viens pas à l'ÉTS à vélo, alors il ne fait pas beau. »

P3 : « Je viens à l'ÉTS à vélo seulement s'il fait beau. »

P4 : « S'il ne fait pas beau, je ne viens pas à l'ÉTS à vélo. »

P5 : « Je viens à l'ÉTS à vélo si et seulement s'il fait beau. »

(b) Indiquez laquelle des propositions est la réciproque de la proposition P1.

(c) Indiquez laquelle des propositions est la contraposée de P1.

(d) Indiquez laquelle des propositions est l'inverse de P1.

Exemple 1.7 (à compléter en classe)

Considérons le détecteur d'erreurs très rudimentaire suivant: lors de la transmission d'un message, disons par paquets de 7 bits, on ajoute un bit de parité au train de bits: 0 si la somme des bits est paire et 1 si elle est impaire. Par exemple,

`111 0001` devient **0**`111 0001` et `101 0001` devient **1**`101 0001`

Lors de la réception, on vérifie si le bit de parité correspond ou non à la parité du message reçu. Si le bit de parité n'est pas bon, alors on sait qu'il y a eu au moins une erreur au cours de la transmission. Soit les propositions

b : « Le bit de parité est bon ».
e : « Il y a eu au moins une erreur au cours de la transmission. »
i : « Il y a eu un nombre impair d'erreurs au cours de la transmission. »

(a) Écrivez les propositions composées suivantes en utilisant les propositions b, e et i et les connecteurs logiques.

P1 : « Si le bit de parité n'est pas bon, alors il y a eu au moins une erreur au cours de la transmission. »

P2 : « Si le bit de parité est bon, alors il n'y a pas eu d'erreur au cours de la transmission. »

P3 : « S'il n'y a pas eu d'erreur au cours de la transmission, alors le bit de parité est bon. »

P4 : « S'il y a eu au moins une erreur au cours de la transmission, alors le bit de parité n'est pas bon. »

P5 : « S'il y a eu un nombre impair d'erreurs au cours de la transmission, alors le bit de parité n'est pas bon. »

(b) Déterminez les propositions qui sont vraies parmi P1 à P5.

(c) Indiquez laquelle des propositions P2 à P5 est la réciproque de P1.

(d) Indiquez laquelle des propositions P2 à P5 est la contraposée de P1.

Exemple 1.8 (à compléter en classe)

L'extrait de code suivant fait intervenir les variables booléennes `p`, `q` et `r`. Chacune de ces variables peut prendre les valeurs **vrai** ou **faux**. Pour chaque bloc indiqué, donnez toutes les valeurs possibles pour `p`, `q` et `r` au moment où le bloc est atteint.

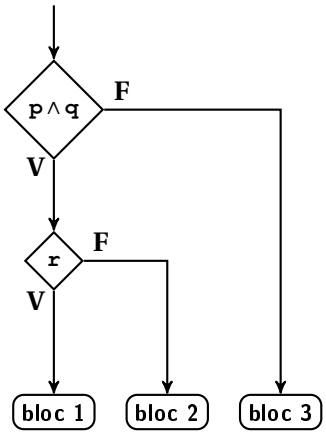
Notation (utilisée notamment en C/C++, C# et Java) :

- l'opérateur de conjonction $\wedge$ est noté `&&`,
- l'opérateur de disjonction $\vee$ est noté `||`.

```
if ( p && q ) {
 if( r ) {
    // bloc (1)
 } else {
    // bloc (2)
  }
} else {
 // bloc (3)
}
```

Solution :

Il est utile, mais pas nécessaire, de représenter le code du programme par un organigramme de programmation puis de compléter une table de vérité correspondant aux conditions menant à chacun des blocs.



p	q	r				
V	**V**	**V**				
V	**V**	**F**				
V	**F**	**V**				
V	**F**	**F**				
F	**V**	**V**				
F	**V**	**F**				
F	**F**	**V**				
F	**F**	**F**				

Exercices

1.1 Les extraits de code suivants font intervenir les variables booléennes `p`, `q` et `r`. Chacune de ces variables peut prendre les valeurs **vrai** ou **faux**. Pour chaque bloc indiqué, donnez toutes les valeurs possibles pour `p`, `q` et `r` au moment où le bloc est atteint.

Notation :

- l'opérateur de conjonction $\wedge$ est noté `&&`,
- l'opérateur de disjonction $\vee$ est noté `||`.

(a)

```
if ( p && q || r ) {
  // bloc (1)
} else {
  // bloc (2)
}
```

(b)

```
if ( p ) {
  if ( q ) {
    if ( r ) {
      // bloc (1)
    }
  } else {
    // bloc (2)
  }
}
```

(c)

```
if ( p && q ) {
  // bloc (1)
} else {
  while ( r ) {
    // bloc (2)
  }
  // bloc (3)
}
```

1.2 Soit p et q les propositions

p : « La température est au-dessous de 0° Celsius. »
q : « L'eau gèle. »

Écrivez les propositions suivantes à l'aide des propositions p et q et des connecteurs logiques.

(a) La température est au-dessous de 0° Celsius et l'eau gèle.

(b) Si l'eau gèle, alors la température est au-dessous de 0° Celsius.

(c) La température est au-dessous de 0° Celsius, mais l'eau ne gèle pas.

(d) La température est au-dessous de 0° Celsius ou l'eau gèle.

(e) Si la température est au-dessous de 0° Celsius, alors l'eau gèle.

(f) La température est de 0° Celsius ou plus, mais l'eau ne gèle pas.

(g) Il est nécessaire et suffisant que la température soit au-dessous de 0° Celsius pour que l'eau gèle.

(h) Si la température est de 0° Celsius ou plus, alors l'eau ne gèle pas.

1.3 Parmi les phrases de l'exercice 1.2, y en a-t-il une qui soit la réciproque de l'énoncé (e)? Sa contraposée? Son inverse? Si oui, identifiez ces phrases.

1.4 Les phrases suivantes sont-elles vraies si on les complète avec *« nécessaire et suffisante »*? Sinon, lequel des termes parmi *« nécessaire »* ou *« suffisante »* faut-il choisir pour que la phrase soit vraie?

(a) La condition $x > 5$ est _____ pour que le nombre x soit positif[1].
(b) La condition $x > -5$ est _____ pour que x soit positif.
(c) La condition $x > -1$ est _____ pour que l'entier x soit positif.
(d) Pour que le nombre x soit premier, il est _____ qu'il soit supérieur ou égal à 2.
(e) Pour qu'un triangle soit rectangle, il est _____ que ses côtés mesurent respectivement 3, 4 et 5 centimètres.

1.5 Réécrivez chacune des phrases de l'exercice précédent sous la forme « il est nécessaire que… pour que … », « il suffit que… pour que … » ou encore « il faut et il suffit que … pour que … ».

1.6 Vrai ou faux?

(a) Il est *nécessaire* que les quatre angles d'un quadrilatère Q mesurent 90° pour que Q soit un carré.
(b) Il *suffit* que le quadrilatère Q ait deux paires de côtés parallèles pour qu'il soit un rectangle.

1. On considère que 0 est un nombre positif et négatif.

1.7 Soit p et q les propositions

p : « Vous avez fait tous les exercices du cours. »
q : « Vous avez obtenu la note A+. »

Écrivez les propositions suivantes à l'aide de p, de q et des connecteurs logiques.

(a) Il est nécessaire que vous ayez fait tous les exercices du cours pour avoir obtenu la note A+.

(b) Vous avez fait tous les exercices du cours, mais vous n'avez pas obtenu la note A+.

(c) Si vous avez fait tous les exercices du cours, alors vous avez obtenu la note A+.

(d) Si vous n'avez pas fait tous les exercices du cours, alors vous n'avez pas obtenu la note A+.

(e) Vous avez obtenu la note A+, mais vous n'avez pas fait tous les exercices du cours.

(f) Si vous avez obtenu la note A+, alors vous avez fait tous les exercices du cours.

(g) Si vous n'avez pas obtenu la note A+, on peut en déduire que vous n'avez pas fait tous les exercices du cours.

(h) Vous n'avez pas fait tous les exercices du cours et vous n'avez pas obtenu la note A+.

(i) Vous avez fait tous les exercices du cours ou vous n'avez pas obtenu la note A+.

(j) Vous avez obtenu la note A+ si et seulement si vous avez fait tous les exercices du cours.

1.8 Parmi les phrases de l'exercice 1.7, identifiez la réciproque, la contraposée et l'inverse de l'énoncé (c) : « Si vous avez fait tous les exercices du cours, alors vous avez obtenu la note A+. »

1.9 Déterminez si p est une condition *nécessaire et suffisante* pour q. Dans le cas contraire, déterminez si p est *suffisante* pour q ou si elle est *nécessaire* pour q. Les domaines des variables sont sous-entendus.

(a) p : « x est un nombre supérieur ou égal à 0 », q : « x est le carré d'un nombre réel. »

(b) p : « t est un triangle rectangle »,
q : « t est un triangle dont un angle mesure 60° et un autre mesure 30°. »

(c) p : « t est un triangle rectangle », q : « t est un triangle dont un des angles mesure 90°. »

(d) p : « t est une partie de tic-tac-toe terminée », q : « t comporte au moins 5 symboles X ou O. »

(e) p : « t est une partie de tic-tac-toe terminée », q : « t comporte au moins 9 symboles X ou O. »

(f) p : « x suit le cours de Mathématiques discrètes de l'ÉTS et fait le présent exercice »,
q : « x étudie à l'ÉTS ou aide gentiment un étudiant de l'ÉTS. »

(g) p : « x est un carré », q : « x est un rectangle. »

1.10 Construisez une table de vérité pour les propositions suivantes.

(a) $(p \to q) \wedge (\neg q)$

p	q	$p \to q$	$\neg q$	$(p \to q) \wedge \neg q$
V	**V**			
V	**F**			
F	**V**			
F	**F**			

(b) $(p \to q) \leftrightarrow (\neg p \vee q)$

p	q	$p \to q$	$\neg p \vee q$	$(p \to q) \leftrightarrow (\neg p \vee q)$
V	**V**			
V	**F**			
F	**V**			
F	**F**			

(c) $(p \to q) \to r$

p	q	r	$p \to q$	$(p \to q) \to r$
V	**V**	**V**		
V	**V**	**F**		
V	**F**	**V**		
V	**F**	**F**		
F	**V**	**V**		
F	**V**	**F**		
F	**F**	**V**		
F	**F**	**F**		

(d) $p \to (q \to r)$

p	q	r		
V	**V**	**V**		
V	**V**	**F**		
V	**F**	**V**		
V	**F**	**F**		
F	**V**	**V**		
F	**V**	**F**		
F	**F**	**V**		
F	**F**	**F**		

1.1.2 Équivalences propositionnelles

Définition 1.4 : Tautologie, contradiction, contingence

Une proposition composée qui est toujours vraie, quelle que soit la valeur de vérité des propositions qui la composent, est appelée une **tautologie**.

Une proposition composée qui est toujours fausse est appelée une **contradiction**.

Une proposition qui n'est ni une tautologie ni une contradiction est appelée une **contingence**.

Exemple 1.9 (à compléter en classe)
Déterminez si chacune des propositions suivantes est une tautologie, une contradiction ou une contingence.

(a) $p \vee \neg p$

(b) $p \rightarrow q$

(c) $p \wedge \neg p$

Définition 1.5 : Équivalence de propositions

Les propositions p et q sont dites **logiquement équivalentes** si la proposition $p \leftrightarrow q$ est une tautologie. Ainsi, deux propositions sont logiquement équivalentes si elles ont la même table de vérité, c'est-à-dire la même valeur de vérité dans tous les cas possibles.

Les notations $p \equiv q$ et $p \Leftrightarrow$ q signifient que p et q sont logiquement équivalentes.

Exemple 1.10 (à compléter en classe)
Vérifiez l'équivalence suivante à l'aide d'une table de vérité.

$$p \rightarrow q \equiv \neg p \vee q$$

Solution :
Afin de vérifier l'équivalence, il faut s'assurer que la proposition $(p \rightarrow q) \leftrightarrow (\neg p \vee q)$ est toujours vraie. Construisons sa table de vérité.

p	q	$p \rightarrow q$	$\neg p \vee q$	$(p \rightarrow q) \leftrightarrow (\neg p \vee q)$
V	**V**			
V	**F**			
F	**V**			
F	**F**			

Exemple 1.11 (à compléter en classe)
Vérifiez l'équivalence suivante.

$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$

Solution :
Construisons la table de vérité de chacune des deux propositions afin de montrer qu'elles ont la même valeur de vérité dans tous les cas possibles.

p	q	$p \vee q$	$\neg(p \vee q)$
V	**V**		
V	**F**		
F	**V**		
F	**F**		

p	q	$\neg p$	$\neg q$	$\neg p \wedge \neg q$
V	**V**			
V	**F**			
F	**V**			
F	**F**			

Pour gagner du temps, on note les équivalences fréquemment utilisées dans une table et on leur donne un nom ou un numéro afin d'y faire référence. Par exemple, l'équivalence de l'exemple 1.11 est appelée *loi de De Morgan*, en l'honneur du mathématicien britannique Auguste De Morgan qui vécut au 19^{e} siècle. On la retrouve à la 8^{e} ligne de la table 1. Quant à l'équivalence de l'exemple 1.10, elle se retrouve à la première ligne de la table 2. Pour gagner du temps, on note les équivalences fréquemment utilisées dans une table et on leur donne un nom ou un numéro afin d'y faire référence. Par exemple, l'équivalence de l'exemple 1.11 est appelée *loi de De Morgan*, en l'honneur du mathématicien britannique Auguste De Morgan qui vécut au 19^{e} siècle. On la retrouve à la 8^{e} ligne de la table 1. Quant à l'équivalence de l'exemple 1.10, elle se retrouve à la première ligne de la table 2.

TABLE 1 Équivalences logiques

1	$p \wedge \mathbf{V} \equiv p$ $p \vee \mathbf{F} \equiv p$	Identité
2	$p \vee \mathbf{V} \equiv \mathbf{V}$ $p \wedge \mathbf{F} \equiv \mathbf{F}$	Domination
3	$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotence
4	$\neg(\neg p) \equiv p$	Double négation
5	$p \wedge q \equiv q \wedge p$ $p \vee q \equiv q \vee p$	Commutativité
6	$(p \vee q) \vee r \equiv p \vee (q \vee r)$ $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	Associativité
7	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	Distributivité
8	$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	Lois de De Morgan
9	$p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$	Absorption
10	$p \vee \neg p \equiv \mathbf{V}$ $p \wedge \neg p \equiv \mathbf{F}$	Négation

TABLE 2 Équivalences logiques (implications)

1	$p \to q \equiv \neg p \vee q$
2	$p \to q \equiv \neg q \to \neg p$
3	$p \vee q \equiv \neg p \to q$
4	$p \wedge q \equiv \neg(p \to \neg q)$
5	$\neg(p \to q) \equiv p \wedge \neg q$
6	$(p \to q) \wedge (p \to r) \equiv p \to (q \wedge r)$
7	$(p \to r) \wedge (q \to r) \equiv (p \vee q) \to r$
8	$(p \to q) \vee (p \to r) \equiv p \to (q \vee r)$
9	$(p \to r) \vee (q \to r) \equiv (p \wedge q) \to r$

TABLE 3 Équivalences logiques (biconditionnelles)

1	$p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$
2	$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$
3	$p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$
4	$p \leftrightarrow q \equiv \neg(p \wedge \neg q) \wedge \neg(\neg p \wedge q)$
5	$\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$

Exemple 1.12

Vérifiez que la proposition

$$\neg(p \to q) \to \neg q$$

est une tautologie

(a) à l'aide d'une table de vérité ;

(b) sans l'aide d'une table de vérité, en utilisant les tables d'équivalences ;

Solution :

(a) À l'aide d'une table de vérité.

p	q	$p \to q$	$\neg(p \to q)$	$\neg q$	$\neg(p \to q) \to \neg q$
V	**V**	**V**	**F**	**F**	**V**
V	**F**	**F**	**V**	**V**	**V**
F	**V**	**V**	**F**	**F**	**V**
F	**F**	**V**	**F**	**V**	**V**

(b) En utilisant les tables d'équivalences

$\neg(p \to q) \to \neg q$	$\equiv$	$\neg(\neg(p \to q)) \vee \neg q$	Table 2.1
	$\equiv$	$(p \to q) \vee \neg q$	Double négation
	$\equiv$	$(\neg p \vee q) \vee \neg q$	Table 2.1
	$\equiv$	$\neg p \vee (q \vee \neg q)$	Associativité
	$\equiv$	$\neg p \vee \mathbf{V}$	Négation
	$\equiv$	$\mathbf{V}$	Domination

Exercices

1.11 À l'aide d'une table de vérité, déterminez si la proposition suivante est une tautologie, une contingence ou une contradiction.

(a) $(p \vee p) \leftrightarrow \neg(p \wedge p)$

(b) $(p \oplus q) \to \neg(p \wedge q)$

(c) $(p \vee (q \wedge r)) \leftrightarrow ((p \vee q) \wedge r)$

(d) $\neg(p \to q) \to p$

1.12 Vérifiez les équivalences suivantes à l'aide d'une table de vérité.

(a) $p \to q \equiv \neg q \to \neg p$

(b) $\neg(p \vee q) \equiv \neg p \wedge \neg q$

1.13 Utilisez les lois de De Morgan pour formuler la négation des énoncés suivants.

(a) Ce programme est rapide et efficace.

(b) Le programme doit être en C+ ou en Java.

1.14 Montrez que, pour les langages Java ou C, les expressions suivantes sont équivalentes.

```
!(x < 10 || x > 20)      et      x >= 10 && x <= 20
```

Rappel:

symbole en Java ou C	symbole logique
!	$\neg$
>=	$\geq$
<=	$\leq$
||	$\vee$
&&	$\wedge$

1.15 Prouvez que

$$\neg(p \vee (\neg p \wedge q)) \equiv \neg p \wedge \neg q$$

(a) à l'aide d'une table de vérité ;

(b) en utilisant les tables d'équivalences ;

(c) en entrant l'expression sur Nspire.

Propositions équivalentes ou non ?

Pour démontrer que les propositions **ne sont pas** équivalentes, il suffit de fournir des valeurs de p, q et r pour lesquelles elles diffèrent.
Pour démontrer que les propositions **sont** équivalentes, on peut procéder de l'une des trois façons suivantes.

1. Fournir leur table de vérité.
2. Utiliser les tables d'équivalence logique de la page 14.
3. Formuler une explication en mots qui montre que les deux propositions sont vraies, ou encore que les deux sont fausses, exactement pour les mêmes combinaisons de valeur de vérité des variables propositionnelles.

1.16 Déterminez si les propositions suivantes sont logiquement équivalentes (consultez l'encadré ci-dessus).

(a) $(p \to q) \to r$ et $p \to (q \to r)$

(b) $(p \to r) \vee (q \to r)$ et $(p \wedge q) \to r$ (sans utiliser la table 2, ligne 9).

1.17 Vérifiez les équivalences suivantes à l'aide des tables d'équivalences.

(a) $(\neg q \to p) \to (r \vee p) \equiv (\neg p \wedge q) \to r$

(b) $((p \vee q) \wedge (\neg p \vee r)) \to (q \vee r) \equiv$ **vrai** (tautologie)

1.18 Une entreprise doit recruter une petite équipe pour réaliser un projet à l'extérieur du pays. Suite aux entrevues de 8 postulants, à l'analyse des compétences et des besoins, ainsi qu'aux questions de bonne entente, les contraintes suivantes ont été identifiées.

1. Au moins un des postulants numéro 2, 4 ou 5 doit être embauché, car eux seuls ont des connaissances en électricité.
2. Si le postulant 2 est embauché ou le postulant 4 est embauché, alors le postulant 5 ne doit pas l'être.
3. Si les postulants 2 et 4 sont embauchés tous les deux, alors le postulant 5 ne doit pas l'être.
4. Les postulants 3 et 5 forment un couple : ils accepteront l'affectation seulement si les deux sont embauchés.
5. Il est impossible d'embaucher à la fois les postulants 2 et 5, et il est impossible d'embaucher à la fois les postulants 4 et 5.
6. Il est impossible d'embaucher le trio de postulants 2, 4 et 5 : on peut en choisir un, deux ou aucun, mais pas les trois.

7. Il faut absolument embaucher au moins 2 personnes parmi les postulants 1 à 4.
8. Il faut embaucher les postulants 1 et 3, ou embaucher le postulant 2 et au moins un des postulants parmi 1, 3 et 4, ou encore embaucher le postulant 4 et au moins un des postulants parmi 1, 2 et 3.

(a) En utilisant les propositions p_1 à p_8, où

$$p_i \ : \ \text{« le postulant } i \text{ est embauché »},$$

traduisez chacune des contraintes à l'aide de connecteurs logiques.

(b) Déterminez quelles contraintes sont équivalentes et démontrez-le à l'aide des règles des tables 1 à 3 de la page 16

1.1.3 Prédicats et quantificateurs

Un énoncé contenant une ou plusieurs variables tel que

$$x < 10 \qquad \text{ou} \qquad x + 2 = 7 - y$$

n'est pas une proposition puisque, tant que la valeur de x ou y n'est pas connue, on ne peut dire s'il est vrai ou faux.

Terminologie

Dans l'énoncé « $x < 10$ », x est le **sujet**, et « est inférieur à 10 » est le **prédicat**. Notons $P(x)$ l'énoncé $x < 10$. On dit que P est une **fonction propositionnelle**.

Une fonction propositionnelle $P(x)$ prend la valeur vrai ou faux quand x est précisé. Par exemple :

- $P(8)$ est une proposition vraie. On écrira parfois $P(8)$ est vrai (au masculin, en sous-entendant l'énoncé est vrai », ou même $P(8) \equiv \mathbf{V}$).
- $P(13)$ est une proposition fausse.
- $P(\text{Julie})$ n'est pas une proposition, car Julie n'est pas une valeur possible pour la variable x.

L'ensemble des valeurs possibles pour la variable x est appelé **univers du discours**, ou **domaine** de la fonction P.

Définition 1.6 : Quantificateurs

$\forall$: quantificateur universel $\quad$ $\exists$: quantificateur existentiel

La proposition $\forall x P(x)$ signifie « Pour toutes les valeurs de x dans l'univers du discours, $P(x)$ ».
Ou encore « Quel que soit x (dans l'univers du discours), $P(x)$. »

La proposition $\exists x P(x)$ signifie « Il existe un élément x de l'univers du discours tel que $P(x)$ ».
Ou encore « Il y a au moins un x (dans l'univers du discours) tel que $P(x)$. »
Ou encore « Pour un certain x (dans l'univers du discours), $P(x)$. »

Notation. Certains auteurs mettent une virgule avant la fonction propositionnelle, surtout quand celle-ci est composée. Par exemple : $\forall x, \big(P_1(x) \to P_2(x) \vee P_3(x)\big)$. Par ailleurs, si l'ensemble U n'a pas déjà été identifié, on peut préciser que la variable x prendra des valeurs dans l'ensemble U ainsi : $\exists x \in U, P(x)$.

Lorsque l'univers du discours est un ensemble fini $\{x_1, x_2, \dots x_n\}$, on a les équivalences logiques suivantes :

$$\forall x\, P(x) \equiv P(x_1) \wedge P(x_2) \wedge \cdots \wedge P(x_n),$$

$$\exists x\, P(x) \equiv P(x_1) \vee P(x_2) \vee \cdots \vee P(x_n).$$

La quantification universelle $\forall x\, P(x)$ est vraie quand $P(x)$ est vraie pour toutes les valeurs de x dans l'univers du discours U. Elle est donc fausse s'il existe un x de U pour lequel $P(x)$ est fausse. Un tel élément est appelé un **contre-exemple** de $\forall x\, P(x)$.

La quantification existentielle $\exists x\, P(x)$ est vraie s'il existe au moins une valeur x dans l'univers du discours telle que $P(x)$ est vraie. Elle est fausse si $P(x)$ est fausse pour toutes les valeurs possibles de x.

Ainsi, pour prouver qu'un énoncé de la forme $\forall x P(x)$ est vrai, fournir un exemple de x tel que $P(x)$ est vrai ne suffit pas. Il faut montrer que la proposition $P(x)$ est vraie pour toutes les valeurs de x, ce qui peut s'avérer particulièrement **difficile lorsque U est un ensemble infini**. Il en va de même lorsqu'on veut prouver qu'un énoncé de la forme $\exists x P(x)$ est faux. Le tableau 1.1 résume les différents cas possibles.

Pour prouver que	est **vrai**	est **faux**
$\exists x\, P(x)$	il suffit de fournir **un exemple** : un x de U tel que $P(x)$ est vrai.	il faut fournir un argument général pour montrer que $P(x)$ est faux quelque soit x de U.
$\forall x P(x)$	il faut fournir un argument général pour montrer que $P(x)$ est vrai quelque soit x de U.	il suffit de fournir **un contre-exemple** : un x de U tel que $P(x)$ est faux.

Tableau 1.1 Comment prouver qu'un énoncé quantifié est vrai ou faux quand l'univers du discours U est infini.

Exemple 1.13 (à compléter en classe)
Si l'univers du discours est l'ensemble des nombres réels et

$P(x)$ désigne « $x \geq 0$ »

$Q(x)$ désigne « x est un nombre premier »

$R(x)$ désigne « $3^x + 4^x = 5^x$ »

$S(x)$ désigne « $x \geq 100$ »,

dites si chacun des énoncés suivants est une proposition vraie, une proposition fausse ou n'est pas une proposition. Donnez un exemple ou un contre-exemple le cas échéant. Dans le cas contraire, indiquez qu'un argument général est requis.

(a) $\forall x\, P(x)$

(b) $\forall x\, \neg P(x)$

(c) $\forall x\, P(x^2)$

(d) $\exists x\, P(x)$

(e) $\exists x\, \neg P(x)$

(f) $\exists x\, Q(x)$

(g) $\exists x\, Q(x^2)$

(h) $\forall x\, R(x)$

(i) $P(x)$

(j) $\forall x \left(S(x) \to P(x)\right)$

(k) $\left(\forall x\, P(x)\right) \to \left(\forall x\, S(x)\right)$

(l) $\forall x\, S(x+100)$

(m) $\forall x\, S(x^2+100)$

Théorème 1.1 : Lois de De Morgan pour les quantificateurs

$$\neg\exists x P(x) \equiv \forall x \neg P(x) \qquad \neg\forall x P(x) \equiv \exists x \neg P(x)$$

Exemple 1.14 (à compléter en classe)
Si l'univers du discours est l'ensemble des employés de l'ÉTS et $M(x)$ désigne l'énoncé « L'employé x peut modifier les fichiers du répertoire U », traduisez clairement les propositions suivantes à l'aide des quantificateurs.

(a) Tous les employés de l'ÉTS peuvent modifier les fichiers du répertoire U.

(b) Il est faux que tous les employés de l'ÉTS peuvent modifier les fichiers du répertoire U.

(c) Au moins un employé de l'ÉTS peut modifier les fichiers du répertoire U.

(d) Il est faux qu'au moins un employé de l'ÉTS peut modifier les fichiers du répertoire U.

(e) Aucun employé de l'ÉTS ne peut modifier les fichiers du répertoire U.

(f) Au moins un employé de l'ÉTS ne peut pas modifier les fichiers du répertoire U.

De plus, déterminez les propositions ci-dessus qui sont équivalentes.

Exemple 1.15 (à compléter en classe)
Si l'univers du discours est l'ensemble des billes contenues dans un bol, et si

$G(x)$ désigne « la bille x est grosse » $\quad$ $R(x)$ désigne « la bille x est rouge »
$J(x)$ désigne « la bille x est jaune » $\quad$ $B(x)$ désigne « la bille x est bleue »

traduisez clairement les propositions suivantes en prenant soin de bien formuler les phrases.

(a) $\forall x \left(R(x) \vee J(x)\right)$

(b) $\left(\forall x\, R(x)\right) \vee \left(\forall x\, J(x)\right)$

(c) Les propositions (a) et (b) sont-elles équivalentes ?

(d) $\exists x\, B(x)$

(e) $\neg\,(\exists x\, B(x))$

(f) Utilisez le quantificateur universel $\forall$ pour écrire une proposition équivalente à la précédente.

(g) $\neg\,(\forall x\, R(x))$

(h) Utilisez le quantificateur existentiel $\exists$ pour écrire une proposition équivalente à la précédente.

(i) $\forall x\,\big(G(x) \to B(x)\big)$

(j) $\exists x\,\big(G(x) \wedge B(x)\big)$

(k) $\big(\exists x\, G(x)\big) \wedge \big(\exists x\, B(x)\big)$

(l) Les deux propositions précédentes sont-elles équivalentes ?

(m) Les deux propositions suivantes sont-elles équivalentes ?

$$\big(\exists x\, R(x)\big) \vee \big(\exists x\, J(x)\big) \qquad \text{et} \qquad \exists x\,\big(R(x) \vee J(x)\big)$$

(n) Les deux propositions suivantes sont-elles équivalentes ?

$$\big(\forall x\, R(x)\big) \wedge \big(\forall x\, G(x)\big) \qquad \text{et} \qquad \forall x\,\big(R(x) \wedge G(x)\big)$$

Les équivalences des deux dernières sous-questions se retrouvent dans la table 4 : Équivalences logiques (énoncés quantifiés).

TABLE 4 Équivalences logiques (énoncés quantifiés)

1	$\forall x \left(P(x) \wedge Q(x)\right) \equiv \forall x\, P(x) \wedge \forall x\, Q(x)$
2	$\exists x \left(P(x) \vee Q(x)\right) \equiv \exists x\, P(x) \vee \exists x\, Q(x)$
3	$\neg\exists x\, P(x) \equiv \forall x \neg P(x)$
4	$\neg\forall x\, P(x) \equiv \exists x \neg P(x)$

Exercices

1.19 Soit $P(x)$ l'énoncé « $x = x^2$ ». Si l'univers du discours est l'ensemble des nombres réels, quelles sont les valeurs de vérité des propositions suivantes ?
De plus, pour chacun des énoncés, dites si un exemple ou un contre-exemple suffit à justifier votre réponse ou si un argument général serait requis.

(a) $\exists x\, P(x)$

(b) $\forall x\, P(x)$

(c) $\exists x \neg P(x)$

(d) $\forall x \neg P(x)$

1.20 Supposons que l'univers du discours d'une fonction propositionnelle $P(x)$ est l'ensemble des entiers $\{0,1,2\}$. Écrivez chacune des propositions suivantes sans avoir recours à un quantificateur.

(a) $\exists x\, P(x)$

(b) $\forall x\, P(x)$

(c) $\neg\exists x\, P(x)$

(d) $\neg\forall x\, P(x)$

(e) $\exists x \neg P(x)$

(f) $\forall x \neg P(x)$

1.21 Associez chacune des phrases à sa traduction ou à une formulation équivalente utilisant les quantificateurs et les prédicats suivants, avec l'ensemble des humains comme univers du discours, $I(x)$: « x est un ingénieur », $P(x)$: « x sait programmer ». *Le même choix de réponse peut être utilisé plus d'une fois.*

(a) Il existe au moins un ingénieur qui sait programmer.

(b) Tous les ingénieurs savent programmer.

(c) Il existe un ingénieur qui ne sait pas programmer.

(d) Aucun humain ne sait programmer.

(e) Aucun ingénieur ne sait programmer.

(f) Il n'existe aucun ingénieur qui ne sait pas programmer.

(g) Il existe quelqu'un qui sait programmer, mais n'est pas ingénieur.

(h) Les ingénieurs ne savent pas programmer.

(i) Les programmeurs ne sont pas tous ingénieurs.

1. $\exists x \left(I(x) \wedge P(x)\right)$
2. $\exists x \left(I(x) \vee P(x)\right)$
3. $\exists x \left(P(x) \wedge \neg I(x)\right)$
4. $\exists x \left(I(x) \wedge \neg P(x)\right)$
5. $\exists x\, I(x)$
6. $\exists x \neg I(x)$
7. $\forall x \neg P(x)$
8. $\forall x \left(I(x) \to P(x)\right)$
9. $\forall x \left(\neg I(x) \vee \neg P(x)\right)$

1.22 Reprenons le contexte de l'exercice précédent, où l'univers du discours est cette fois l'ensemble des six employés d'une petite entreprise. Pour chacune des propositions, dites si elle est **vraie ou fausse** en vous fiant au tableau des caractéristiques des employés : le 1 dans une case désigne que l'employé possède la caractéristique, tandis qu'une case vide signifie le contraire.

Employés / caractéristiques	Alain	Baba	Claudia	Denis	Émile	France
ingénieur.e		1	1		1	
sait programmer	1	1	1		1	

(a) $\exists x \left(I(x) \wedge P(x)\right)$

(b) $\forall x \left(I(x) \vee P(x)\right)$

(c) $\forall x \neg\left(I(x) \vee P(x)\right)$

(d) $\exists x \neg\left(I(x) \vee P(x)\right)$

(e) $\exists x \left(P(x) \wedge \neg I(x)\right)$

(f) $\forall x \neg P(x)$

(g) $\forall x \left(I(x) \to P(x)\right)$

(h) $\forall x \left(P(x) \to I(x)\right)$

(i) $\forall x \left(I(x) \leftrightarrow P(x)\right)$

1.23 Soit $M(x)$ l'énoncé « x a étudié au cégep Maisonneuve » et soit $J(x)$ l'énoncé « x connaît le langage Java », où l'univers du discours est l'ensemble des étudiants de l'ÉTS. Exprimez chacune des phrases suivantes en fonctions de $M(x)$, de $J(x)$, de quantificateurs et de connecteurs logiques.

(a) Il y a un étudiant de l'ÉTS qui a étudié au cégep Maisonneuve et qui connaît le langage Java.

(b) Il y a un étudiant de l'ÉTS qui a étudié au cégep Maisonneuve, mais ne connaît pas le langage Java.

(c) Chaque étudiant de l'ÉTS connaît Java.

(d) Aucun étudiant de l'ÉTS a étudié au cégep Maisonneuve ou connaît le langage Java.

(e) Tous les étudiants de l'ÉTS ayant étudié au cégep Maisonneuve connaissent le langage Java.

1.24 Déterminez la valeur de vérité de chacun des énoncés suivants si l'univers du discours est l'ensemble des nombres entiers. **De plus,** pour chacun des énoncés, dites si un exemple ou un contre-exemple suffit à justifier votre réponse ou si un argument général serait requis.

(a) $\exists n\,(n = -2n)$

(b) $\forall n\,(2n \leq 3n)$

(c) $\exists n\,(n^2 < n)$

(d) $\forall n\,(n^2 > 0)$

(e) $\forall n\,(n < n+2)$

1.25 Traduisez chacun des énoncés suivants en expressions logiques à l'aide de quantificateurs, de connecteurs logiques et des fonctions propositionnelles suivantes. $A(x)$ désigne « x est votre ami » ; $V(x)$ désigne « x a une voiture électrique » ; l'univers du discours est l'ensemble des humains.

(a) Au moins un de vos amis n'a pas de voiture électrique.

(b) Tous ceux qui ont une voiture électrique sont vos amis.

(c) Personne n'a de voiture électrique.

(d) Ce n'est pas tout le monde qui a une voiture électrique.

(e) Tous vos amis ont une voiture électrique.

(f) Au moins un de vos amis a une voiture électrique.

(g) Tout le monde est votre ami et a une voiture électrique.

(h) Ce n'est pas tout le monde qui est votre ami ou il y a quelqu'un qui n'a pas de voiture électrique.

1.26 Considérez les prédicats suivants, où l'univers du discours est l'ensemble des dix chiffres : $C = \{0, 1, 2, 3, \ldots, 9\}$.

$$P(x) : \text{« } (x^5 \bmod 10) = x \text{ »}$$
$$T(x) : \text{« } (x^3 \bmod 10) = x \text{ ».}$$

Pour le lecteur qui ne connaît pas encore le modulo, disons simplement que le résultat modulo 10 d'un nombre naturel est le dernier chiffre du nombre (le chiffre des unités) : $173 \bmod 10 = 3$.

Quelles sont les valeurs de vérité des propositions suivantes ? Justifiez.

(a) $P(2) \wedge T(4)$

(b) $\forall x\, T(x)$

(c) $\exists x\, T(x)$

(d) $\exists x \neg T(x)$

(e) $\forall x\, P(x)$

(f) $\exists x \neg P(x)$

1.27 Considérez la proposition suivante : « Quel que soit le nombre naturel x, $(x^5 \bmod 10) = (x \bmod 10)$. »

Complétez la phrase suivante avec un ou plusieurs des choix de réponses. « Pour déterminer les valeurs de vérité de la proposition $\forall x\, P(x), \ldots$ »

1. il suffit de fournir un exemple pour montrer qu'elle est vraie ;
2. il suffit de fournir un contre-exemple pour montrer qu'elle est fausse ;
3. il faut présenter une preuve générale pour montrer qu'elle est vraie, car l'univers du discours est infini ;
4. il faut présenter une preuve générale pour montrer qu'elle est fausse, car l'univers du discours est infini.

1.1.4 Quantificateurs imbriqués

Pour les quantifications à deux variables, on a les équivalences logiques suivantes :

$$\forall x\, \forall y\, P(x,y) \equiv \forall y\, \forall x\, P(x,y) \quad \text{et} \quad \exists x\, \exists y\, P(x,y) \equiv \exists y\, \exists x\, P(x,y).$$

Par contre, il faut faire attention à l'ordre lorsque les quantificateurs universels et existentiels sont imbriqués. En général,

$$\forall x\, \exists y\, P(x,y) \not\equiv \exists y\, \forall x\, P(x,y).$$

Exemple 1.16

Soit E un ensemble[2] d'employés et R l'ensemble des répertoires de leur système informatique.

Désignons par $\boldsymbol{M(x,B)}$ l'énoncé *« l'employé x peut* ***modifier*** *le répertoire B »*
et par $\boldsymbol{L(x,B)}$ l'énoncé *« l'employé x peut* ***lire*** *le répertoire B ».*

Exprimez la proposition suivante à l'aide de quantificateurs et des fonctions propositionnelles M et L.

« Il existe un répertoire pouvant être lu par tous les employés »

Solution :

$$\exists B \in R, \forall x \in E, L(x,B) \qquad \text{ou} \qquad \exists B\, \forall x\, L(x,B)$$

Dans la deuxième expression, on peut déduire quels sont les ensembles de références en observant la position de la variable dans le prédicat. Première variable : employé. Deuxième variable : répertoire.

Plusieurs étudiants éprouvent des difficultés avec les quantificateurs universel et existentiel, particulièrement lorsqu'ils sont imbriqués. Voici deux conseils, suivis de quelques exemples utilisant les mêmes fonctions propositionnelles M et L. Mais attention de ne pas toujours chercher le truc, il faut vraiment réfléchir au sens de la phrase.

1. **Réécrire la phrase sous forme passive avant de traduire du français à l'expression logique.**
2. **Placer les sujets du prédicat au début de la phrase.**

Dans les exemples suivants, on demande encore d'exprimer la proposition à l'aide de quantificateurs et des fonctions propositionnelles M et L. De plus, **le symbole de négation ne doit apparaître que devant les propositions simples** (pas devant un quantificateur ni devant une proposition contenant un ou des connecteurs).

2. La notion d'ensemble ainsi que le symbole d'appartenance $\in$ sont définis à la page 48.

Exemple 1.17

Chaque employé peut modifier au moins un répertoire.

Solution :

Réécrivons d'abord cette phrase sous forme passive avec les sujets du prédicat au début. Ajoutons aussi les variables x et B.

Pour chaque employé x, il existe au moins un répertoire B (qui dépend de l'employé x) *qui peut être modifié par l'employé x.* Passons ensuite à l'expression logique.

$$\forall x \in E, \exists B \in R, M(x,B)$$

Attention! Puisque le répertoire B dépend de l'employé x, B apparaît après x. Chaque employé peut modifier un répertoire, mais il ne s'agit peut-être pas du même répertoire.

Erreur type : inversion des quantificateurs. $\exists B \in R, \forall x \in E, M(x,B)$ Ceci signifie : *il existe au moins un répertoire B tel que, pour chaque employé x, x peut modifier B*. Ou encore : *il existe au moins un répertoire pouvant être modifié par tous les employés.*

Exemple 1.18

Il est faux de dire que chaque employé peut modifier au moins un répertoire.

Solution :

Comme nous l'avons vu avec les lois de De Morgan (1.1), on construit la négation d'une proposition contenant un quantificateur en changeant $\forall$ pour $\exists$ ou réciproquement puis en remplaçant la proposition quantifiée par sa négation.

$$\neg \forall x P(x) \equiv \exists x \neg P(x)$$

Ainsi, en appliquant deux fois cette règle, on obtient

$$\begin{aligned} &\neg(\forall x \in E, \exists B \in R, M(x,B)) \\ &\equiv \exists x \in E \, \neg\big(\exists B \in R, M(x,B)\big) \\ &\equiv \exists x \in E \, \forall B \in R, \neg M(x,B) \end{aligned}$$

Exemple 1.19

Guy peut modifier tous les répertoires que Manon peut lire.

Solution :

Forme passive : *Chaque répertoire qui peut être lu par Manon peut être modifié par Guy.*

Puisqu'il s'agit d'une implication, introduisons les mots *si... alors...*

Pour chaque répertoire B, si ce répertoire B peut être lu par Manon, alors ce répertoire B peut être modifié par Guy.

$$\forall B \in R, L(Manon, B) \rightarrow M(Guy, B)$$

Exemple 1.20

Il est faux de dire que Guy peut modifier tous les répertoires que Manon peut lire.

Solution :

Appliquons une des lois de De Morgan puis utilisons ensuite les tables des équivalences logiques pour trouver une proposition équivalente où le symbole de négation n'apparaît que devant les propositions simples : $\neg(p \to q) \equiv p \wedge \neg q$.

$$\begin{aligned}
&\neg(\forall B \in R, L(Manon, B) \to M(Guy, B)) \\
&\equiv \exists B \in R, \neg(L(Manon, B) \to M(Guy, B)) \\
&\equiv \exists B \in R, (L(Manon, B) \wedge \neg M(Guy, B))
\end{aligned}$$

Exemple 1.21

Les répertoires qui ne peuvent être modifiés par Guy ne peuvent l'être par Manon.

Solution :

Remarquons que l'expression « Les répertoires » a ici le sens de « tous les répertoires » ou « chaque répertoire ».

Passons à la forme passive et introduisons clairement les mots clés *si.. alors...*

Pour chaque répertoire B, si ce répertoire B ne peut pas être modifié par Guy alors ce répertoire B ne peut pas être modifié par Manon.

$$\forall B \in R, \neg M(Guy, B) \to \neg M(Manon, B)$$

Exemple 1.22

Il existe un employé qui peut modifier tous les répertoires.

Solution :

Forme passive où les sujets apparaissent au début de la phrase :

Il existe un employé x tel que chaque répertoire B peut être modifié par cet employé x.

$$\exists x \in E, \forall B \in R, M(x, B)$$

Voici deux erreurs types. La première consiste à copier la structure de la phrase initiale. *Il existe un employé qui peut modifier tous les répertoires.*

$$\exists x \in E, M(x, \forall B)$$

Mais un quantificateur ne peut pas être sujet d'un prédicat : pas de $\forall$ ou $\exists$ à l'intérieur du M.

La deuxième erreur consiste à inverser l'ordre des quantificateurs, ce qui modifie complètement le sens de la phrase.

$$\forall B \in R, \exists x \in E, M(x,B)$$

Cette proposition signifie que pour chaque répertoire B, il existe au moins un employé x (qui dépend du répertoire B) tel que x peut modifier B. Chaque répertoire peut donc être modifié par au moins un employé, mais ce n'est pas nécessairement le même employé qui peut modifier tous les répertoires.

Exercices

1.28 Soit X l'ensemble des 6 employés d'une petite équipe et R l'ensemble des 5 répertoires de leur système informatique. Désignons par $\boldsymbol{M(x,B)}$ l'énoncé *« l'employé x peut **modifier** le répertoire B »* et par $\boldsymbol{L(x,B)}$ l'énoncé *« l'employé x peut **lire** le répertoire B »*.

Écrivez les propositions suivantes **en mots**. De plus, pour chacune des propositions, dites si elle est **vraie ou fausse** en vous fiant au tableau de caractéristiques des employés : un L dans une case désigne que l'employé peut lire le répertoire de cette ligne, un M signifie qu'il peut le modifier, tandis que l'absence d'une lettre L ou M signifie que l'employé ne peut lire ou modifier le répertoire.

(a) $\forall B \in R,\ \exists x \in X, M(x,B)$

(b) $\exists x \in X,\ \forall B \in R, M(x,B)$

(c) $\exists B \in R,\ \forall x \in X, M(x,B)$

(d) $\exists B \in R,\ \forall x \in X, L(x,B)$

(e) $\exists B \in R,\ \forall x \in X, \neg L(x,B)$

(f) $\forall x \in X,\ \exists B \in R, \neg L(x,B)$

(g) $\forall B \in R, \big(L(\text{Manon},B) \to M(\text{Guy},B)\big)$

(h) $\forall B \in R,\ \forall x \in X,\ \big(M(x,B) \to L(x,B)\big)$

(i) $\exists B \in R,\ \exists! x \in X, M(x,B)$

(j) $\exists! B \in R,\ \exists x \in X, M(x,B)$

(k) $\exists! B \in R,\ \exists! x \in X, L(x,B)$

R \ X	Alice	Bartez	Guy	Julien	Manon	Ugo
K		L	L,M			
J	L,M					
P	L	L	L,M	L,M	L,M	L,M
S			L,M		L,M	
Z	L	L	L,M	L	L	L

1.29 Soit $C(x,y)$ l'énoncé « x connaît y », où l'univers du discours de x et de y est l'ensemble de tous les habitants de la Terre. Utilisez des quantificateurs pour exprimer chacun des énoncés suivants :

(a) Ève connaît tout le monde.

(b) Il y a quelqu'un que Julie ne connaît pas.

(c) Il y a quelqu'un que personne ne connaît.

(d) Chaque personne se connaît elle-même.

(e) Il y a quelqu'un qui ne connaît personne d'autre que lui-même.

(f) Tout le monde connaît Karim.

(g) Tout le monde connaît quelqu'un.

(h) Il y a quelqu'un que tout le monde connaît.

(i) Personne ne connaît tout le monde.

1.30 Soit $F(u,p)$ l'énoncé « l'usine u fabrique le produit p », où l'univers du discours de u est l'ensemble U des 5 usines d'une compagnie et l'univers du discours de p est l'ensemble P des 7 produits fabriqués par cette même compagnie. De façon pratique, les 7 produits sont identifiés par les lettres a,b,c,d,e,f,g et une usine est identifiée par la ville où elle se trouve. Écrivez les propositions suivantes **en mots**. De plus, pour chacune des propositions, dites si elle est **vraie ou fausse** en vous fiant au tableau de production de la compagnie : le 1 dans une case désigne que l'usine fabrique le produit, tandis qu'une case vide signifie le contraire.

(a) $\exists u \in U,\ F(u,c)$

(b) $\forall p \in P,\ \big(\neg F(\text{Montréal},p) \vee \neg F(\text{Chicoutimi},p)\big)$

(c) $\forall p \in P,\ \big(F(\text{Montréal},p) \rightarrow \neg F(\text{Gatineau},p)\big)$

(d) $\exists p \in P,\ \big(\neg F(\text{Montréal},p) \wedge \neg F(\text{Québec},p)\big)$

(e) $\exists p \in P,\ \forall u \in U,\ F(u,p)$

(f) $\forall u \in U,\ \exists p \in P,\ F(u,p)$

(g) $\forall u \in U,\ \forall p \in P,\ F(u,p)$

(h) $\forall u \in U,\ \big(F(u,b) \rightarrow \neg F(u,d) \wedge \neg F(u,e)\big)$

(i) $\forall u \in U,\ \big(F(u,a) \rightarrow F(u,f) \vee F(u,g)\big)$

(j) $\forall p \in P,\ \exists u \in U,\ F(u,p)$

(k) $\forall u \in U,\ \big(\neg F(u,c) \vee \neg F(u,d)\big)$

(l) $\forall u \in U,\ \big(F(u,a) \leftrightarrow F(u,b)\big)$

(m) $\exists u_1 \in U,\ \exists u_2 \in U,\ F(u_1,a) \wedge F(u_2,a) \wedge u_1 \neq u_2$

(n) $\exists p \in P,\ \exists! u \in U,\ F(u,p)$

P \ U	Amos	Chicoutimi	Gatineau	Montréal	Québec
a		1		1	1
b		1		1	
c	1	1			
d	1		1		
e	1				1
f	1	1		1	
g			1		

1.31 ★ Considérons le jeu de Sudoku de 9 lignes et 9 colonnes (voir figure 1.1). Désignons $C(i, j, k)$ l'énoncé « la case située dans la ligne i et la colonne j contient le nombre k. L'univers du discours pour les variables i, j et k est l'ensemble des nombres entiers compris entre 1 et 9 inclusivement.

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

Figure 1.1 Exemple de grille du jeu de Sudoku.

Traduisez clairement les règles suivantes à l'aide des quantificateurs.

(a) Il y a au moins un chiffre par case.

(b) Il y a au plus un chiffre par case.

(c) Les chiffres 1 à 9 apparaissent dans chaque colonne.

(d) Les chiffres 1 à 9 apparaissent dans chaque ligne.

(e) Chaque chiffre apparaît une seule fois sur une ligne.

(f) Chaque chiffre apparaît une seule fois sur une colonne.

(g) ★ ★ Les chiffres 1 à 9 apparaissent dans chaque bloc. (La grille est constituée de 9 blocs 3 par 3, voir figure 1.1.)

(h) ★ ★ Chaque chiffre apparaît une seule fois dans chaque bloc.

1.32 Soit $P(x, y)$ le prédicat désignant « $x < y$ ». L'univers du discours pour les variables x et y est l'ensemble des nombres naturels. Pour chacune des propositions suivantes, déterminez sa valeur de vérité (**vrai** ou **faux**). Justifiez en donnant une démonstration, un exemple ou un contre-exemple selon le cas.

(a) $\exists x\, P(x, 4)$

(b) $\exists x\, P(x, 0)$

(c) $\forall x\, \exists y\, P(x, y)$

(d) $\neg\, \forall x\, \forall y\, P(x, y)$

(e) $\neg\, \exists x\, \forall y\, P(x, y)$

(f) $\exists x\, \forall y\, \big((x \neq y) \to P(x, y)\big)$

(g) $\forall x\, \forall y\, \big(P(x, y) \vee P(y, x)\big)$

1.33 Trouvez un contre-exemple, si possible, pour chacune des quantifications suivantes, où l'univers du discours des variables est l'ensemble des nombres entiers.

(a) $\forall x\, \forall y\, (x^2 = y^2 \to x = y)$

(b) $\forall x\, \forall y\, (xy \geq x)$

(c) $\forall x\, \exists y\, (xy = 1)$

1.2 Raisonnements

Un **raisonnement** est un énoncé de la forme $(p_1 \wedge p_2 \wedge \cdots \wedge p_n) \longrightarrow q$, où $p_1, p_2, \ldots p_n$ sont les prémisses (ou hypothèses) et q est la conclusion. On utilise la notation

$$\begin{array}{l} p_1 \\ p_2 \\ \vdots \\ p_n \\ \hline q \end{array}$$

1.2.1 Règles d'inférence

La table 5 de la page 35 résume les principales **règles d'inférence**. Ces formes de raisonnement sont valides : elles garantissent la véracité de la conclusion quand toutes les prémisses (ou hypothèses) sont vraies. Plus formellement, un raisonnement est **valide** si $(p_1 \wedge p_2 \wedge \cdots \wedge p_n) \longrightarrow q$ est une tautologie.

Les hypothèses p_i ne sont pas toujours des propositions simples. Par exemple, voici la règle d'inférence appelée *syllogisme hypothétique*.

$$\begin{array}{ll} p_1 : & p \rightarrow s \\ p_2 : & s \rightarrow r \\ \hline q : & p \rightarrow r \end{array}$$

Les tables suivantes présentent des raisonnements valides auxquels nous ferons référence par la suite.

TABLE 5 Règles d'inférence

p $p \to q$ $\overline{q}$	Modus ponens
$\neg q$ $p \to q$ $\overline{\neg p}$	Modus tollens
$p \to q$ $q \to r$ $\overline{p \to r}$	Syllogisme hypothétique
$p \vee q$ $\neg p$ $\overline{q}$	Syllogisme disjonctif
p $\overline{p \vee q}$	Addition
$p \wedge q$ $\overline{p}$	Simplification
p q $\overline{p \wedge q}$	Conjonction
$p \vee q$ $\neg p \vee r$ $\overline{q \vee r}$	Résolution

TABLE 6 Règles d'inférence (énoncés quantifiés)

$\forall x\, P(x)$ $\overline{P(c)}$	Instanciation universelle
$P(c)$ pour c quelconque $\overline{\forall x\, P(x)}$	Généralisation universelle
$\exists x\, P(x)$ $\overline{P(c)$ pour un certain $c}$	Instanciation existentielle
$P(c)$ pour un certain c $\overline{\exists x\, P(x)}$	Généralisation existentielle

Exemple 1.23 (à compléter en classe)
Analysez chacun des raisonnements suivants pour voir s'il est valide ou non. Indiquez sa structure à l'aide de connecteurs logiques. Si le raisonnement est valide, tentez de trouver sa structure dans les tables 5 et 6.

(a) *S'il pleut, alors les trottoirs sont mouillés.*
Les trottoirs ne sont pas mouillés.
Donc, il ne pleut pas.

Solution :
Le raisonnement est de la forme suivante, appelée *modus tollens*.

$$\begin{array}{l} p \to m \\ \neg m \\ \hline \neg p \end{array}$$

Le *modus tollens* est valide. En effet, la table de vérité suivante montre que la proposition $\big((p \to m) \wedge \neg m\big) \to \neg p$ est une tautologie.

p	m	$\neg m$	$p \to m$	$(p \to m) \wedge \neg m$	$\neg p$	$\big((p \to m) \wedge \neg m\big) \to \neg p$
V	**V**	**F**	**V**	**F**	**F**	**V**
V	**F**	**V**	**F**	**F**	**F**	**V**
F	**V**	**F**	**V**	**F**	**V**	**V**
F	**F**	**V**	**V**	**V**	**V**	**V**

(b) *S'il pleut, alors les trottoirs sont mouillés.*
Les trottoirs sont mouillés.
Donc, il pleut.

Solution :
Ce raisonnement (invalide) est de la forme

$$\begin{array}{l} p \to m \\ m \\ \hline p \end{array}$$

Invalide

où p désigne « il pleut »
et m désigne « les trottoirs sont mouillés ».
Ce raisonnement est invalide. En effet, les prémisses $(p \to m)$ et (m) peuvent être toutes les deux vraies sans que la conclusion p ne soit vraie, comme le montre la table de vérité suivante.

p	m	$p \to m$	$(p \to m) \wedge m$	$\big((p \to m) \wedge m\big) \to p$
F	**V**	**V**	**V**	**F**

(c) *Sarah étudie en LOG ou en TI.*
Sarah n'étudie pas en LOG.
Donc Sarah étudie en TI.

(d) *Si je suis à Montréal, alors je suis au Québec.*
Je ne suis pas à Montréal.
Donc je ne suis pas au Québec.

(e) *Si je suis à Montréal, alors je suis au Québec.*
Je ne suis pas au Québec.
Donc je ne suis pas à Montréal.

(f) *Si je suis à Montréal, alors je suis au Québec.*
Je suis à Montréal.
Donc je suis au Québec.

(g) *Les grandes villes ont des gratte-ciel.*
Montréal est une grande ville.
Donc Montréal a des gratte-ciel.

(h) *Tous les chats sont mortels.*
Socrate est mortel.
Donc Socrate est un chat.

(i) *Les martiens sont verts.*
Xip est un martien.
Donc Xip est vert.

(j) *Alex fume.*
Certains fumeurs développeront un cancer.
Donc Alex développera un cancer.

1.2.2 Cohérence d'un ensemble de spécifications

Dans la conception d'un logiciel, l'ingénieur.e logiciel doit décrire un ensemble de spécifications (tâches qui devront être exécutées par l'ordinateur). Traduire ces spécifications (phrases du langage courant) en expressions logiques permet d'éviter toute ambiguïté dans le développement du logiciel.

Cet ensemble de spécifications doit être **cohérent**, c'est-à-dire qu'il ne doit mener à aucune contradiction. Pour vérifier qu'un ensemble de spécifications est cohérent, on traduit premièrement les spécifications en expressions logiques composées de propositions simples (aussi appelées variables) et on s'assure ensuite qu'elles puissent être toutes vraies simultanément, pour certaines valeurs de vérité des variables. On dit alors qu'il existe une **affectation** des variables qui satisfait l'ensemble de spécifications.

Plusieurs approches permettent de résoudre ce type de problèmes, par exemple :

1. Vérifier si la table de vérité des spécifications contient une unique ligne où les propositions sont vraies en même temps.
2. Utiliser les règles d'inférences ainsi que les équivalences logiques pour déduire les valeurs des variables puis vérifier si elles satisfont bien chacune des spécifications.

Dans cette deuxième approche, le niveau de détails de la justification fournie peut être plus ou moins grand. Soit on explique son raisonnement en quelques mots, soit on le formalise en citant chacune des règles logiques utilisées.

La vérification de la cohérence d'un ensemble de spécifications devient rapidement très longue quand le nombre de variables augmente : c'est un problème qui suscite encore beaucoup de recherche. Dans le cours, nous travaillerons avec de petits ensembles de spécifications composées d'un petit nombre de variables. Plusieurs énigmes et jeux mathématiques peuvent aussi être résolus grâce à une analyse de satisfaisabilité.

Exemple 1.24

Un étudiant entre au Pub *Le 100 génies* et fait cinq déclarations.

1. Les jours où je ne fais pas de maths et je dors, je suis content.
2. Les jours où je fais des maths, je suis content et je dors.
3. Les jours où je ne bois pas de café, ou bien je suis content, ou bien je dors, ou les deux.
4. Les jours où je bois du café, je fais des maths ou je ne suis pas content, ou les deux.
5. Je ne suis pas content aujourd'hui.

En supposant que les cinq déclarations sont vraies, avez-vous suffisamment d'informations pour déterminer si cet étudiant a fait des maths aujourd'hui? S'il a bu du café? S'il a dormi? Si oui, répondez aux questions en expliquant votre raisonnement. Sinon, dites pourquoi.

Solution :

Identifions les propositions élémentaires suivantes (aussi appelées variables).

m : « l'étudiant a fait des maths »;

b : « l'étudiant a bu du café »;

c : « l'étudiant est content »;

d : « l'étudiant a dormi ».

Nous savons que les cinq propositions suivantes sont vraies.

$p1$: $(\neg m \wedge d) \rightarrow c$

$p2$: $m \rightarrow (c \wedge d)$

$p3$: $\neg b \rightarrow (c \vee d)$

$p4$: $b \rightarrow (m \vee \neg c)$

$p5$: $\neg c$

Voyons si nous pouvons trouver une unique affectation des propositions m, b, d et c qui satisfont les propositions 1 à 5, c'est-à-dire qui font en sorte que $p1$ à $p5$ soient vraies.

D'abord, pour que $p5$ soit vraie, il est nécessaire que la variable c soit fausse, ce qui entraîne alors que $(c \wedge d)$ est fausse. Ainsi, l'affirmation $p2$ nous permet de conclure que m doit être fausse : l'étudiant n'a pas fait de maths.

Ensuite, pour que $p1$ soit vraie tout en ayant c fausse, il faut que $(\neg m \wedge d)$ soit fausse, et donc que d soit fausse (car $\neg m$ est vraie) : l'étudiant n'a pas dormi.

Puis, pour que $p3$ soit vraie sachant que $(c \vee d)$ est fausse, il est nécessaire que $\neg b$ soit fausse : l'étudiant a bu du café.

Finalement, nous vérifions que l'affectation trouvée, c'est-à-dire $b = \mathbf{V}$, $c = \mathbf{F}$, $d = \mathbf{F}$ et $m = \mathbf{F}$, satisfait aussi $p4$: comme b est vraie, il faut que $(\neg c \vee m)$ soit vraie, ce qui est le cas, car $\neg c$ est vraie.

Réponse. Oui, nous pouvons déduire que l'étudiant a bu du café, n'a pas dormi et n'a pas fait de maths.

Voici une solution plus détaillée citant chacune des règles d'inférences et des formules d'équivalence logique utilisée, ainsi que chacune des propositions satisfaites (énoncés de l'étudiant).

	proposition	justification	affectation	proposition satisfaite
1.	$(\neg m \wedge d) \rightarrow c$			
2.	$m \rightarrow (c \wedge d)$			
3.	$\neg b \rightarrow (c \vee d)$			
4.	$b \rightarrow (\neg c \vee m)$			
5.	$\neg c$		$c = \mathbf{F}$	$p5$
6.	$\neg c \vee \neg d$	par *addition* à la proposition de la ligne 5.		
7.	$\neg(c \wedge d)$	par *De Morgan* de 6.		
8.	$\neg m$	par *modus tollens* de 2 et 7.	$m = \mathbf{F}$	$p2$
9.	$\neg(\neg m \wedge d)$	par *modus tollens* de 1 et 5.		
10.	$\neg\neg m \vee \neg d$	par *De Morgan* de 9.		
11.	$m \vee \neg d$	par *double négation* de 10.		
12.	$\neg d$	par *syllogisme disjonctif* de 8 et 11.	$d = \mathbf{F}$	$p1$
13.	$\neg c \wedge \neg d$	par *conjonction* de 5 et 12.		
14.	$\neg(c \vee d)$	par *De Morgan* de 13.		
15.	$\neg\neg b$	par *modus tollens* de 14 et 3.		
16.	b	par *double négation* de 15.	$b = \mathbf{V}$	$p3$
17.	$b \wedge \neg c \wedge \neg d \wedge \neg m$	par *conjonction* de 5, 8, 12 et 16.		

Finalement, il faut vérifier que l'affectation trouvée, c'est-à-dire $b = \mathbf{V}$, $c = \mathbf{F}$, $d = \mathbf{F}$ et $m = \mathbf{F}$, satisfait le 4^{e} énoncé de l'étudiant (laissé au lecteur).

Exemple 1.25

Si l'étudiant de l'exemple précédent entre au pub et refait les quatre premières déclarations, mais affirme cette fois « Je n'ai pas bu de café aujourd'hui, mais je suis content », avez-vous suffisamment d'informations pour déterminer si cet étudiant a fait des maths aujourd'hui et s'il a dormi? Si oui, répondez aux questions en expliquant votre raisonnement. Sinon, dites pourquoi.

Solution :

Gardons les mêmes propositions simples b, c, d et m (aussi appelées variables), ainsi que les quatre premières propositions (énoncées par l'étudiant) et posons :

$p5$: $\neg b \wedge c$

Voyons si nous pouvons trouver une unique affectation des variables qui satisfont les propositions 1 à 5.

	proposition	justification	affectation	proposition satisfaite
1.	$(\neg m \wedge d) \rightarrow c$			
2.	$m \rightarrow (c \wedge d)$			
3.	$\neg b \rightarrow (c \vee d)$			
4.	$b \rightarrow (\neg c \vee m)$			
5.	$\neg b \wedge c$			
6.	c	par *simplification* de 5.	$c = \mathbf{V}$	$p1$
7.	$\neg b$	par *simplification* de 5.	$b = \mathbf{F}$	$p4, p5$
8.	$c \vee d$	par *addition* à 6.		$p3$
9.	$\neg m \vee (c \wedge d)$	équivalent à 2 par table 2.1		
10.	$\neg m \vee d$	$c \wedge d = d$ par *identité*, car $c = \mathbf{V}$		

Il est impossible d'aller plus loin et de déterminer si l'étudiant a fait des maths aujourd'hui et s'il a dormi, car il y a plusieurs possibilités pour les valeurs de vérité de m et d qui satisfont les cinq énoncés :

$$m = \mathbf{F} \text{ et } d = \mathbf{V} \quad \text{ou} \quad m = \mathbf{F} \text{ et } d = \mathbf{F} \quad \text{ou} \quad m = \mathbf{V} \text{ et } d = \mathbf{V}.$$

Cependant, nous pouvons conclure que si l'étudiant a fait des maths, alors il a dormi ! En effet :

$$\neg m \vee d \equiv m \rightarrow d.$$

Exercices

1.34 Analysez chacun des raisonnements suivants pour voir s'il est valide ou non. Indiquez sa structure à l'aide de connecteurs logiques. Si le raisonnement est valide, tentez de trouver sa structure dans la table de la page 35.

(a) S'il pleut, alors les trottoirs sont mouillés. Il ne pleut pas. Donc les trottoirs ne sont pas mouillés.

(b) S'il pleut, alors les trottoirs sont mouillés. Il pleut. Donc les trottoirs sont mouillés.

(c) Si tu fais tous les exercices du livre de référence, tu réussiras le cours. Tu as réussi le cours. Donc, tu as fait tous les exercices du livre.

(d) Ou bien la science peut expliquer ce phénomène, ou bien c'est un miracle. La science ne peut expliquer ce phénomène. C'est donc un miracle.

(e) Ce sandwich ne contient pas de viande ou contient des champignons. Ce sandwich contient de la viande ou de la mayonnaise. Donc ce sandwich contient des champignons ou de la mayonnaise.

1.35 En utilisant les quantificateurs $\exists$ et $\forall$, traduisez les énoncés suivants.

(a) Il existe un et un seul élément de l'univers du discours pour lequel le prédicat P est vrai.
Remarque : on utilise parfois la forme $\exists !x\, P(x)$ *pour désigner cet énoncé, mais on demande ici de l'exprimer sans le symbole* !

(b) Il existe au moins deux éléments de l'univers du discours pour lesquels P est vrai.

1.36 Annie se rend à l'ÉTS et elle se rend compte qu'elle a oublié son téléphone chez elle. Elle se fait les réflexions suivantes :

1. Si mon téléphone est sur ma table de chevet, alors je l'ai vu en me levant
2. Si j'ai été réveillée par la sonnerie de mon téléphone, alors il se trouve sur ma table de chevet.
3. Si j'ai parlé au téléphone dans la cuisine, alors mon téléphone est sur le comptoir de la cuisine.
4. J'ai été réveillée par la sonnerie de mon téléphone ou j'ai parlé au téléphone dans la cuisine.
5. Je n'ai pas vu mon téléphone en me levant.

Où est son téléphone ?

1.37 Montrer que les quatre hypothèses ci-dessous mènent à la conclusion : $\neg a \wedge b$.

1. $a \vee \neg b \rightarrow \neg c \wedge d$
2. $e \rightarrow c \vee \neg d$
3. $\neg f$
4. $e \vee f$

1.38 Traduisez les informations suivantes concernant un vol. Utilisez $P(x)$ pour « x était présent » et $C(x)$ pour « x est coupable ».

1. Il y a un et un seul coupable.
2. Le ou les coupables étaient forcément présents.
3. Au moins une des personnes qui étaient présentes n'est pas coupable.
4. Si Alain et Diane étaient présents, alors Claude est coupable.
5. Si Alain ou Claude étaient présents, Benoît est coupable.
6. Diane n'était pas présente.
7. Si Diane n'est pas coupable, alors Alain non plus.
8. Alain était présent si et seulement si Claude ne l'était pas.

1.39 À l'aide des informations de l'exercice précédent et sachant que seules quatre personnes pouvaient être présentes (Alain, Benoît, Claude et Diane), est-il possible de déterminer qui est le coupable ? Si oui, déterminez qui est coupable et justifiez votre raisonnement.

1.40 Montrez que les hypothèses :

1. Un étudiant inscrit au cours de Mathématiques discrètes n'a pas fait les exercices suggérés.
2. Tous les étudiants inscrits au cours de Mathématiques discrètes ont réussi l'intra.

mènent à la conclusion : « Il y a un étudiant qui a réussi l'intra, mais n'a pas fait les exercices suggérés. »

1.41 Analysez chacun des raisonnements suivants pour voir s'il est valide ou non.

(a) Tous les hommes sont mortels.
Socrate est un homme.
Donc Socrate est mortel.

(b) Tous les hommes sont bleus.
Socrate est un homme.
Donc Socrate est bleu.

(c) Quelques hommes sont bleus.
Socrate est un homme.
Donc Socrate est bleu.

(d) Certains hommes sont mortels.
Socrate est un homme.
Donc Socrate est mortel.

1.42 Parmi les trois raisonnements suivants, deux sont valides et un est invalide.

A	B	C
1. $a \to (b \wedge c)$ 2. $\neg c$ $\therefore \neg a$	1. $a \to b$ 2. $\neg a$ $\therefore \neg b$	1. $(a \vee b) \wedge c$ 2. $d \to \neg c$ 3. $d \vee \neg a$ $\therefore b$

(a) Déterminez quel raisonnement est invalide et expliquez pourquoi.

(b) Démontrez que les deux autres raisonnements sont valides en utilisant les règles d'inférence et les tables d'équivalences.

1.43 L'ensemble de spécifications suivant est-il cohérent ?

1. La communication est fluide si le logiciel Communik est installé.
2. Pour que le logiciel Communik soit installé, il est nécessaire que la version 10 ou plus du système d'exploitation soit installée.
3. C'est la version 9 du système d'exploitation qui est installée.
4. La communication est fluide .

1.44 L'ensemble de spécifications de l'exercice précédent est-il cohérent si on modifie la première spécification ainsi ?

1. La communication est fluide seulement si le logiciel Communik est installé.

1.45 L'ensemble de spécifications suivant est-il cohérent ?

1. La communication est fluide seulement si le logiciel Communik est installé.
2. Pour que le logiciel Communik soit installé, il est nécessaire que la version 10 ou plus du système d'exploitation soit installée.
3. la version 10 ou plus du système d'exploitation assure une communication fluide.
4. C'est la version 9 du système d'exploitation qui est installée.

1.46 Si a, b, c et d désignent des propositions, l'ensemble de spécifications suivant est-il cohérent ?

1. $a \rightarrow (b \wedge \neg c)$
2. $b \rightarrow \neg d$
3. $c \rightarrow b$
4. $d \leftrightarrow \neg a$

1.47 Trouvez toutes les assignations des variables a, b, c et d qui satisfont l'ensemble de spécifications de l'exercice précédent.

1.48 Si a, b, c et d désignent des propositions, l'ensemble de spécifications suivant est-il cohérent ?

1. $b \rightarrow c$
2. $\neg a \rightarrow d$
3. $\neg c \vee d$
4. $a \leftrightarrow b$
5. $\neg d$

1.49 Déterminer si le système de spécifications ci-dessous est cohérent. Si oui, donner toutes les valeur de vérité des propositions p, q, r, s et t qui satisfont chacune des spécifications.

1. $r \wedge (\neg p \rightarrow q)$
2. $\neg q \vee s$
3. $t \rightarrow q \vee \neg s$
4. $\neg(\neg q \wedge s) \rightarrow t$
5. $s \rightarrow \neg r$

1.50 Quatre amis sont identifiés comme les suspects d'un crime. Voici leurs déclarations :

1. Anouk dit : « Xavier est coupable. »
2. Geneviève dit : « Je ne suis pas coupable. »
3. Xavier dit : « Stéphane est coupable. »
4. Stéphane dit : « Xavier a menti quand il a dit que j'étais coupable. »

Sachant qu'il y a exactement un suspect qui dit la vérité et qu'il n'y a qu'un seul coupable, déterminez qui est coupable.

1.51 Trois amis ont été identifiés comme les suspects d'un crime et sont interrogés. Le coupable ment et les autres disent la vérité. Voici leurs déclarations :

1. Anouk a dit : « Xavier est coupable. »
2. Geneviève a dit : « Xavier ment. »
3. Xavier a dit : « Geneviève est coupable. »

Déterminez qui est coupable. *Conseil : complétez un tableau des états possibles (sincère ou menteur) et analysez la cohérence des énoncés en fonction des états.*

1.52 Anouk ment systématiquement du dimanche au mardi et dit la vérité les autres jours de la semaine, tandis que son amie Geneviève ment toujours du mercredi au vendredi et est sincère du samedi au mardi. Xavier rencontre ses amies. Anouk lui dit « hier, j'ai menti » et Geneviève ajoute « moi aussi ». Quel jour de la semaine a lieu cette discussion ? *Conseil : complétez un tableau des états possibles selon les journées (sincère ou menteur) et analysez la cohérence des énoncés en fonction des états.*

1.53 Sur une île vivent deux catégories d'habitants : les sincères qui disent toujours la vérité, et les menteurs qui mentent toujours. Vous rencontrez deux habitants, A et B. Déterminez, si possible, de quelles catégories d'habitants sont A et B si ces deux personnes s'adressent à vous de la manière suivante.

(a) A dit : « Au moins un de nous deux est un menteur. » et B ne dit rien.

(b) A dit : « Nous sommes les deux sincères. » et B dit : « A est un menteur. »

(c) A dit : « Je suis un menteur ou B est sincère. » et B ne dit rien.

(d) A et B disent chacun : « Je suis sincère. »

(e) A dit : « Nous sommes les deux des menteurs. » et B ne dit rien.

1.54 ★ Deux gardiens, l'un toujours sincère et l'autre toujours menteur, sont devant deux portes. L'une mène au cours de Mathématiques discrètes, l'autre en enfer. Vous avez le droit de poser une seule question à un seul gardien, sans savoir lequel est sincère. Formulez une question qui vous permettra de déterminer quelle porte mène au cours de Mathématiques discrètes.

1.2.3 Types de preuve

Définition 1.7 : Théorème, démonstration, axiome

Un **théorème** est un énoncé que l'on peut démontrer.

Une **démonstration** (on dit aussi une **preuve**) consiste à déduire que l'énoncé est vrai en utilisant un raisonnement logique (règles d'inférence) reposant sur des **axiomes** ou **postulats** (énoncés considérés vrais sans démonstration) ou sur des théorèmes déjà démontrés.

Définition 1.8 : Conjecture

Une **conjecture** est une proposition que l'intuition ou l'observation nous porte à croire vraie, mais qui n'a pas encore été formellement démontrée (ni réfutée). Si elle est démontrée, la conjecture devient un théorème.

Le travail des mathématiciens consiste à formuler des conjectures puis à démontrer qu'elles sont vraies (ou fausses). L'équivalent en informatique consiste à inventer un algorithme, ou un programme, puis à démontrer qu'il produit toujours le résultat attendu.

Au cours de la session, vous serez amenés à lire différentes preuves et à en produire vous-mêmes. Démontrer un résultat n'est pas facile; il n'y a pas de recette magique. Par contre, connaître les principaux types de preuves est un atout précieux.

Types de preuves

- **Preuve directe** de $p \to q$
  On suppose que p est vrai et l'on démontre (en se basant sur des définitions, des axiomes, des théorèmes déjà démontrés et des règles d'inférence), qu'il s'en suit forcément que q est vrai aussi.

- **Preuve directe** de $\forall x \in U\ P(x) \to Q(x)$
  On suppose que x est un élément arbitraire de U tel que $P(x)$ est vrai. On démontre qu'il s'en suit forcément que $Q(x)$ est vrai aussi.

- **Preuve indirecte** de $p \to q$, aussi appelée **preuve par contraposée**
  On suppose que q est faux et l'on démontre qu'il s'en suit forcément que p est faux.

$$\frac{\neg q \to \neg p}{p \to q}$$

- **Preuve par contradiction** de p, aussi appelée **preuve par l'absurde**
  On suppose que p est faux et l'on montre que cela entraîne une contradiction. Ainsi, p doit être vrai.

$$\frac{\neg p \to F}{p}$$

- **Preuve d'équivalence** de $p \leftrightarrow q$
  On montre que $p \to q$ et $q \to p$ sont vrais. Ainsi, $p \leftrightarrow q$ est vrai.

- **Preuve par cas** de $p \to q$
  On démontre que p entraîne un nombre fini de possibilités. On étudie chacune séparément pour vérifier qu'elle entraîne q. On conclut donc que p entraîne q.

$$\frac{\begin{array}{l} p \to (p_1 \vee p_2 \vee \ldots \vee p_n) \\ p_1 \to q \\ p_2 \to q \\ \vdots \\ p_n \to q \end{array}}{p \to q}$$

- **Preuve triviale** de $p \to q$
  On montre que q est vrai. Ainsi, l'implication $p \to q$ est vraie (peu importe la valeur de p).

$$\frac{q}{p \to q}$$

- **Preuve exhaustive** de $\forall x \in U\ P(x)$
  Si l'ensemble U possède un nombre fini d'éléments, disons $U = \{x_1, x_2, \ldots, x_n\}$, et que l'on démontre que pour chacun d'eux la propriété P est vraie, alors on peut conclure que la propriété P est vraie pour tous les éléments de U.

$$\frac{\begin{array}{l} U = \{x_1, x_2, \ldots, x_n\} \\ P(x_1) \wedge P(x_2) \wedge \ldots \wedge P(x_n) \end{array}}{\forall x \in U\ P(x)}$$

- **Preuve constructive** de $\exists x \in U\ P(x)$
  Il suffit de trouver un élément a de l'ensemble U tel que $P(a)$ est vrai.

$$\frac{\begin{array}{l} a \in U \\ P(a) \end{array}}{\exists x \in U\ P(x)}$$

- **Preuve par récurrence** de $\forall n \in N\ P(n)$
  Pour démontrer que la propriété P est vraie pour tous les nombres naturels, on démontre qu'elle est vraie pour le nombre 0 (cas de base), et que si elle est vraie pour un nombre n, alors elle est aussi vraie pour le nombre suivant (étape de récurrence).

$$\frac{\begin{array}{l} P(0) \\ \forall n \in N \left(P(n) \to P(n+1)\right) \end{array}}{\forall n \in N\ P(n)}$$

De nombreux exemples de preuves de chaque type seront donnés en classe, en commençant par des preuves assez courtes et « simples » semblables à celle de l'exercice de la page 58. Les preuves par récurrence seront abordées dans la deuxième moitié de la session.

1.3 Théorie des ensembles

1.3.1 Notions de base sur les ensembles

Définition 1.9 : Ensemble, élément

Un **ensemble** est une collection non ordonnée d'objets. Les objets sont appelés **éléments** de l'ensemble et on dit qu'ils appartiennent à l'ensemble.

Notation : $x \in F$ signifie que x est un élément de l'ensemble F. On dit aussi que x appartient à l'ensemble F.

Définition 1.10 : Ensemble fini ou infini, cardinalité

Soit A un ensemble composé de n éléments distincts. On dit que A est un **ensemble fini** de **cardinalité** n et on note $|A| = n$. Un ensemble est dit **infini** s'il n'est pas fini.

Exemple 1.26

Soit

$$F = \{2, \pi, 7\}.$$

Utilisez les symboles introduits pour traduire les énoncés suivants : l'ensemble F contient 3 éléments, π appartient à F, 5 n'appartient pas à F.

Solution :

$$|F| = 3, \qquad \pi \in F, \qquad 5 \notin F.$$

On peut décrire un ensemble **en extension** (on énumère ses éléments que l'on place entre accolades)

$$A = \{5, 7, 9, 11\} \qquad B = \{1, 8, 27, 64\}$$

ou **en compréhension**, comme ceci :

$$A = \left\{x \in \mathbb{N} \mid (x \text{ est impair }) \wedge (5 \leq x \leq 11)\right\} \qquad B = \left\{x \in \mathbb{N} \mid (x \leq 64) \wedge \left(\exists y \in \mathbb{N}, y^3 = x\right)\right\}$$

1.3.2 Ensembles de nombres $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$

$\emptyset = \{\}$	Ensemble vide (ne contient aucun élément $\lvert\emptyset\rvert = 0$)
$\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$	Ensemble des nombres entiers
$\mathbb{Z}^* = \{\ldots, -2, -1, 1, 2, \ldots\}$	Ensemble des entiers non nuls
$\mathbb{N} = \{0, 1, 2, 3, \ldots\}$	Ensemble des nombres naturels
$\mathbb{N}^* = \{1, 2, 3, \ldots\}$	Ensemble des nombres naturels strictement positifs
$\mathbb{Q} = \left\{\frac{p}{q} \mid p \in \mathbb{Z}, q \in \mathbb{Z} \text{ et } q \neq 0\right\}$	Ensemble des nombres rationnels
$\mathbb{R}$	Ensemble des nombres réels
$\mathbb{R}^+ = \left\{x \in \mathbb{R} \mid x \geq 0\right\}$	Ensemble des nombres réels positifs
$\mathbb{C} = \left\{a + bi \mid a \in \mathbb{R} \text{ et } b \in \mathbb{R}\right\}$ avec $i = \sqrt{-1}$, donc $i^2 = -1$	Ensemble des nombres complexes

TABLE 6 Propriétés des ensembles de nombres $\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R}$

Si A est un ensemble de nombre parmi $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ et $a, b, c \in A$, alors	
$a + b \in A$	Clos pour l'addition
$a \cdot b \in A$	Clos pour la multiplication
$a + b = b + a$	Commutativité de l'addition
$a \cdot b = b \cdot a$	Commutativité de la multiplication
$a(b + c) = ab + ac$	Distributivité de la multiplication sur l'addition
$a \cdot b = 0 \leftrightarrow (a = 0 \vee b = 0)$	Équation produit-nul
Si $a, b \in \mathbb{N}$, alors	
$a + b = 0 \rightarrow (a = 0 \wedge b = 0)$	Absence d'un inverse additif
Si $a, b \in \mathbb{Z}$, alors	
$a \cdot b = 1 \rightarrow ((a = 1 \wedge b = 1) \vee (a = -1 \wedge b = -1))$	Absence d'un inverse multiplicatif
Si A est un ensemble de nombre parmi $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$ et $a \in A$, alors	
$a - b = a + (-1) \cdot b$	Notation
$a - a = 0$	Inverse additif
Si $a, b, c, d \in \mathbb{Z}$, $b \neq 0$ et $d \neq 0$, alors $\frac{a}{b}, \frac{c}{d} \in \mathbb{Q}$ et	
$\frac{a}{b} = \frac{c}{d} \leftrightarrow ad = bc$	Égalité de fractions
$\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}$	Addition de fractions
$\frac{a}{b} \cdot \frac{c}{d} = \frac{ac}{bd}$	Multiplication de fractions

Définition 1.11 : Nombres pairs et impairs

- Un nombre x est **pair** s'il existe $n \in \mathbb{Z}$ tel que $x = 2n$.
- Un nombre x est **impair** s'il existe $n \in \mathbb{Z}$ tel que $x = 2n+1$.

Exemple 1.27

Montrez que le carré d'un nombre impair est impair.

Solution :

On veut montrer : « $\forall x \in \mathbb{Z}, (x \text{ est impair}) \rightarrow (x^2 \text{ est impair})$ »,

1.	x est impair	
2.	$x = 2n+1,\ n \in \mathbb{Z}$	(définition d'*impair*)
3.	$x^2 = (2n+1)^2$	(par 2.)
4.	$x^2 = 4n^2 + 4n + 1$	(distributivité)
5.	$x^2 = 2(2n^2 + 2n) + 1$	(distributivité)
6.	$x^2 = 2m+1$, où $m = 2n^2 + 2n \in \mathbb{Z}$	($2, n \in \mathbb{Z}$ et $\mathbb{Z}$ est clos pour la multiplication)
7.	x^2 est impair	(définition d'*impair*)

Théorème 1.2 : Représentation réduite des fractions

Pour tout élément $x \in \mathbb{Q}$, il existe une unique paire $p \in \mathbb{Z}$, $q \in \mathbb{N}^*$, telle que $x = \frac{p}{q}$ et l'unique diviseur commun à p et q est 1.

Exemple 1.28

Montrez que le nombre $\sqrt{2}$ n'est pas rationnel.

(Aide, $\sqrt{2}$ est défini comme étant un nombre qui lorsque multiplié par lui-même donne 2.)

Solution :

Démontrer que le nombre $\sqrt{2}$ n'est pas rationnel n'est pas si simple que cela ! Nous allons procéder par contradiction, c'est-à-dire que nous allons démontrer l'implication suivante :

$$(\sqrt{2} \in \mathbb{Q}) \rightarrow \mathbf{F}.$$

Ainsi, nous pourrons conclure que

$$(\sqrt{2} \in \mathbb{Q}) \equiv \mathbf{F},$$

ou encore que

$$\sqrt{2} \notin \mathbb{Q}.$$

Par contradiction, on suppose que $\sqrt{2} \in \mathbb{Q}$. Par le théorème 1.2, on a qu'il existe une unique fraction $\frac{p}{q} = \sqrt{2}$ telle que $p \in \mathbb{Z}$, $q \in \mathbb{N}^*$ et l'unique diviseur commun à p et q est 1. De manière plus formelle, cette dernière propriété se formule de la façon suivante : $\forall a \in \mathbb{N}, \forall p' \in \mathbb{Z}, \forall q' \in \mathbb{Z}, (p = ap' \wedge q = aq') \rightarrow a = 1$.

1.	$p \in \mathbb{Z}$	
2.	$q \in \mathbb{N}^*$	
3.	$\left(\frac{p}{q}\right)^2 = 2$	
4.	$\forall a \in \mathbb{N}, \forall p' \in \mathbb{Z}, \forall q' \in \mathbb{Z}, (p = ap' \wedge q = aq') \rightarrow a = 1$, c'est-à-dire que $\text{pgcd}(p,q) = 1$	
5.	$\frac{p^2}{q^2} = 2$	(par 3. et multiplication de fractions)
6.	$p^2 = 2q^2 = 2k$, où $k = q^2 \in \mathbb{Z}$	(par 5. et $\mathbb{N}^* \subset \mathbb{Z}$ est clos pour la multiplication.)
7.	p^2 est pair	(par 6. et définition de *pair*)
8.	p^2 pair $\rightarrow$ p pair	(contraposée de l'exemple 1.27)
9.	p est pair	(modus ponens 7. et 8.)
10.	$p = 2l$, où $l \in \mathbb{Z}$	(9. et définition de *pair*)
11.	$p^2 = (2l)^2 = 2 \cdot 2l^2$	(par 10.)
12.	$2q^2 = p^2 = 2 \cdot 2l^2$	(par 6. et 11.)
13.	$q^2 = 2l^2 = 2m$, où $m \in \mathbb{Z}$	(par 12. et $\mathbb{Z}$ clos par multiplication)
14.	q^2 est pair	(par 14. et définition de *pair*)
15.	q^2 pair $\rightarrow$ q pair	(contraposée de l'exercice 1.27)
16.	q est pair	(modus ponens 14. et 15.)
17.	$q = 2n$, $n \in \mathbb{Z}$	(par 16. et définition de *pair*)
18.	$p = 2l \wedge q = 2n$	(conjonction de 10. et 17.)
19.	$(p = 2l \wedge q = 2n) \rightarrow 2 = 1$	(instanciation exist. de 4. avec $a = 2$, $p' = l$, $q' = n$)
20.	$2 = 1$	(modus ponens 18. et 19.)
21.	**Faux**	(logiquement équivalent à 20.)

Définition 1.12 : Égalité d'ensembles

Deux ensembles sont dits égaux si et seulement s'ils contiennent exactement les mêmes éléments.

$$A = B \longleftrightarrow \forall x\big(x \in A \leftrightarrow x \in B\big)$$

Exemple 1.29

$\{1,3,5\} = \{3,5,1\}$

$\{1,3,5\} \neq \{\{1\},\{3\},\{5\}\}$, puisque l'élément 1 n'est pas égal à l'élément $\{1\}$.

Définition 1.13 : Sous-ensemble

L'ensemble A est **sous-ensemble** de l'ensemble B si et seulement si tous les éléments de A sont aussi des éléments de B :

$$A \subseteq B \longleftrightarrow \forall x\big(x \in A \rightarrow x \in B\big).$$

L'ensemble A est **sous-ensemble strict (ou propre)** de l'ensemble B si et seulement si tous les éléments de A sont aussi des éléments de B et A n'est pas égal à B :

$$A \subset B \longleftrightarrow A \subseteq B \ \wedge A \neq B.$$

Exemple 1.30

$$\{1,2\} \subseteq \{1,2,3,4,5\}$$

$$\{1,2\} \subset \{1,2,3,4,5\}$$

$$\{2k \mid k \in \mathbb{N}\} = \{0,2,4,6,\ldots\} \subset \mathbb{N}$$

Théorème 1.3

Pour tout ensemble A, on a

1. $\emptyset \subseteq A$
2. $A \subseteq A$

Théorème 1.4

$A = B$ si et seulement si $A \subseteq B$ et $B \subseteq A$.

Rappelons qu'un théorème est un énoncé que l'on a *démontré*. Or les démonstrations des théorèmes n'apparaissent toujours pas dans ce texte. Certaines sont présentées en classe, d'autres sont laissées en exercice au lecteur.

1.3.3 Produit cartésien

Définition 1.14 : Produit cartésien

Le **produit cartésien** des ensembles A et B, noté $\boldsymbol{A \times B}$, est l'ensemble de tous les couples (paires ordonnées) dont le premier élément appartient à A et le second, à B :

$$A \times B = \{(a,b) \mid a \in A \text{ et } b \in B\}.$$

On généralise cette définition au produit cartésien de n ensembles :

$$A_1 \times A_2 \times \ldots \times A_n = \{(a_1, a_2, \ldots, a_n) \mid a_1 \in A_1, \ldots, a_n \in A_n\}$$

Exemple 1.31

Décrivez en extension les produits cartésiens $A \times B$ et $B \times A$, où $A = \{0,1,2\}$ et $B = \{a,c\}$.

Solution :

$$A \times B = \{(0,a),(0,c),(1,a),(1,c),(2,a),(2,c)\}$$
$$B \times A = \{(a,0),(c,0),(a,1),(c,1),(a,2),(c,2)\}$$

Définition 1.15 : Relation

Une **relation** entre les ensembles A et B est un sous-ensemble du produit cartésien $A \times B$.

Exemple 1.32

Soit $A = \{0,1,2\}$ et $B = \{a,c\}$. L'ensemble

$$R = \{(0,a),(1,c),(2,a)\} \subseteq A \times B$$

est une relation de A dans B.

Définition 1.16 : Ensemble des parties

L'**ensemble des parties de $\boldsymbol{A}$**, noté $\wp(\boldsymbol{A})$, est l'ensemble de tous les sous-ensembles de A.

$$B \in \wp(A) \longleftrightarrow B \subseteq A$$

Exemple 1.33

Décrivez $\wp(A)$, l'ensemble des parties de A, où $A = \{0,1,2\}$.

Solution :

k	Sous-ensembles de A ayant k éléments	nombre de sous-ensembles
0	$\emptyset$	1
1	$\{0\},\{1\},\{2\}$	3
2	$\{0,1\},\{0,2\},\{1,2\}$	3
3	$\{0,1,2\}$	1

Ainsi, $\wp(A)$ contient 8 éléments, soit $2^{|A|}$.

$$\wp(A) = \big\{\emptyset,\{0\},\{1\},\{2\},\{0,1\},\{0,2\},\{1,2\},\{0,1,2\}\big\}$$

Exemple 1.34

Décrivez l'ensemble des parties de A, où $A = \{0,1,2,3\}$.

Solution :

k	Sous-ensembles de A ayant k éléments	nombre de sous-ensembles
0	$\emptyset$	1
1	$\{0\},\{1\},\{2\},\{3\}$	4
2	$\{0,1\},\{0,2\},\{0,3\},\{1,2\},\{1,3\},\{2,3\}$	6
3	$\{0,1,2\},\{0,1,3\},\{0,2,3\},\{1,2,3\}$	4
4	$\{0,1,2,3\}$	1

Ainsi, $\wp(A)$ contient 16 éléments, soit $2^{|A|}$.

$$\wp(A) = \big\{\emptyset,\{0\},\{1\},\{2\},\{3\},\{0,1\},\{0,2\},\{0,3\},\{1,2\},\{1,3\},\{2,3\},\{0,1,2\},\{0,1,3\},\{0,2,3\},\{1,2,3\},\{0,1,2,4\}\big\}$$

Exemple 1.35

Décrivez l'ensemble des parties de A, où $A = \{\} = \emptyset$.

Solution :

k	Sous-ensembles de A ayant k éléments	nombre de sous-ensembles
0	$\emptyset$	1

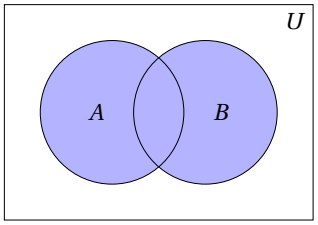
Ainsi, bien que A ne possède aucun élément, $\wp(A)$ en contient un : $\wp(A) = \big\{\emptyset\big\}$

1.3.4 Opérations sur les ensembles $\cap, \cup \oplus, -$

Soit U l'ensemble universel et A et B des sous-ensembles de U. Les opérations suivantes génèrent des sous-ensembles de U.

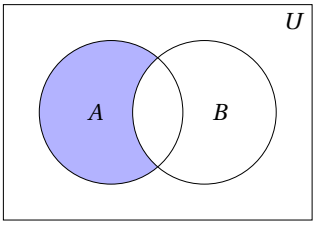
Union :

$A \cup B = \{x \in U \mid x \in A \vee x \in B\}$



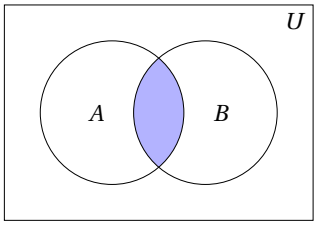
Différence :

$A - B = \{x \in U \mid x \in A \wedge x \notin B\} = A \backslash B$



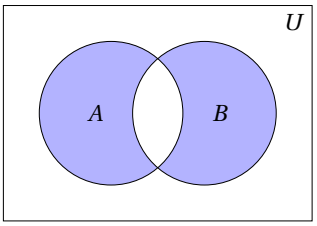
Intersection :

$A \cap B = \{x \in U \mid x \in A \wedge x \in B\}$



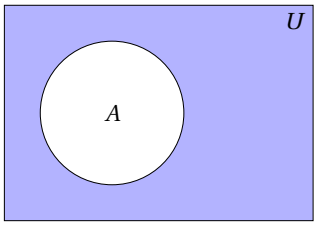
Différence symétrique :

$A \oplus B = \{x \in U \mid x \in A \oplus x \in B\}$



Complément :

$\overline{A} = \{x \in U \mid x \notin A\} = U - A$



La table suivante présente les propriétés des opérations sur les ensembles.

TABLE 8 Propriétés des ensembles

$A \cap U = A$ $A \cup \emptyset = A$	Identité
$A \cup U = U$ $A \cap \emptyset = \emptyset$	Domination
$A \cup A = A$ $A \cap A = A$	Idempotence
$\overline{\overline{A}} = A$	Double négation
$A \cup B = B \cup A$ $A \cap B = B \cap A$	Commutativité
$A \cup (B \cup C) = (A \cup B) \cup C$ $A \cap (B \cap C) = (A \cap B) \cap C$	Associativité
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	Distributivité
$\overline{A \cap B} = \overline{A} \cup \overline{B}$ $\overline{A \cup B} = \overline{A} \cap \overline{B}$	Lois de De Morgan
$A \cup (A \cap B) = A$ $A \cap (A \cup B) = A$	Absorption
$A \cup \overline{A} = U$ $A \cap \overline{A} = \emptyset$	Négation

Exemple 1.36

Utilisez la table des propriétés des ensembles pour obtenir une expression simplifiée de l'ensemble

$$\overline{\overline{A \cap B \cap \overline{C}} \cup \overline{\overline{A} \cap B}}.$$

Solution :

$$\begin{aligned}
\overline{\overline{A \cap B \cap \overline{C}} \cup \overline{\overline{A} \cap B}} &= \overline{\overline{A \cap B \cap \overline{C}}} \cap \overline{\overline{\overline{A} \cap B}} && \text{De Morgan} \\
&= (A \cap B \cap \overline{C}) \cap (\overline{A} \cap B) && \text{Complément du complément : } \overline{\overline{A}} = A \\
&= (A \cap \overline{A}) \cap B \cap \overline{C} \cap B && \text{Commutativité et associativité de l'intersection} \\
&= \emptyset \cap B \cap \overline{C} \cap B && \text{Complément : } A \cap \bar{A} = \emptyset \\
&= \emptyset && \text{Domination : intersection avec } \emptyset \text{ donne } \emptyset.
\end{aligned}$$

1.3.5 Représentation de sous-ensembles par trains de bits

On peut représenter un sous-ensemble d'un ensemble à l'aide d'un train de bits. Pour ce faire, il faut fixer un ordre pour les éléments de A, puis construire le train de bits en posant `1` à la position j si le j-ième élément appartient au sous-ensemble et `0` sinon.

Par exemple, si $A = \{0,1,2,4\}$, en choisissant l'ordre croissant, on obtient les représentations suivantes.

train de bits	sous-ensemble de A
`0000`	$\emptyset$
`1010`	$\{0,2\} = B$
`0011`	$\{2,4\} = C$
`0010`	$\{2\} = D$
`1011`	$\{0,2,4\} = E$
`1100`	$\{0,1\} = F$

Remarquons que les opérations d'union, d'intersection et de complément sur les sous-ensembles peuvent alors être effectuées directement sur les trains de bits correspondants.

$$\begin{array}{ccccc} \texttt{1010} & \wedge & \texttt{0011} & = & \texttt{0010} \\ B & \cap & C & = & D \end{array}$$

$$\begin{array}{ccccc} \texttt{1010} & \vee & \texttt{0011} & = & \texttt{1011} \\ B & \cup & C & = & E \end{array}$$

$$\begin{array}{ccc} \sim\texttt{0011} & = & \texttt{1100} \\ \overline{C} & = & F \end{array}$$

Exercices

L'exercice suivant n'est pas facile. Il vous permettra de vous exercer à traduire des énoncés à l'aide des quantificateurs existentiels et universels (voir page 20) et à **rédiger différents types de preuves** (voir page 46), le tout avec différents ensembles de nombres.

1.55 Réécrivez chacun des énoncés à l'aide de quantificateurs, de connecteurs et des prédicats suivants : $Q(x)$: x est rationnel, $P(x)$: x est pair. Sauf pour le (a), l'univers du discours est l'ensemble $\mathbb{R}$ des nombres réels.

De plus, pour chaque énoncé, dites s'il est **vrai ou faux**. Justifiez en donnant une preuve, un exemple ou un contre-exemple selon le cas.

Rappelons que $\sqrt{2}$ est irrationnel, tel que prouvé à la page 50.

(a) Pour tout nombre entier n, si n^2 est pair, alors n est pair.

(b) La somme d'un nombre rationnel et d'un nombre irrationnel est irrationnelle.

(c) Quand on multiplie un nombre rationnel non nul avec un nombre irrationnel, on obtient un nombre irrationnel.

(d) Le nombre $\frac{\sqrt{2}}{3}$ est irrationnel.

(e) Le nombre $\frac{1}{\sqrt{2}}$ est irrationnel.

(f) Si un nombre est irrationnel, alors son inverse l'est aussi. Rappel : l'inverse de x est $1/x$.

(g) La somme de deux nombres irrationnels est irrationnelle.

(h) Le produit de deux nombres irrationnels est irrationnel.

(i) L'inverse d'un nombre rationnel non nul est rationnel.

(j) L'inverse d'un nombre irrationnel est irrationnel.

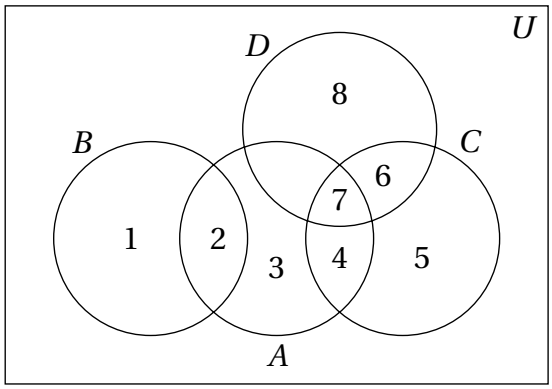
1.56 Vrai ou faux.

(a) $\mathbb{N} \in \mathbb{Z}$ (b) $\mathbb{N} \subseteq \mathbb{Z}$ (c) $\mathbb{N} \subset \mathbb{Z}$ (d) $\emptyset \in \mathbb{N}$ (e) $\emptyset \subseteq \mathbb{N}$ (f) $\mathbb{N} \subseteq \mathbb{N}$ (g) $\emptyset \subset \mathbb{N}$ (h) $\{1,2\} \subseteq \wp(\{1,2,3\})$ (i) $\{1,2\} \in \wp(\{1,2,3\})$ (j) $\{\{1\}\} \subseteq \wp(\{1,2,3\})$

1.57 Vrai ou faux.

(a) $x \in \{x\}$ (b) $x \subseteq \{x\}$ (c) $\{x\} \subset \{x\}$ (d) $\{x\} \in \{x\}$ (e) $\{x\} \subseteq \{\{x\}\}$ (f) $\emptyset \subseteq \{x\}$ (g) $\emptyset \in \{x\}$

1.58 Le diagramme de Venn suivant représente fidèlement les ensembles A, B, C, D et U. Décrivez en extension les ensembles suivants.



(a) $A \cup B$
(b) $A \cap B$
(c) $A \cap \overline{C}$
(d) $A \cap C \cap D$
(e) $\wp(B)$
(f) $\overline{U}$
(g) $\overline{A \cup C}$
(h) $B \cap D$
(i) $C - A$
(j) $D - B$
(k) $A \oplus C$
(l) $D \times B$

1.59 Soit X l'ensemble des 6 employés d'une petite équipe et R l'ensemble des 5 répertoires de leur système informatique. Désignons par $\boldsymbol{M(x,B)}$ l'énoncé *« l'employé x peut **modifier** le répertoire B »* et par $\boldsymbol{L(x,B)}$ l'énoncé *« l'employé x peut **lire** le répertoire B »*. Le tableau suivant indique les caractéristiques des employés : un L dans une case désigne que l'employé peut lire le répertoire de cette ligne, un M signifie qu'il peut le modifier, tandis que l'absence d'une lettre L ou M signifie que l'employé ne peut lire ou modifier le répertoire.

Décrivez chacun des ensembles suivants en extension. De plus, pour chacun des ensembles, donnez sa cardinalité.

(a) $A = \{r \in R \mid M(\text{Alice}, r)\}$
(b) $B = \{x \in X \mid M(x, P)\}$
(c) $C = \{x \in X \mid L(x, K)\}$
(d) $D = \{r \in R \mid \forall x \in X, L(x, r)\}$
(e) $E = \{r \in R \mid \exists x \in X, \neg L(x, r)\}$
(f) $D \cup E$
(g) $D \cap E$
(h) $B \cap C$
(i) $X - B$
(j) $\overline{E}$

R \ X	Alice	Bartez	Guy	Julien	Manon	Ugo
K		L	L, M			
J	L, M					
P	L	L	L, M	L, M	L, M	L, M
S			L, M		L, M	
Z	L	L	L, M	L	L	L

1.60 Soit $M = \{m_1, m_2, m_3, m_4, m_5\}$ l'ensemble des magasins d'une bannière et $P = \{p_1, p_2, \ldots, p_7\}$ l'ensemble des produits vendus. Dans tableau suivant, le symbole 1 dans une case désigne que le produit est vendu par le magasin, tandis qu'une case vide signifie le contraire.

Notons $L_i \subseteq P$ l'ensemble des produits vendus par le magasin i. Par exemple, on a $L_1 = \{p_3, p_4, p_5, p_6\}$.

Décrivez chacun des ensembles suivants en extension. De plus, pour chacun des ensembles, donnez sa cardinalité.

(a) $L_1 \cup L_2$

(b) $L_2 \cup L_4$

(c) $\big(L_1 \cup L_2\big) \cap L_4$

(d) $L_2 - L_4$

(e) $\overline{L_1 \cup L_2}$

(f) $\bigcup_{i=2}^{4} L_i$

(g) $\bigcap_{i=3}^{5} L_i$

P \ M	m_1	m_2	m_3	m_4	m_5
p_1		1		1	1
p_2		1		1	
p_3	1	1			
p_4	1		1		
p_5	1				1
p_6	1	1		1	
p_7			1		

1.61 Dans le contexte de l'exercice 1.60, déterminez si chacun des énoncés suivants est vrai ou faux.

(a) $L_4 \subset L_2$

(b) $L_4 \subseteq L_2$

(c) $L_1 = L_2$

(d) $p_4 \subset L_3$

(e) $p_4 \in L_3$

(f) $m_4 \in M$

(g) $L_2 \in P$

(h) $L_2 \subset P$

(i) $\bigcup_{i=2}^{5} L_i = P$

(j) $\{p_2, p_4\} \subseteq P$

P \ M	m_1	m_2	m_3	m_4	m_5
p_1		1		1	1
p_2		1		1	
p_3	1	1			
p_4	1		1		
p_5	1				1
p_6	1	1		1	
p_7			1		

1.62 Dans le contexte de l'exercice 1.60, on définit

$$x_{ij} = \begin{cases} 1 & \text{si le produit } p_i \text{ est vendu par le magasin } m_j \\ 0 & \text{sinon} \end{cases}.$$

Associez chacun des énoncés suivants à une des 14 phrases de la liste. De plus, déterminez si chacun des énoncés est vrai ou faux.

(a) $\sum_{i=1}^{7} x_{i3} \geq 3$

(b) $\sum_{j=1}^{5} x_{3j} \geq 3$

(c) $\forall j \in \{1,2,3,4,5\}, \sum_{i=1}^{7} x_{ij} \geq 3$

(d) $\forall i \in \{1,2,3,4,5,6,7\}, \sum_{j=1}^{5} x_{ij} \geq 3$

(e) $\exists j \in \{1,2,3,4,5\}, x_{3j} = 0$

(f) $\sum_{j=1}^{5} x_{3j} \leq 3$

(g) $\exists j \in \{1,2,3,4,5\}, \sum_{i=1}^{7} x_{ij} > 3$

1. Le magasin 3 vend au moins 3 produits.
2. Le magasin 3 vend moins de 3 produits.
3. Le magasin 3 vend au plus 3 produits.
4. Le magasin 3 vend plus de 3 produits.
5. Le magasin 3 vend au moins 3 produits.
6. Le magasin 3 vend moins de 3 produits.
7. Le produit 3 est vendu dans moins de 3 magasins.
8. Le produit 3 est vendu dans au moins 3 magasins.
9. Le produit 3 est vendu dans au plus 3 magasins.
10. Au moins un des magasins ne vend pas le produit 3.
11. Au moins un des produits n'est pas vendu dans le magasin 3.
12. Au moins un des magasins vend plus de 3 produits.
13. Chaque produit est disponible dans au moins 3 magasins.
14. Chaque magasin vend au moins 3 produits.

1.63 Chacune des expressions suivantes désigne un ensemble qui peut être décrit par une expression plus simple. Utilisez la table des propriétés des ensembles pour obtenir une expression simplifiée. Indiquez la propriété utilisée à chaque étape.

(a) $\overline{A \cap B \cap C} \cup C$

(b) $A \cap (H \cup D \cup A)$

(c) $\overline{(A \cap B) \cup \overline{B}}$

(d) $(\overline{A \cup B}) \cup \overline{B}$

(e) $\overline{(A \cup \overline{B}) \cup (C \cup \overline{A})}$

1.64 Écrivez une expression équivalente à celle présentée en utilisant uniquement les connecteurs logiques $\wedge$ ou $\vee$, les symboles d'inégalités $\leq$ ou $\geq$ et le symbole d'égalité $=$.

(a) $x \in [2, 4]$

(b) $x \in [2, \infty[$

(c) $x \in [2, 4] \cup [7, 9]$

(d) $x \in]-\infty, 0] \cup [10, 15]$

(e) $x \in [2, 4[\cup]4, 8]$

1.4 Fonctions

Définition 1.17 : Fonction

Une **fonction** f d'un ensemble A vers un ensemble B est une règle qui, à chaque élément a de l'ensemble A, associe un et un seul élément b de l'ensemble B. Cet élément b est noté $f(a)$. On écrit alors parfois $(a,b) \in f$.

La notation usuelle pour désigner une fonction f d'un ensemble A vers un ensemble B est

$$f : A \longrightarrow B$$

L'ensemble A est appelé le **domaine** de la fonction f, noté $\mathrm{Dom}(f)$, et le sous-ensemble de B formé des éléments atteints par f est appelé l'**image** de f, noté $\mathrm{Im}(f)$.

$$Im(f) = \{b \in B \mid \exists a \in A,\ f(a) = b\} \subseteq B$$

Par ailleurs, on peut aussi voir une fonction f de A vers B comme un sous-ensemble du produit cartésien $A \times B$ ayant la propriété suivante :

$$\forall a \in A,\ \exists! b \in B,\ (a,b) \in f$$

où le symbole $\exists!$ désigne « **il existe un et un seul** ».

Exemple 1.37

Considérons T_8, l'ensemble des trains de bits de longueur 8 et la fonction $f : T_8 \longrightarrow \mathbb{N}$ définie par

$$f(t) = \text{nombre de 0 dans le train de bits } t.$$

Par exemple, $f(11001011) = 3$. Donnez le domaine et l'image de la fonction f.

Solution :

On a

$$\mathrm{Dom}(f) = T_8 \quad \text{et} \quad \mathrm{Im}(f) = \{0,1,2,3,4,5,6,7,8\}.$$

Exemple 1.38

Soit $M = \{m_1, m_2, m_3, m_4, m_5\}$ l'ensemble des magasins d'une bannière et $P = \{p_1, p_2, \ldots, p_7\}$ l'ensemble des produits vendus. Étant donné un magasin $m \in M$ et un produit $p \in P$, on définit

$$f(m,p) = \begin{cases} 1 & \text{si le magasin } m \text{ vend le produit } p \\ 0 & \text{sinon} \end{cases}.$$

P \ M	m_1	m_2	m_3	m_4	m_5
p_1	0	1	0	1	1
p_2	0	1	0	1	0
p_3	1	1	0	0	0
p_4	1	0	1	0	0
p_5	1	0	0	0	1
p_6	1	1	0	1	0
p_7	0	0	1	0	0

Donnez le domaine et l'image de la fonction f.
De plus, donnez $f(m_3, p_4)$ si les images de la fonction f sont données par le tableau ci-dessus.

Solution :

On a

$$\text{Dom}(f) = M \times P \quad \text{et} \quad \text{Im}(f) = \{0,1\}.$$

De plus, $f(m_3, p_4) = 1$.

Exemple 1.39

Dans le contexte de l'exemple précédent, étant donné un magasin $m \in M$, on définit

$$v(m) = \text{ensemble des produits vendus dans le magasin } m.$$

Donnez $v(m_2)$.
De plus, déterminez si l'image de la fonction v est un sous-ensemble de P ou de l'ensemble des parties de P.

Solution :

On a

$$v(m_2) = \{p_1, p_2, p_3, p_6\} \in \wp(P).$$

Ainsi, l'image de la fonction v est un sous-ensemble de l'ensemble des parties de P :

$$\text{Im}(v) \subseteq \wp(P).$$

1.4.1 Fonctions plancher et plafond

Définition 1.18 : Fonctions plancher et plafond

La fonction **plancher** associe à tout nombre réel x, le plus grand entier n tel que $n \leq x$. On note $\lfloor x \rfloor = n$. La fonction **plafond** associe à tout nombre réel x, le plus petit entier n tel que $n \geq x$. On note $\lceil x \rceil = n$.

Exemple 1.40

$$\left\lfloor \frac{1}{3} \right\rfloor = 0, \quad \left\lceil \frac{1}{3} \right\rceil = 1, \quad \lfloor -9.2 \rfloor = -10 \quad \text{et} \quad \lceil -9.2 \rceil = -9.$$

Théorème 1.5 : Propriétés des fonctions plancher et plafond

1. $\lfloor x \rfloor = n \leftrightarrow n \leq x < n+1$
2. $\lceil x \rceil = n \leftrightarrow n-1 < x \leq n$
3. $x-1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x+1$

Exemple 1.41

Un magasin vend de la farine en sacs de 2 kg uniquement, au coût de 5$ le sac.

Donnez une formule décrivant chacune des fonctions suivantes :

- la fonction m qui retourne le nombre minimal de sacs que l'on doit acheter si l'on a besoin de x kg de farine ;
- la fonction s qui retourne le nombre maximal de sacs que l'on peut acheter avec d dollars ;
- la fonction c qui retourne la quantité de farine à commander si l'on a besoin de x kg de farine.

Solution :

$$m(x) = \left\lceil \frac{x}{2} \right\rceil \qquad s(d) = \left\lfloor \frac{d}{5} \right\rfloor \qquad c(x) = 2\left\lceil \frac{x}{2} \right\rceil.$$

Chapitre 2

Modélisation

Afin d'optimiser un processus, il est essentiel de commencer par le modéliser. Il s'agit d'identifier la quantité à optimiser, les variables et les paramètres, établir la fonction à maximiser ou à minimiser en précisant son domaine de validité, traduire les contraintes physiques ou économiques qui entrent en jeu sous la forme d'équations ou d'inéquations. Une fois le modèle établi, on utilise diverses méthodes pour obtenir la solution optimale. Par exemple, en calcul différentiel [1], on dérive la fonction à maximiser, on trouve ses points critiques, etc.

La traduction d'un problème particulier en un modèle mathématique n'est pas toujours facile. C'est précisément cette habileté que nous souhaitons développer dans ce chapitre. Nous présentons quelques exemples simples du processus de modélisation, sans nous attarder aux techniques de résolution. Cette paire d'habiletés, modélisation et résolution, sera approfondie dans plusieurs cours du baccalauréat en Génie des opérations et de la logistique.

2.1 Exemple 1 : plan de production de l'entreprise 3P

L'entreprise 3P fabrique trois produits différents. Chacun d'eux nécessite un passage dans chacun des trois ateliers : coupe, assemblage et finition.

- Le produit 1 engendre une contribution marginale de 120 $ par unité vendue. Chaque unité vendue requiert 2 heures de coupe, 4 heures d'assemblage et 1 heure de finition.
- Le produit 2 engendre une contribution marginale de 150 $ par unité vendue. Chaque unité vendue requiert 1 heure de coupe, 5 heures d'assemblage et 2 heures de finition.
- Le produit 3 engendre une contribution marginale de 90 $ par unité vendue. Chaque unité vendue requiert 1 heure de coupe, 3 heures d'assemblage et 1 heure de finition.

Les nombres d'heures disponibles dans les ateliers le mois prochain sont de 400 h à l'atelier 1 (coupe), 1400 h à l'atelier 2 (assemblage) et 500 h à l'atelier 3 (finition).

La demande est très forte : toutes les unités produites seront vendues.

Combien d'unités de chaque produit devrait fabriquer l'entreprise le mois prochain afin de maximiser sa contribution marginale totale [2] ?

1. Voir la section 3.2 Optimisation des Notes de cours de MAT145. https://cours.etsmtl.ca/seg/GSAVARD/MAT145V1.pdf
2. Rappel : la contribution marginale unitaire représente le prix d'un article moins le coût pour le fabriquer (on suppose ici que le coût variable unitaire est constant, quelle que soit la quantité fabriquée). Le profit est donc égal à la contribution marginale totale de laquelle on soustrait les coûts fixes. Ainsi, en maximisant la contribution marginale totale, on maximise aussi le profit.

2.1.1 Un premier modèle pour l'entreprise 3P

Un premier modèle pour l'entreprise 3P

But : maximiser la contribution marginale totale en \$, que l'on note z.

Variables : $\forall i \in \{1,2,3\}$, x_i = nombre d'unités à fabriquer du produit i.

Contraintes naturelles : c'est ainsi que l'on appelle le domaine des variables. On ne peut pas avoir un nombre d'unités négatif ou fractionnaire. Ainsi,

$$\forall i \in \{1,2,3\},\ x_i \in \mathbb{N}.$$

Fonction-objectif : c'est ainsi que l'on appelle la fonction à optimiser

$$z = 120x_1 + 150x_2 + 90x_3.$$

Contraintes : pour chacun des ateliers, le temps d'utilisation requis doit être inférieur ou égal au temps disponible. Ainsi, on obtient

$$\begin{array}{lrcrcrcr} \text{atelier 1}: & 2x_1 & + & 1x_2 & + & 1x_3 & \leq & 400 \\ \text{atelier 2}: & 4x_1 & + & 5x_2 & + & 3x_3 & \leq & 1400 \\ \text{atelier 3}: & 1x_1 & + & 2x_2 & + & 1x_3 & \leq & 500 \end{array}$$

Explication de la formule utilisée pour les contraintes

La contrainte sur le temps disponible dans l'atelier 1 tient compte du temps requis pour la fabrication de chacun des produits :

(temps requis dans l'atelier 1 pour fabriquer une unité du produit 1) · (nombre d'unités du produit 1) +
(temps requis dans l'atelier 1 pour fabriquer une unité du produit 2) · (nombre d'unités du produit 2) +
(temps requis dans l'atelier 1 pour fabriquer une unité du produit 3) · (nombre d'unités du produit 3)
≤ temps disponible dans l'atelier 1

Ainsi,

$$\frac{2\,\text{h}}{\text{u}} \cdot x_1\,\text{u} + \frac{1\,\text{h}}{\text{u}} \cdot x_2\,\text{u} + \frac{1\,\text{h}}{\text{u}} \cdot x_3\,\text{u} \quad \leq \quad 400\,\text{h}$$

donc

$$\text{Atelier 1}: \quad 2x_1 \quad + \quad 1x_2 \quad + \quad 1x_3 \quad \leq \quad 400.$$

Il en va de même pour chacun des ateliers.

Autre forme pour la fonction-objectif

En introduisant les paramètres

$$c_1 = 120, \quad c_2 = 150, \quad c_3 = 90$$

comme coefficients des variables x_p, pour p allant de 1 à 3, et en définissant l'ensemble $P = \{1,2,3\}$, on peut écrire la fonction-objectif de façon plus compacte :

$$z \;=\; c_1 x_1 + c_2 x_2 + c_3 x_3 \;=\; \sum_{p \in P} c_p \cdot x_p.$$

Ceci se lit « la somme des $c_p \cdot x_p$ pour toutes les valeurs de p appartenant à l'ensemble P ».

2.1.2 Utilisation d'un tableur muni d'un solveur

Notre attention dans ce cours portera surtout sur la modélisation : la rédaction de modèles simples et de modèles algébriques à partir de situations décrites en mots, et, réciproquement, la compréhension des formules utilisées dans certains modèles présentés. D'autres cours du baccalauréat en Génie des opérations et de la logistique présenteront les aspects géométriques et algorithmiques de la recherche de solutions optimales. Nous présentons tout de même ici un premier outil de résolution : le solveur d'Excel. **L'installation de ce module complémentaire**[3] **et son utilisation seront présentées en classe. Dans cette section, nous incluons simplement quelques captures d'écran.**

Pour utiliser le solveur d'Excel, il faut d'abord construire une feuille de calcul représentant le modèle, comme à la figure 2.1 Dans cet exemple, on a colorié les cellules selon le **code de couleurs** suivant :

- Les cellules correspondant aux valeurs des **variables** du modèle sont en bleu.
- Les **coefficients des contraintes** sont en jaune pâle et les membres de droites en jaune foncé.
- Les **coefficients de la fonction-objectif** sont en saumon.
- La **fonction-objectif** est en vert : elle est nommée « z ».
- La colonne « H » indique vrai ou faux selon que le nombre d'heures utilisées respecte ou non le temps disponible pour cet atelier. Nous avons utilisé un affichage conditionnel : vert pâle si égal à « vrai » et rouge si égal à « faux ».

Une fois la feuille de calcul créée, on peut tester si une affectation des variables satisfait ou non les contraintes. Par exemple, l'affectation $x_1 = 150$, $x_2 = 50$, $x_3 = 100$ ne satisfait pas à la contrainte de l'atelier 1 : ce n'est pas une solution admissible (voir figure 2.1). Par ailleurs, $x_1 = 130$, $x_2 = 30$, $x_3 = 110$ satisfait aux 3 contraintes : c'est une **solution admissible** qui engendre 30 000$ (voir figure 2.2). Par contre, il ne s'agit pas d'une **solution optimale**. En effet, la figure 2.3 montre que l'on peut faire mieux, par exemple avec $x_1 = 100$, $x_2 = 200$, $x_3 = 0$, qui engendre 42 000$ (contribution marginale). Cette solution est optimale ; nous l'avons obtenue avec l'outil « solveur » .

3. Voir la documentation présentée sur `https://support.microsoft.com/en-ie/office/load-the-solver-add-in-in-excel-612926fc-d53b-46b4-872c-e24772f078ca`.

	A	B	C	D	E	F	G	H
1	**But : maximiser la fonction-objectif, soit la contribution marginale totale du mois prochain.**							
2								
3	**Fonction-objectif** : $z = \Sigma\ c_p\ x_p$							
4	**Ensemble** des produits P	1	2	3				
5	**Noms des variables :** x_p	x_1	x_2	x_3				
6	**Définition** x_p **:** nombre d'unités du produit p fabriquées le mois prochain							
7					z =			
8	**Coefficients** c_p (en $/unité) et **valeur de z**	120	150	90	**34500**			
9								
10	**Contraintes (en h/unité)**				Nb heures utilisées (t)	signe	Nb d'heures disponibles (d)	Vrai ou faux
11	Atelier 1 : coupe	2	1	1	450	<=	400	FAUX
12	Atelier 2 : assemblage	4	5	3	1150	<=	1400	VRAI
13	Atelier 3 : finition	1	2	1	350	<=	500	VRAI
14								
15	**Contraintes naturelles** : x_p>=0 et ENT							
16								
17	**Valeurs des variables** x_p	**150**	**50**	**100**				

Figure 2.1 Feuille de calcul du modèle de l'entreprise 3P. Les valeurs $x_1 = 150$, $x_2 = 50$, $x_3 = 100$ ne satisfont pas à la contrainte de l'atelier 1.

	A	B	C	D	E	F	G	H
1	**But : maximiser la fonction-objectif, soit la contribution marginale totale du mois prochain.**							
2								
3	**Fonction-objectif** : $z = \Sigma\ c_p\ x_p$							
4	**Ensemble** des produits P	1	2	3				
5	**Noms des variables :** x_p	x_1	x_2	x_3				
6	**Définition** x_p **:** nombre d'unités du produit p fabriquées le mois prochain							
7					z =			
8	**Coefficients** c_p (en $/unité) et **valeur de z**	120	150	90	**30000**			
9								
10	**Contraintes (en h/unité)**				Nb heures utilisées (t)	signe	Nb d'heures disponibles (d)	Vrai ou faux
11	Atelier 1 : coupe	2	1	1	400	<=	400	VRAI
12	Atelier 2 : assemblage	4	5	3	1000	<=	1400	VRAI
13	Atelier 3 : finition	1	2	1	300	<=	500	VRAI
14								
15	**Contraintes naturelles** : x_p>=0 et ENT							
16								
17	**Valeurs des variables** x_p	**130**	**30**	**110**				

Figure 2.2 Feuille de calcul du modèle de l'entreprise 3P. Les valeurs $x_1 = 130$, $x_2 = 30$, $x_3 = 110$ satisfont à toutes les contraintes et engendrent 30 000$.

	A	B	C	D	E	F	G	H
1	**But : maximiser la fonction-objectif, soit la contribution marginale totale du mois prochain.**							
2								
3	**Fonction-objectif** : $z = \Sigma\ c_p\ x_p$							
4	**Ensemble** des produits P	1	2	3				
5	**Noms des variables :** x_p	x_1	x_2	x_3				
6	**Définition** x_p **:** nombre d'unités du produit p fabriquées le mois prochain							
7					z =			
8	**Coefficients** c_p (en $/unité) et **valeur de z**	120	150	90	**42000**			
9								
10	**Contraintes (en h/unité)**				Nb heures utilisées (t)	signe	Nb d'heures disponibles (d)	Vrai ou faux
11	Atelier 1 : coupe	2	1	1	400	<=	400	VRAI
12	Atelier 2 : assemblage	4	5	3	1400	<=	1400	VRAI
13	Atelier 3 : finition	1	2	1	500	<=	500	VRAI
14								
15	**Contraintes naturelles** : x_p>=0 et ENT							
16								
17	**Valeurs des variables** x_p	**100**	**200**	**0**				

Figure 2.3 Feuille de calcul du modèle de l'entreprise 3P. Les valeurs $x_1 = 100$, $x_2 = 200$, $x_3 = 0$ satisfont à toutes les contraintes et engendrent 42 000$, ce qui est la contribution marginale maximale.

2.1.3 Présentation d'une solution optimale

Nous pouvons maintenant fournir une réponse claire à la question de l'exemple 1 présentée à la page 65.

Une solution optimale pour l'exemple 1 (plan de production de l'entreprise 3P)

Afin de maximiser sa contribution marginale totale le mois prochain, l'entreprise 3P peut répartir sa production ainsi : 100 unités du produit 1, 200 unités du produit 2 et aucune du produit 3. La contribution marginale totale sera alors de 42 000$.

2.1.4 Bonne pratique : nommer des plages de cellules d'une feuille de calcul

Il est utile de nommer des plages de valeurs de la feuille de calcul pour s'y référer au lieu d'utiliser les adresses des cases. Par exemple, comme le montre la figure 2.4, la formule de la case `E8`, qui correspond à la fonction-objectif, est simplement

```
SOMMEPROD(cp;xp)
```

au lieu de

```
B8*B17+C8*C17+D8*D17.
```

Ceci est d'autant plus utile s'il y a des centaines ou milliers de variables.

Le **Gestionnaire de noms** affiche la liste des noms et permet d'en créer, modifier ou supprimer (voir figure 2.5).

Une des façons de **nommer une cellule** ou une plage de cellules en Excel est de la sélectionner et d'entrer le nom voulu dans la case supérieure gauche de l'écran (voir figure 2.4).

Nommer des plages de cellules permet aussi d'indiquer plusieurs contraintes en une seule ligne dans les paramètres du solveur (voir figure 2.6).

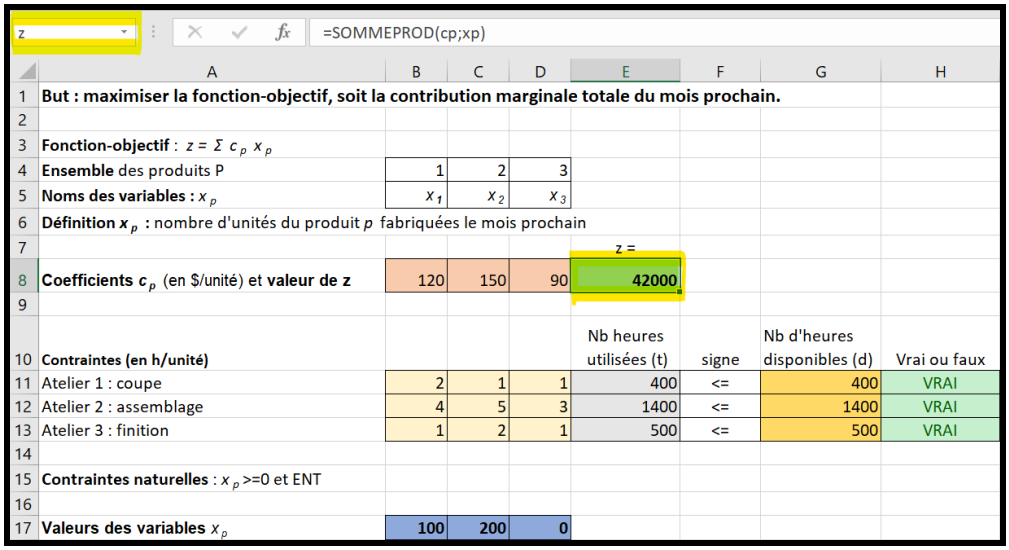


Figure 2.4 Pour nommer une plage de cellules, on la sélectionne puis on entre le nom voulu dans la case supérieure gauche de l'écran.

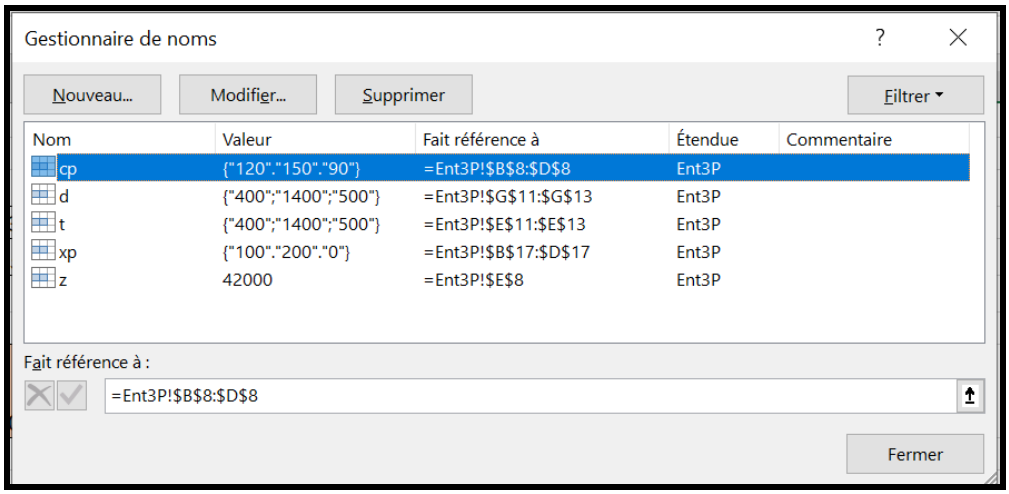


Figure 2.5 Le **Gestionnaire de noms** est dans le **menu Formules**.

Paramètres du solveur

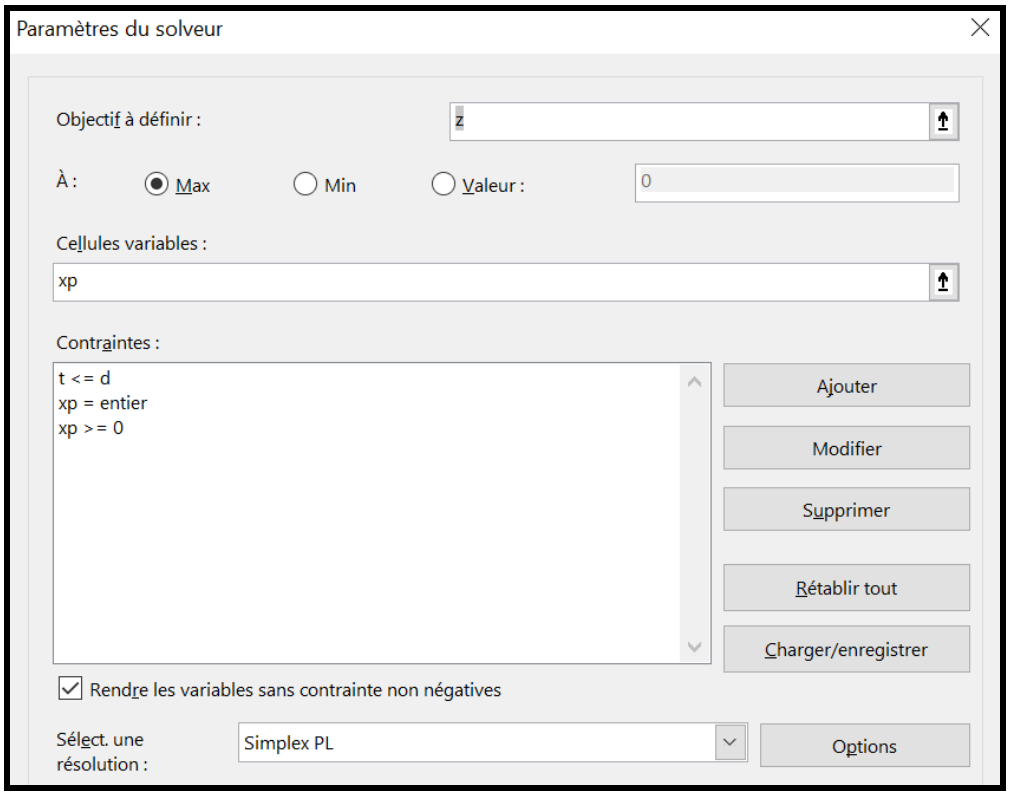


Figure 2.6 Utilisation du solveur pour maximiser la contribution marginale, en ayant préalablement défini les noms des plages z, xp, t, et d. Le **solveur** est dans le menu **Données**.

La vidéo *Optimisation: création d'une feuille de calcul et utilisation du solveur de Excel*[4] présente pas à pas la préparation de la feuille de calcul du présent exemple et l'utilisation du solveur. Pour plus de détails, nous vous suggérons de consulter *La résolution de modèles linéaires par Excel*, une annexe du livre *Méthodes d'optimisation pour la gestion*, de Norbert, Ouellet, Parent, paru chez Chenelière Éducation.

4. https://youtu.be/hoVHVv2diF4

2.1.5 Un modèle algébrique: compact et adaptable

Si l'on désire maximiser les profits pour une compagnie qui fabrique 300 produits nécessitant 50 ateliers, il faudra construire un nouveau modèle avec 300 variables et écrire une contrainte pour chacun des 50 ateliers. Pour gagner en temps et en flexibilité, il vaut mieux construire un modèle algébrique: toutes les valeurs numériques sont remplacées par des paramètres (des lettres qui désignent des constantes).

Le premier avantage d'un modèle algébrique est qu'on peut l'adapter facilement: il suffit de modifier les ensembles ou les valeurs des paramètres pour mettre à jour le modèle. Les équations et inéquations ne changent pas, de même que la fonction-objectif. Le second avantage est sa forme compacte: elle permet de décrire des dizaines, centaines ou milliers de contraintes en une seule ligne, grâce à l'emploi d'indices, du symbole de sommation et de quantificateurs.

Un modèle algébrique pour l'entreprise 3P

But: maximiser la contribution marginale totale en $, que l'on note z.

Ensemble:

$P = \{1,2,3,\ldots,n\}$ ensemble des numéros des produits fabriqués.
$Q = \{1,2,3,\ldots,m\}$ ensemble des numéros des ateliers.

Variables: $\forall p \in P$, x_p = nombre d'unités à fabriquer du produit p.

Contraintes naturelles: $\forall p \in P$, $x_p \in \mathbb{N}$.

Paramètres:

$\forall p \in P$, c_p = contribution marginale engendré par une unité du produit p.
$\forall q \in Q$, d_q = nombre d'heures disponibles à l'atelier q.
$\forall q \in Q, \forall p \in P$, a_{qp} = nombre d'heures requises à l'atelier q pour fabriquer une unité du produit p. Les éléments a_{qp} forment une matrice de m lignes et n colonnes:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \ldots & a_{mn} \end{pmatrix}$$

Fonction-objecti:

$$z = \sum_{p \in P} c_p \cdot x_p$$

Contraintes:

$$\forall q \in Q, \sum_{p \in P} \left(a_{qp} \cdot x_p\right) \leq d_q$$

Ce modèle, et tous ceux que nous verrons par la suite, est un **modèle linéaire**: la fonction-objectif ainsi que les membres de gauche des contraintes sont des polynômes de degré 1 à n variables.

Explication de la formule utilisée pour les contraintes

La contrainte sur le temps disponible dans l'atelier q tient compte du temps requis pour la fabrication de chacun des produits :

(temps requis dans l'atelier q pour fabriquer une unité du produit 1) · (nombre d'unités du produit 1) +
(temps requis dans l'atelier q pour fabriquer une unité du produit 2) · (nombre d'unités du produit 2) +
... +
(temps requis dans l'atelier q pour fabriquer une unité du produit n) · (nombre d'unités du produit n)
≤ (temps disponible dans l'atelier q)

Ainsi,

$$\sum_{p \in P} (a_{qp} \cdot x_p) \leq d_q.$$

2.2 Exemple 2 : répartition des points de service

Un organisme public veut déployer des points de service [5] dans une ville qui est divisée en huit quartiers. Ces quartiers sont numérotés de 1 à 8 et leur disposition est donnée par la carte de la Figure 2.7.

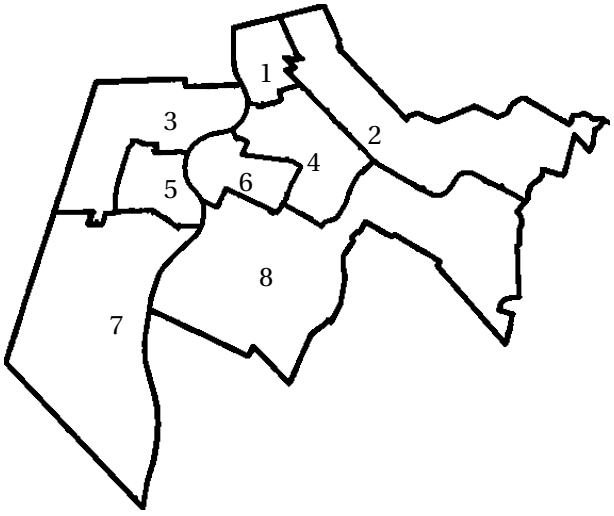


Figure 2.7 Carte des quartiers de la ville où des points de service seront déployés.

L'organisme veut s'assurer de respecter la contrainte suivante :

Chaque citoyen a accès à un point de service qui est situé
soit dans son propre quartier,
soit dans l'un des quartiers adjacents au sien.

Autrement dit, pour tout quartier i :

S'il n'y a pas de point de service dans le quartier i ,
alors il y a en au moins un dans un quartier adjacent au quartier i .

5. Par exemple, on pourrait vouloir déployer des cliniques de dépistage, des centres de dépôt de déchets dangereux, des maisons de jeunes, des points de services de la société d'assurance automobile du Québec.

Une solution évidente serait de déployer un point de service dans chaque quartier et la contrainte serait ainsi satisfaite. Ceci nécessiterait 8 points de services. On devine que l'on peut diminuer ce nombre total de points de service tout en respectant la contrainte, mais jusqu'à combien ?

Question : *Dans quels quartiers doit-on déployer des points de service si l'on souhaite en minimiser le nombre total, tout en respectant la contrainte de proximité ?*

2.2.1 Un premier modèle, simple mais peu adaptable

Construisons un modèle mathématique dans le but de choisir les quartiers pour minimiser le nombre de points de service.

Un premier modèle pour la répartition des points de service

But : minimiser le nombre total de points de service, que l'on note z.

Variables : s_1, s_2, s_3, …, s_8 indiquent si l'on déploie ou non un point de service dans le quartier i. Autrement dit :

$$\forall i \in \{1,2,3,\dots 8\},\ s_i = \begin{cases} 1 & \text{si un point de service est déployé dans le quartier } i, \\ 0 & \text{sinon.} \end{cases}.$$

Contraintes naturelles : $\forall i \in \{1,2,3,\dots 8\},\ s_i \in \{0,1\}$.

Fonction-objectif : $z = s_1 + s_2 + s_3 + \cdots + s_8$.

Contraintes : *Pour chacun des quartiers, s'il n'y a pas de point de service dans ce quartier, alors il y a en au moins un dans un quartier adjacent.* Autrement dit, le nombre de points de service dans un quartier et ses quartiers adjacents doit être supérieur ou égal à 1 :

$$\begin{array}{lccccccccccccccccc}
\text{Quartier 1 :} & s_1 & + & s_2 & + & s_3 & + & s_4 & & & & & & & & \geq & 1 \\
\text{Quartier 2 :} & s_1 & + & s_2 & & & + & s_4 & & & & & & + & s_8 & \geq & 1 \\
\text{Quartier 3 :} & s_1 & & & + & s_3 & + & s_4 & + & s_5 & + & s_6 & + & s_7 & & \geq & 1 \\
\text{Quartier 4 :} & s_1 & + & s_2 & + & s_3 & + & s_4 & & & + & s_6 & & & + & s_8 & \geq & 1 \\
\text{Quartier 5 :} & & & & & s_3 & & & + & s_5 & + & s_6 & + & s_7 & + & s_8 & \geq & 1 \\
\text{Quartier 6 :} & & & & & s_3 & + & s_4 & + & s_5 & + & s_6 & & & + & s_8 & \geq & 1 \\
\text{Quartier 7 :} & & & & & s_3 & & & + & s_5 & & & + & s_7 & + & s_8 & \geq & 1 \\
\text{Quartier 8 :} & & & s_2 & & & + & s_4 & + & s_5 & + & s_6 & + & s_7 & + & s_8 & \geq & 1.
\end{array}$$

Si l'on doit répondre à la même question pour une autre ville, alors il faut construire un nouveau modèle avec plus ou moins de variables et réécrire la contrainte pour chacun des quartiers. Pour gagner du temps et de la flexibilité, il vaut mieux construire un modèle algébrique.

2.2.2 Un modèle algébrique : compact et adaptable

Un modèle algébrique pour la répartition des points de service

But : minimiser le nombre total de points de service, que l'on note z.

Ensemble : $Q = \{1, 2, 3, \dots n\}$ ensemble des quartiers de la ville.

Variables : s_q, où $q \in Q$, indique si l'on déploie ou non un point de service dans le quartier q.

$$\forall q \in Q,\ s_q = \begin{cases} 1 & \text{si un point de service est déployé dans le quartier } q, \\ 0 & \text{sinon.} \end{cases}$$

Contraintes naturelles : $\forall q \in Q,\ s_q \in \{0, 1\}$.

Paramètres : pour modéliser le fait que deux quartiers donnés sont identiques ou adjacents, on utilise une matrice de n lignes et n colonnes ($n = 8$ dans notre exemple, voir Figure 2.8). La relation de proximité entre les quartiers est donc décrite par la matrice A,

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix},$$

où $\forall i \in Q,\ \forall j \in Q$, l'élément situé à la i^{e} ligne et j^{e} colonne de la matrice A est

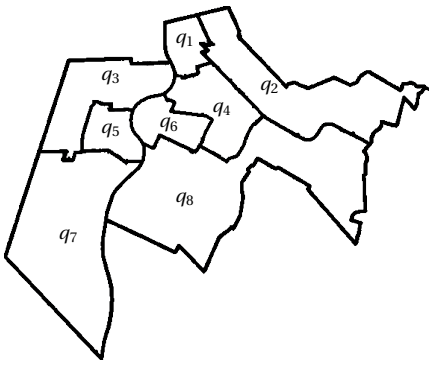
$$a_{ij} = \begin{cases} 1 & \text{si le quartier } i \text{ est adjacent au quartier } j \text{ ou si } i = j \\ 0 & \text{sinon} \end{cases}.$$

Fonction-objectif :

$$z = \sum_{q \in Q} s_q.$$

Contraintes :

$$\forall i \in Q,\ \sum_{q \in Q} \left(a_{iq} \cdot s_q\right) \geq 1.$$



i \ j	1	2	3	4	5	6	7	8
1	1	1	1	1	0	0	0	0
2	1	1	0	1	0	0	0	1
3	1	0	1	1	1	1	1	0
4	1	1	1	1	0	1	0	1
5	0	0	1	0	1	1	1	1
6	0	0	1	1	1	1	0	1
7	0	0	1	0	1	0	1	1
8	0	1	0	1	1	1	1	1

Figure 2.8 Matrice A modélisant la relation de proximité entre les quartiers.

Explication de la formule utilisée pour les contraintes

Voyons maintenant comment généraliser les 8 contraintes du modèle de la section 2.2.1 en une seule ligne. Observons de près les 2 premières contraintes :

$$\begin{aligned}
&\text{Quartier 1}: && s_1 + s_2 + s_3 + s_4 && && \geq 1\\
&\text{Quartier 2}: && s_1 + s_2 + s_4 && + s_8 && \geq 1.
\end{aligned}$$

On constate que chaque contrainte est une inéquation comportant le symbole $\geq$, dont le membre de droite est le nombre 1 et celui de gauche une somme de quelques-unes des variables s_1 à s_8. On peut aussi voir ce membre de gauche comme une somme des 8 variables multipliées par un coefficient 0 ou 1. Par exemple, pour les deux premiers quartiers, on a :

$$\begin{aligned}
&\text{Quartier 1}: && 1s_1 + 1s_2 + 1s_3 + 1s_4 + 0s_5 + 0s_6 + 0s_7 + 0s_8 && \geq 1\\
&\text{Quartier 2}: && 1s_1 + 1s_2 + 0s_3 + 1s_4 + 0s_5 + 0s_6 + 0s_7 + 1s_8 && \geq 1.
\end{aligned}$$

On remarque que les coefficients en rouge dans la première contrainte correspondent aux valeurs de la première ligne de la matrice A, alors que les coefficients de la seconde contrainte correspondent aux valeurs de la deuxième ligne de A, et ainsi de suite pour chacun des quartiers :

$$\begin{aligned}
&\text{Quartier 1}: && a_{11}s_1 + a_{12}s_2 + a_{13}s_3 + a_{14}s_4 + a_{15}s_5 + a_{16}s_6 + a_{17}s_7 + a_{18}s_8 && \geq 1\\
&\text{Quartier 2}: && a_{21}s_1 + a_{22}s_2 + a_{23}s_3 + a_{24}s_4 + a_{25}s_5 + a_{26}s_6 + a_{27}s_7 + a_{28}s_8 && \geq 1\\
&\quad\vdots\\
&\text{Quartier } i: && a_{i1}s_1 + a_{i2}s_2 + a_{i3}s_3 + a_{i4}s_4 + a_{i5}s_5 + a_{i6}s_6 + a_{i7}s_7 + a_{i8}s_8 && \geq 1\\
&\quad\vdots\\
&\text{Quartier 8}: && a_{81}s_1 + a_{82}s_2 + a_{83}s_3 + a_{84}s_4 + a_{85}s_5 + a_{86}s_6 + a_{87}s_7 + a_{88}s_8 && \geq 1.
\end{aligned}$$

Pour le quartier 1, l'indice i de a_{ij} est toujours 1. Pour le quartier 2, l'indice i est toujours 2. Ainsi, on peut écrire de manière plus compacte les contraintes comme :

$$\begin{aligned}
&\text{Quartier 1}: && \sum_{q\in Q}\left(a_{1q}\cdot s_q\right) \geq 1\\
&\text{Quartier 2}: && \sum_{q\in Q}\left(a_{2q}\cdot s_q\right) \geq 1\\
&\quad\vdots\\
&\text{Quartier 8}: && \sum_{q\in Q}\left(a_{8q}\cdot s_q\right) \geq 1.
\end{aligned}$$

Par ailleurs, l'indice i de a_{ij} varie de 1 à 8 pour chacun des quartiers. On peut donc généraliser nos contraintes en une seule ligne :

$$\forall i \in Q,\ \sum_{q\in Q}\left(a_{iq}\cdot s_q\right) \geq 1.$$

2.2.3 Utilisation d'un tableur muni d'un solveur

Pour utiliser le solveur d'Excel, il faut d'abord construire une feuille de calcul représentant le modèle, comme à la figure 2.9. Dans cet exemple, on a colorié les cellules selon le **code de couleurs** suivant :

- Les cellules contenant les **valeurs des variables** du modèle sont en bleu.
- La matrice A modélisant la relation de proximité entre les quartiers est en jaune pâle.
- Les cellules affichant le besoin minimal pour chacun des quartiers sont en jaune plus foncé.
- La cellule correspondant à la **fonction-objectif** est en vert.
- Pour chacun des quartiers, une cellule affiche le nombre de points de service qui sont accessibles, c'est-à-dire situés soit dans ce quartier, soit dans un quartier adjacent : cette colonne est en gris.
- À droite des cellules de la colonne « p », une autre cellule indique vrai ou faux selon que le nombre de points de service est supérieur ou non au besoin dans ce quartier. Nous avons utilisé un affichage conditionnel : vert si égal à « vrai » et rouge si égal à « faux ».

Une fois la feuille de calcul créée, on peut tester si un ensemble de quartiers satisfait les contraintes. Par exemple, la figure 2.9 montre que le choix des quartiers 2 et 7 (visible à la ligne 22) ne répond pas au besoin du quartier 6. Ce n'est donc pas une solution admissible.

La figure 2.10 montre que le choix des quartiers 2, 4, 6 et 8 satisfait le besoin de chaque quartier et requiert 4 points de service. Il s'agit donc d'une solution admissible.

En utilisant le solveur, on trouve une solution optimale (c'est-à-dire qui minimise la valeur de la fonction-objectif). Tel qu'indiqué à la figure 2.11, le choix des quartiers 2 et 3 satisfait les besoins de chacun des quartiers et requiert seulement 2 points de service.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**But : minimiser la fonction-objectif, soit le nombre total de points de services à déployer.**												
2													
3	**Fonction-objectif :** $z = \Sigma c_q s_q$												
4	**Ensemble** des quartiers Q	1	2	3	4	5	6	7	8				
5	**Noms des variables :** s_q	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8				
6	**Définition** s_q = 1 si un point de service est déployé dans le quartier q et 0 sinon												
7										z =			
8	**Coefficients** c_q **et valeur de z**	1	1	1	1	1	1	1	1	**2**			
9													
10	**Contraintes**									Nb de points de service accessibles (p)		Besoin (b)	p>=b
11	Quartier 1	1	1	1	1	0	0	0	0	1	>=	1	VRAI
12	Quartier 2	1	1	0	1	0	0	0	1	1	>=	1	VRAI
13	Quartier 3	1	0	1	1	1	1	1	0	1	>=	1	VRAI
14	Quartier 4	1	1	1	1	0	1	0	1	1	>=	1	VRAI
15	Quartier 5	0	0	1	0	1	1	1	1	1	>=	1	VRAI
16	Quartier 6	0	0	1	1	1	1	0	1	0	>=	1	FAUX
17	Quartier 7	0	0	1	0	1	0	1	1	1	>=	1	VRAI
18	Quartier 8	0	1	0	1	1	1	1	1	2	>=	1	VRAI
19													
20	**Contraintes naturelles :** s_q = 0 ou 1												
21													
22	**Valeurs des variables** s_q	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**				

Figure 2.9 Le choix des quartiers 2 et 7 ne répond pas au besoin du quartier 6.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**But : minimiser la fonction-objectif, soit le nombre total de points de services à déployer.**												
2													
3	**Fonction-objectif :** $z = \Sigma c_q s_q$												
4	**Ensemble** des quartiers Q	1	2	3	4	5	6	7	8				
5	**Noms des variables :** s_q	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8				
6	**Définition** s_q = 1 si un point de service est déployé dans le quartier q et 0 sinon												
7										z =			
8	**Coefficients** c_q **et valeur de z**	1	1	1	1	1	1	1	1	**4**			
9													
10	**Contraintes**									Nb de points de service accessibles (p)		Besoin (b)	p>=b
11	Quartier 1	1	1	1	1	0	0	0	0	2	>=	1	VRAI
12	Quartier 2	1	1	0	1	0	0	0	1	3	>=	1	VRAI
13	Quartier 3	1	0	1	1	1	1	1	0	2	>=	1	VRAI
14	Quartier 4	1	1	1	1	0	1	0	1	4	>=	1	VRAI
15	Quartier 5	0	0	1	0	1	1	1	1	2	>=	1	VRAI
16	Quartier 6	0	0	1	1	1	1	0	1	3	>=	1	VRAI
17	Quartier 7	0	0	1	0	1	0	1	1	1	>=	1	VRAI
18	Quartier 8	0	1	0	1	1	1	1	1	4	>=	1	VRAI
19													
20	**Contraintes naturelles :** s_q = 0 ou 1												
21													
22	**Valeurs des variables** s_q	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**				

Figure 2.10 Le choix des quartiers 2, 4, 6 et 8 satisfait le besoin de chaque quartier et requiert 4 points de service.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**But : minimiser la fonction-objectif, soit le nombre total de points de services à déployer.**												
2													
3	**Fonction-objectif :** $z = \Sigma c_q s_q$												
4	**Ensemble** des quartiers Q	1	2	3	4	5	6	7	8				
5	**Noms des variables :** s_q	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8				
6	**Définition** s_q = 1 si un point de service est déployé dans le quartier q et 0 sinon												
7										z =			
8	**Coefficients** c_q **et valeur de z**	1	1	1	1	1	1	1	1	**2**			
9													
10	**Contraintes**									Nb de points de service accessibles (p)		Besoin (b)	p>=b
11	Quartier 1	1	1	1	1	0	0	0	0	2	>=	1	VRAI
12	Quartier 2	1	1	0	1	0	0	0	1	1	>=	1	VRAI
13	Quartier 3	1	0	1	1	1	1	1	0	1	>=	1	VRAI
14	Quartier 4	1	1	1	1	0	1	0	1	2	>=	1	VRAI
15	Quartier 5	0	0	1	0	1	1	1	1	1	>=	1	VRAI
16	Quartier 6	0	0	1	1	1	1	0	1	1	>=	1	VRAI
17	Quartier 7	0	0	1	0	1	0	1	1	1	>=	1	VRAI
18	Quartier 8	0	1	0	1	1	1	1	1	1	>=	1	VRAI
19													
20	**Contraintes naturelles :** s_q = 0 ou 1												
21													
22	**Valeurs des variables** s_q	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**				

Figure 2.11 Le choix des quartiers 2 et 3 satisfait tous les besoins et requiert 2 points de service. C'est une solution optimale.

2.2.4 Présentation d'une solution optimale

Nous pouvons maintenant répondre à la question de l'exemple 2 (page 73).

Une solution optimale pour l'exemple 2 (répartition des points de service)

Afin de minimiser le nombre total de points de service tout en respectant la contrainte de proximité, l'organisme peut choisir les quartiers 2 et 3.

2.3 Exemple 3 : planification d'un horaire pour minimiser le coût des salaires.

Voici un problème d'emploi du temps (Work-flow scheduling problem). Le but est de répartir les employés sur les différentes plages horaires pour combler les besoins du département d'une compagnie, tout en minimisant le coût total des salaires.

Pour simplifier la modélisation, nous supposons ici que le salaire d'un employé ne dépend pas de ses années d'expérience, mais qu'il varie seulement en fonction de la plage horaire où il travaille. Supposons aussi que les employés de cette équipe travaillent 6 ou 12 heures d'affilée sans prendre de pauses ! Il y a 5 « plages » de travail : 4 de 12 heures et 1 de 6h, décrites dans le premier tableau 2.1. Par ailleurs, le département doit s'assurer d'avoir suffisamment d'employés pour chacun des 4 « blocs » de la journée, comme indiqué dans le second tableau (il peut y avoir plus d'employés ou autant, mais pas moins que le nombre minimal requis).

Plage	Horaire de travail	Salaire horaire en $
1	00h à 12h	34
2	06h à 18h	30
3	12h à 24h	32
4	18h à 06h	35
5	00h à 06h	31

Bloc	Horaire	Besoins minimaux (nombre d'employés)
1	00h à 06h	5
2	06h à 12h	8
3	12h à 18h	10
4	18h à 24h	6

Tableau 2.1 Présentation des salaires horaires associés aux 5 plages horaires et des besoins minimaux en nombre d'employés à différentes périodes de la journée (blocs).

Puisque les horaires de travail diffèrent des horaires des blocs identifiant les besoins minimaux, il sera pratique de décrire les plages à l'aide d'une matrice X définie par

$$x_{ij} = \begin{cases} 1 & \text{Le bloc } i \text{ est couvert par les employés de la plage } j \\ 0 & \text{sinon} \end{cases}.$$

	plage 1 (00h à 12h)	plage 2 (06h à 18h)	plage 3 (12h à 24h)	plage 4 (18h à 06h)	plage 5 (00h à 06h)
Bloc 1 (0h à 6h)	1	0	0	1	1
Bloc 2 (6h à 12h)	1	1	0	0	0
Bloc 3 (12h à 18h)	0	1	1	0	0
Bloc 4 (18h à 24h)	0	0	1	1	0

Figure 2.12 Matrice X établissant le lien entre les blocs et les plages.

2.3.1 Une première tentative de répartition.

Le chef d'équipe propose la répartition suivante: 10 employés travailleront de 06h à 18h, et 6 employés travailleront de 18h à 06h.

- Cette répartition respecte-t-elle les contraintes en besoins minimaux?
- Quel coût entraîne-t-elle en salaire?

Puisqu'il y aura 10 employés de jour et 6 en soirée et nuit, les besoins minimaux seront comblés. Calculons le coût des salaires:

$$\text{plage 2:}\quad 10\ \text{employés}\cdot\frac{30\ \$/\text{h}}{\text{employé}}\cdot 12\text{h} \;=\; 3600\$ \qquad \text{plage 4:}\quad 6\ \text{employés}\cdot\frac{35\$/\text{h}}{\text{employé}}\cdot 12\text{h} \;=\; 2520\$.$$

$$\text{total}\quad 3600\$ + 2520\$ \;=\; 6120\$.$$

Ainsi, la répartition proposée par le chef d'équipe respecte les contraintes et et coûte 6120$ en salaire. Ces résultats sont illustrés à la figure 2.13.

	A	B	C	D	E	F	G	H	I	J
1	**But : minimiser la fonction-objectif, soit le salaire total des employés.**									
2										
3	**Fonction-objectif** : $z = \Sigma\ s_p\ n_p$ (salaire total)									
4	**Ensemble** des plages horaires P	1	2	3	4	5				
5	**Noms des variables** : n_p	n_1	n_2	n_3	n_4	n_5				
6	**Définition n_p** : nombre d'employés travaillant à la plage p									
7										
8	Salaire horaire de la plage p: h_p	34	30	32	35	31				
9	Durée d'un bloc d:	6								
10							z=			
11	**Coefficients s_p de la f-o et valeur de z**	408	360	384	420	186	**6120**			
12	*s_p : salaire pour plage p au complet pour 1 employé*									
13										
14	**Contraintes**	plage 1 (00h à 12h)	plage 2 (06h à 18h)	plage 3 (12h à 24h)	plage 4 (18h à 06h)	plage 5 (00h à 06h)	Effectif (f)		Besoin (b)	f>=b
15	Bloc 1 (0h à 6h)	1	0	0	1	1	6	>=	5	VRAI
16	Bloc 2 (6h à 12h)	1	1	0	0	0	10	>=	8	VRAI
17	Bloc 3 (12h à 18h)	0	1	1	0	0	10	>=	10	VRAI
18	Bloc 4 (18h à 24h)	0	0	1	1	0	6	>=	6	VRAI
19										
20	**Contraintes naturelles** : n_p>=0 et ENT									
21										
22	**Valeurs des variables n_p**	**0**	**10**	**0**	**6**	**0**				

Figure 2.13 Cette planification respecte toutes les contraintes et coûte 6120$ en salaire.

2.3.2 Une seconde tentative de répartition.

Le patron propose alors une seconde planification, qu'il prétend être moins coûteuse: 7 employés à la plage 2, 3 à la plage 3, 3 à l plage 4, 2 à la plage 5 et aucun à la plage 1.

- La proposition du patron respecte t-elle les contraintes en besoins minimaux?
- Quel coût entraîne-t-elle en salaire?

Le coût de cette proposition est de 5304 $, ce qui est effectivement moins cher que le 6120$ obtenu précédemment. Mais comme on le voit à la figure 2.13, la proposition du patron ne respecte pas toutes les contraintes : il y aura seulement 7 employés sur place entre 6h et 12h, alors qu'il en faut au moins 8. En effet, le bloc de 6h à 12h (bloc 2) est couvert par les employés des plages 1 et 2. Or il y a 0 employé sur la plage 1 et 7 sur la plage 2, pour un total de 7. Ces résultats sont illustrés à la figure 2.14.

	A	B	C	D	E	F	G	H	I	J
3	**Fonction-objectif** : $z = \Sigma\ s_p\ n_p$ (salaire total)									
4	**Ensemble** des plages horaires P	1	2	3	4	5				
5	**Noms des variables** : n_p	n_1	n_2	n_3	n_4	n_5				
6	**Définition n_p** : nombre d'employés travaillant à la plage p									
7										
8	Salaire horaire de la plage p: h_p	34	30	32	35	31				
9	Durée d'un bloc d :	6								
10							z=			
11	**Coefficients s_p** de la f-o et **valeur de z**	408	360	384	420	186	**5304**			
12	*s_p : salaire pour plage p au complet pour 1 employé*									
13										
14	**Contraintes**	plage 1 (00h à 12h)	plage 2 (06h à 18h)	plage 3 (12h à 24h)	plage 4 (18h à 06h)	plage 5 (00h à 06h)	Effectif (f)		Besoin (b)	f>=b
15	Bloc 1 (0h à 6h)	1	0	0	1	1	5	>=	5	VRAI
16	Bloc 2 (6h à 12h)	1	1	0	0	0	7	>=	8	FAUX
17	Bloc 3 (12h à 18h)	0	1	1	0	0	10	>=	10	VRAI
18	Bloc 4 (18h à 24h)	0	0	1	1	0	6	>=	6	VRAI
19										
20	**Contraintes naturelles** : n_p >=0 et ENT									
21										
22	**Valeurs des variables n_p**	**0**	**7**	**3**	**3**	**2**				

Figure 2.14 Cette planification ne respecte pas toutes les contraintes : il manque d'effectif entre 6h et 12h.

2.3.3 Un modèle mathématique avec coefficients numériques

Dans le but de trouver une planification qui entraîne le coût minimal en salaire, il faut construire un modèle mathématique.

Modèle avec coefficients numériques

But: minimiser le coût total des salaires, noté z et calculé en dollars. Il s'agit de la fonction-objectif. On l'obtient en multipliant, pour chacune des plages, le salaire horaire par le nombre d'heures et le nombre d'employés, puis en additionnant le tout.

Variables: nombre d'employés affectés à chacune des 5 plages

$$n_1, n_2, n_3, n_4, n_5.$$

Fonction-objectif: (en $)

$$z \;=\; (34\cdot 12)\cdot n_1 + (30\cdot 12)\cdot n_2 + (32\cdot 12)\cdot n_3 + (35\cdot 12)\cdot n_4 + (31\cdot \mathbf{6})\cdot n_5$$

(Remarquez que la 5e plage dure 6 heures.)

Contraintes: respecter les besoins minimaux en nombre d'employés pour chacun des 4 blocs.

$$\begin{array}{lrcl} \text{Bloc 1:} & n_1 + n_4 + n_5 & \geq & 5 \\ \text{Bloc 2:} & n_1 + n_2 & \geq & 8 \\ \text{Bloc 3:} & n_2 + n_3 & \geq & 10 \\ \text{Bloc 4:} & n_3 + n_4 & \geq & 6 \end{array}$$

Contraintes naturelles: (domaine des variables):

$$\forall\, i \in \{1,2,3,4,5\}, n_i \in \mathbb{N}.$$

2.3.4 Un modèle algébrique: adaptable et compact, mais grosse soupe à l'alphabet!

Si le salaire horaire des employés est modifié, ou si une nouvelle plage de travail est ouverte, comme de midi à 18h par exemple, alors il faudra recommencer le travail de modélisation. Pour plus de flexibilité, construisez un modèle algébrique. Nommez chacun des paramètres. Indiquez les contraintes et la fonction-objectif: on ne doit pas voir les coefficients numériques.

Remarque: voici un modèle très général. On aurait aussi pu fournir un modèle plus simple, où les paramètres s_p sont entrés directement au lieu d'être calculés à l'aide des autres paramètres.

Un modèle algébrique (avec paramètres).

But : **minimiser** le coût total des salaires, noté z et calculé en dollars.

Ensembles :

- $P = \{1,2,3,4,5\}$ Ensemble des plages de travail des employés.
- $Q = \{1,2,3,4\}$ Ensemble des blocs résultant du découpage d'une journée en périodes pour lesquelles les besoins en personnel sont constants.

Variables :

n_p : nombre d'employés affectés à la plage p, où $p \in P$.

Paramètres :

- d : durée en heures de la période de journée appelée « bloc »
  (dans notre exemple, un bloc dure 6 heures, mais on pourrait découper la journée en périodes plus courtes ou plus longues, de durée d)
- b_q : besoin minimal en employés durant le bloc $q \in Q$
- h_p : salaire horaire d'un employé travaillant à la plage $p \in P$
- s_p : salaire pour plage complète d'un employé travaillant à la plage $p \in P$ (s'obtient en multipliant le salaire horaire par le nombre d'heures de la plage, et ce dernier s'obtient en multipliant le nombre de blocs contenus dans la plage par la durée de chaque bloc).

$$s_p = h_p \cdot d \cdot \sum_{q \in Q} x_{qp}$$

- Description des plages avec la matrice X :

$$\forall q \in Q,\ \forall p \in P,\ x_{qp} = \begin{cases} 1 & \text{si la plage } p \text{ couvre le bloc } q \\ 0 & \text{sinon} \end{cases}.$$

Fonction-objectif :

$$z \ = \ \sum_{p \in P} \left(s_p \cdot n_p \right)$$

Contraintes : respecter les besoins minimaux pour chacun des blocs.

$$\forall q \in Q,\ \sum_{p \in P} \left(x_{qp} \cdot n_p \right) \ \geq \ b_q$$

Contraintes naturelles (domaine des variables) :

$$\forall p \in P,\ n_p \in \mathbb{N}.$$

2.3.5 Utilisation du solveur pour obtenir une solution optimale

L'utilisation du solveur pour le problème des horaires permet d'obtenir une solution qui satisfait toutes les contraintes et qui entraîne un cout minimal de 5514$. Pour obtenir ce coût minimal, il faut que 8 employés travaillent à la plage 2, 2 employés à la plage 3, 4 employés à la plage 4 et 1 employé à la plage 5.

	A	B	C	D	E	F	G	H	I	J
3	**Fonction-objectif** : $z = \Sigma\ s_p\ n_p$ (salaire total)									
4	**Ensemble** des plages horaires P	1	2	3	4	5				
5	**Noms des variables** : n_p	n_1	n_2	n_3	n_4	n_5				
6	**Définition n_p** : nombre d'employés travaillant à la plage p									
7										
8	Salaire horaire de la plage p: h_p	34	30	32	35	31				
9	Durée d'un bloc d:	6								
10							z=			
11	**Coefficients s_p** de la f-o et **valeur de z**	408	360	384	420	186	5514			
12	*s_p : salaire pour plage p au complet pour 1 employé*									
13										
14	**Contraintes**	plage 1 (00h à 12h)	plage 2 (06h à 18h)	plage 3 (12h à 24h)	plage 4 (18h à 06h)	plage 5 (00h à 06h)	Effectif (f)		Besoin (b)	f>=b
15	Bloc 1 (0h à 6h)	1	0	0	1	1	5	>=	5	VRAI
16	Bloc 2 (6h à 12h)	1	1	0	0	0	8	>=	8	VRAI
17	Bloc 3 (12h à 18h)	0	1	1	0	0	10	>=	10	VRAI
18	Bloc 4 (18h à 24h)	0	0	1	1	0	6	>=	6	VRAI
19										
20	**Contraintes naturelles** : n_p >=0 et ENT									
21										
22	**Valeurs des variables n_p**	0	8	2	4	1				

Figure 2.15 Le solveur a trouvé une solution qui satisfait toutes les contraintes et qui entraîne un cout minimal de 5514$.

2.4 Exemple 4: encore une planification d'un horaire pour minimiser le coût des salaires.

Toujours dans le contexte de la planification d'horaire, on considère maintenant le cas où les plages horaires et où les besoins en effectifs sont donnés au tableau 2.2.

Plage de travail	Salaire horaire en $
Plage 1 (00h à 8h)	29
Plage 2 (04h à 12h)	24
Plage 3 (8h à16h)	20
Plage 4 (8h à 12h)	35
Plage 5 (12h à 20h)	26
Plage 6 (16h à 00h)	28

Bloc	Besoin minimaux (nombre d'employés)
Bloc 1 (0h à 4h)	5
Bloc 2 (4h à 8h)	8
Bloc 3 (8h à 12h)	21
Bloc 4 (12h à 16h)	19
Bloc 5 (16h à 20h)	10
Bloc 6 (20h à 24h)	6

Tableau 2.2 Présentation des salaires horaires associés aux 6 plages horaires et des besoins minimaux en nombre d'employés à différentes périodes de la journée (blocs).

2.4.1 Avec le solveur

Créez une feuille de calcul afin d'utiliser le solveur de Excel pour minimiser les salaires pour les blocs et les plages donnés. Présentez une capture d'écran. Donnez la solution sous forme d'une phrase.

Une solution optimale est donnée à la figure 2.16. Pour obtenir le coût minimal de 6732$, il faut que 5 employés travaillent à la plage 1, 3 employés à la plage 2, 15 employés à la plage 3, 3 employés à la plage 4, 4 employés à la plage 5 et 6 employés à la plage 6.

But : minimiser la fonction-objectif, soit le salaire total des employés.

Fonction-objectif : $z = \Sigma\, s_p\, n_p$ (salaire total)

Ensemble des plages horaires P	1	2	3	4	5					
Noms des variables : n_p	n_1	n_2	n_3	n_4	n_5					
Définition n_p : nombre d'employés travaillant à la plage p										
Salaire horaire de la plage p: h_p	29	24	20	35	26	28				
Durée d'un bloc d :	4									
							z=			
Coefficients s_p de la f-o et valeur de z	232	192	160	140	208	224	6732			
s_p : salaire pour plage p au complet pour 1 employé										

Contraintes	plage 1 (00h à 8h)	plage 2 (04h à 12h)	plage 3 (8h à16h)	plage 4 (8h à 12h)	plage 5 (12h à 20h)	plage 6 (16h à 00h)	Effectif (f)		Besoin (b)	f>=b
Bloc 1 (0h à 4h)	1	0	0	0	0	0	5	>=	5	VRAI
Bloc 2 (4h à 8h)	1	1	0	0	0	0	8	>=	8	VRAI
Bloc 3 (8h à 12h)	0	1	1	1	0	0	21	>=	21	VRAI
Bloc 4 (12h à 16h)	0	0	1	0	1	0	19	>=	19	VRAI
Bloc 5 (16h à 20h)	0	0	0	0	1	1	10	>=	10	VRAI
Bloc 6 (20h à 24h)	0	0	0	0	0	1	6	>=	6	VRAI

Contraintes naturelles : n_p>=0 et ENT

Valeurs des variables n_p	5	3	15	3	4	6

Figure 2.16 Le solveur a trouvé une solution qui satisfait toutes les contraintes et qui entraîne un cout minimal de 6732$.

2.4.2 Sans le solveur

Trouvez une façon d'obtenir la solution optimale *sans utiliser le solveur*, en procédant à l'aide d'un raisonnement logique en quelques étapes. Expliquez votre raisonnement.

On remarque que le bloc 1 est couvert uniquement par les employés de la plage 1. Comme on a besoin d'un minimum de 5 employés durant le bloc 1, il faut minimalement affecter 5 personnes à la plage 1. Essayons donc avec 5.

$$\text{Besoins du bloc 1}: \quad n_1 \geq 5$$

$$\text{Essai}: \quad n_1 = 5.$$

De même, le bloc 6 est couvert uniquement par les employés de la plage 6. Comme on a besoin d'un minimum de 6 employés durant le bloc 6, il faut minimalement affecter 6 personnes à la plage 6. Essayons avec 6. (Beaucoup de 6 ici !)

$$\text{Besoins du bloc 6}: \quad n_6 \geq 6$$

$$\text{Essai}: \quad n_6 = 6.$$

Les employés de la plage 6 couvrent aussi le bloc 5. Or il faut minimalement 10 employés durant le bloc 5. Il faut donc du renfort aux 6 employés de la plage 6 : il faut minimalement affecter minimalement 4 employés à la plage 5. Essayons donc avec 4.

$$\text{Besoins du bloc 5}: \quad n_5 + n_6 \geq 10$$

$$\text{Ainsi} \quad n_5 \geq 10 - n_6 = 10 - 6 = 4.$$

$$\text{Essai}: \quad n_5 = 4.$$

En passant au bloc 4, où il faut au moins 19 employés, et qui est couvert par les plages 3 et 5, on déduit qu'il faut au moins 15 employés à la plage 3 pour compléter les 4 employés de la plage 5.

$$\text{Besoins du bloc 4 :} \quad n_3 + n_5 \geq 19$$

$$\text{Ainsi } n_3 \geq 19 - n_5 = 19 - 4 = 15.$$

$$\text{Essai :} \quad n_3 = 15.$$

En passant au bloc 2, où il faut au moins 8 employés, et qui est couvert par les plages 1 et 2, on déduit qu'il faut au moins 3 employés à la plage 2 pour compléter les 5 employés de la plage 1.

$$\text{Besoins du bloc 2 :} \quad n_1 + n_2 \geq 8$$

$$\text{Ainsi } n_2 \geq 8 - n_1 = 8 - 5 = 3.$$

$$\text{Essai :} \quad n_2 = 3.$$

En passant au bloc 3, où il faut au moins 21 employés, et qui est couvert par les plages 2, 3 et 4, on déduit qu'il faut au moins 3 employés à la plage 4 pour compléter les 3 employés de la plage 2 et les 15 employés de la plage 3.

$$\text{Besoins du bloc 3 :} \quad n_2 + n_3 + n_4 \geq 21$$

$$\text{Ainsi } n_4 \geq 21 - n_2 - n_3 = 21 - 3 - 15 = 3.$$

$$\text{Essai :} \quad n_4 = 3.$$

Comme on a pris le nombre minimal d'employés à chacune des étapes et que toutes les contraintes sont satisfaites, on est certain d'avoir trouvé une solution optimale.

$$n_1 = 5, n_2 = 3, n_3 = 15, n_4 = 3, n_5 = 4, n_6 = 6.$$

Exercices

2.1 Selon le premier modèle de l'entreprise 3P présenté à la section 2.1.1 et rappelé ci-dessous, déterminez, pour chacune des affectations suivantes, si :

- c'est une solution admissible optimale
- c'est une solution admissible mais pas optimale
- ce n'est pas une solution admissible.

Un premier modèle pour l'entreprise 3P

But : maximiser la contribution marginale totale en $, que l'on note z.

Variables : $\forall i \in \{1,2,3\}$, x_i = nombre d'unités à fabriquer du produit i.

Contraintes naturelles (domaine des variables) : on ne peut pas avoir un nombre d'unités négatif ou fractionnaire. Ainsi,

$$\forall i \in \{1,2,3\},\ x_i \in \mathbb{N}.$$

Fonction-objectif : c'est ainsi que l'on appelle la fonction à optimiser

$$z = 120x_1 + 150x_2 + 90x_3.$$

Contraintes : pour chacun des ateliers, le temps d'utilisation requis doit être inférieur ou égal au temps disponible.

$$\begin{array}{lrcrcrcr} \text{Atelier 1}: & 2x_1 & + & 1x_2 & + & 1x_3 & \leq & 400 \\ \text{Atelier 2}: & 4x_1 & + & 5x_2 & + & 3x_3 & \leq & 1400 \\ \text{Atelier 3}: & 1x_1 & + & 2x_2 & + & 1x_3 & \leq & 500 \end{array}$$

(a) $x_1 = 100, x_2 = 125, x_3 = 125$.

(b) $x_1 = 0, x_2 = 350, x_3 = 0$.

(c) $x_1 = 0, x_2 = 50, x_3 = 300$.

(d) $x_1 = 50, x_2 = 150, x_3 = 150$.

(e) $x_1 = 100.5, x_2 = 99.5, x_3 = 50$.

2.2 Adaptez le premier modèle de l'entreprise 3P présenté à la page 66 afin de tenir compte des 3 contraintes supplémentaires suivantes. Écrivez les contraintes sous la forme suivante :

$$(\text{coefficient 1}) \cdot x_1 + (\text{coefficient 2}) \cdot x_2 + (\text{coefficient 3}) \cdot x_3 \quad [\text{signe} \leq, =, \geq] \quad \text{constante}.$$

1. L'entreprise doit produire au moins 150 unités du produit 3 pour respecter son carnet de commandes.
2. L'entreprise doit produire au moins 250 unités des produits 1 et 3 combinés pour respecter son carnet de commandes.
3. L'entreprise doit produire au moins autant d'unités du produit 1 que du produit 2.

Utilisez un solveur pour obtenir une solution optimale.

La figure 2.17 montre comment ajouter des contraintes dans les paramètres du solveur. Il faut préalablement insérer les lignes et formules nécessaires dans la feuille de calcul.

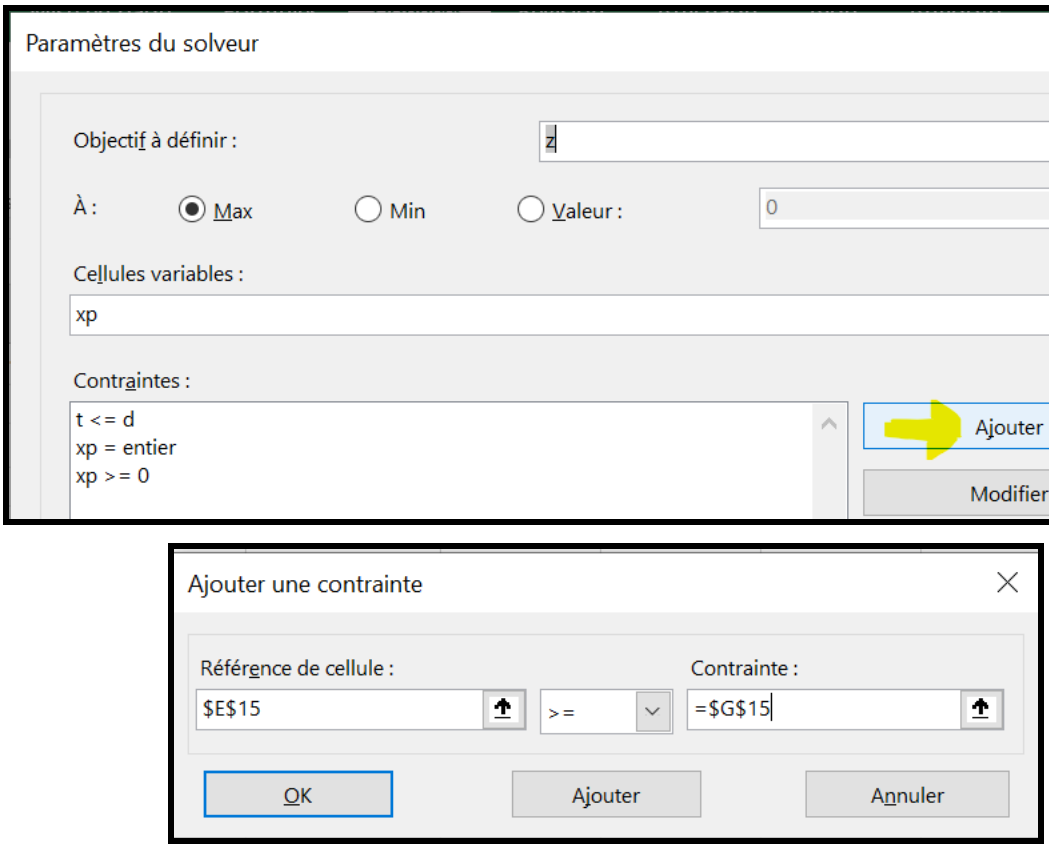


Figure 2.17 Ajouter une contrainte dans les paramètres du solveur.

Pour chacun des exercices 2.3 à 2.8, indépendants les uns des autres, adaptez le modèle algébrique de la Section 2.2.2 de la page 75 en y apportant les modifications nécessaires : **But, Ensembles**, **Variables**, **Contraintes naturelles**, **Paramètres**, **Fonction-objectif**, **Contraintes**.

2.3 L'organisme public doit maintenant répondre à la contrainte suivante : *les citoyens de chaque quartier doivent avoir accès à* ***au moins 3*** *points de service situés soit dans leur propre quartier, soit dans un quartier adjacent.* Dans quels quartiers doit-on déployer des points de service si l'on souhaite en minimiser le nombre total tout en respectant la contrainte ? Quel sera alors le nombre total de points de service ?

2.4 L'organisme public doit maintenant répondre à la contrainte supplémentaire suivante : ***le quartier 5 est mis en quarantaine****; ses citoyens doivent rester dans leur quartier et personne d'autre ne peut y entrer.* Dans quels quartiers doit-on déployer des points de service si l'on souhaite en minimiser le nombre total tout en respectant la contrainte de proximité (au moins un point de service adjacent à leur quartier) et la quarantaine ? Quel sera alors le nombre total de points de service ?

2.5 L'organisme public doit maintenant répondre à la contrainte suivante qui tient compte de la **densité de la population** : *les citoyens des quartiers centraux (4, 5 et 6) doivent avoir accès à au moins 3 points de service situés soit dans leur propre quartier, soit dans un quartier adjacent ; les citoyens des autres quartiers doivent avoir accès à au moins 1 point de service situé soit dans leur propre quartier, soit dans un quartier adjacent.* Dans quels quartiers doit-on implanter des points de service si l'on souhaite en minimiser le nombre total tout en respectant la contrainte ? Quel sera alors le nombre total de points de service ?

Indice : vous devez ajouter un paramètre au modèle.

2.6 L'organisme public doit maintenant répondre à la contrainte suivante : *conserver la contrainte initiale (besoin de 1 point de service adjacent), avec un* ***coût différent dans chacun des quartiers*** *pour le déploiement d'un point de service.* Le coût de location est 60% plus élevé dans les quartiers 2, 3, 4, 5 et 8 que dans les autres quartiers. On cherche maintenant à minimiser le coût de location total : dans quels quartiers devrait-on déployer les points de service ?

2.7 Une compagnie veut installer des **franchises** dans la ville dont la carte est donnée à la figure 2.7 de la page 73. La compagnie garantie à ses franchisés que la contrainte suivante sera respectée : *les citoyens de chaque quartier auront accès à* ***au plus 2*** *franchises situées soit dans leur propre quartier, soit dans un quartier adjacent.* Notez qu'il pourrait y avoir plusieurs franchises dans le même quartier. Quel est le nombre maximal de franchises qui peuvent être déployées dans cette ville tout en respectant la contrainte ? Pour atteindre ce nombre, où devrait-on déployer les franchises ?

2.8 Un compagnie veut installer des **franchises** dans la ville dont la carte est donnée à la figure 2.7 de la page 73. La compagnie garantie à ses franchisés que la contrainte suivante sera respectée : *les citoyens de chaque quartier auront accès à* ***3 franchises ou moins*** *situées soit dans leur propre quartier, soit dans un quartier adjacent.* Notez qu'il pourrait y avoir plusieurs franchises dans le même quartier. Quel est le nombre maximal de franchises qui peuvent être déployées dans cette ville tout en respectant la contrainte ? Pour atteindre ce nombre, où devrait-on déployer les franchises ?

Pour le prochain exercice, vous devez **créer un modèle** et non adapter un exemple.

2.9 Une entreprise doit recruter une petite équipe pour réaliser un projet à l'extérieur du pays. Suite aux entrevues de 8 postulants, à l'analyse des compétences et des besoins, ainsi qu'aux questions de bonne entente, les contraintes suivantes ont été identifiées.

1. Il faut embaucher exactement quatre personnes.
2. Au moins un des postulants numéro 3, 5 ou 8 doit être embauché, car eux seuls ont des connaissances en électricité.
3. Si le postulant 2 est embauché ou le postulant 4 est embauché, alors le postulant 5 ne doit pas l'être.
4. Si les postulants 1 et 3 sont embauchés tous les deux, alors le postulant 6 ne doit pas l'être.
5. Les postulants 3 et 5 forment un couple : ils accepteront l'affectation seulement si les deux sont embauchés.

Selon la convention collective, le salaire annuel varie en fonction du nombre d'années d'expérience. Le tableau suivant indique le salaire pour chacun des postulants.

Postulant	1	2	3	4	5	6	7	8
Salaire annuel en 1000$	65	55	58	74	55	74	70	81

Est-ce possible de choisir les quatre personnes tout en respectant toutes ces contraintes ? Si oui, quels postulants l'entreprise devrait embaucher si elle souhaite minimiser le salaire annuel total de l'équipe ? Quel sera alors le salaire total ?

Attention de bien décrire le modèle et non de sauter trop vite à la feuille de calcul!

Chapitre 3

Théorie des graphes

3.1 Terminologie et types de graphes

Définition 3.1 : Graphe non orienté

Un **graphe non orienté** $G = (V, E)$ est un couple d'ensembles, où V est un ensemble de **sommets** et E un ensemble d'**arêtes**. Chaque arête est associée à un ensemble d'un ou deux sommets.

Comme l'illustre la figure 3.1, les arêtes représentent les liens qui existent entre les sommets du graphe. Lorsque plusieurs arêtes sont associées au même ensemble de sommets, on les appelle des **arêtes multiples**. Une arête qui relie un sommet à lui-même est appelée une **boucle**.

Deux sommets $u, v \in V$ sont **adjacents** s'ils sont joints par une arête $e \in E$. On dit alors que e est une **arête incidente** aux sommets u et v. Le **degré** d'un sommet v, noté $\deg(v)$, désigne le nombre d'arêtes qui lui sont incidentes.

Note : Une boucle compte pour deux dans le degré d'un sommet.

Notation : Lorsqu'il n'y a qu'une seule arête reliant une paire de sommets u et v, on la dénote parfois par $\{u, v\}$ ou encore plus simplement par $u-v$.

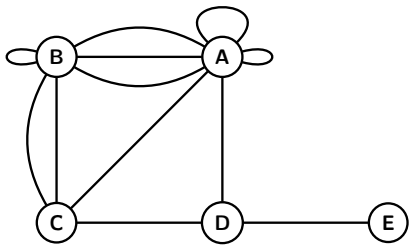


Figure 3.1 Exemple de graphe non orienté comportant 5 sommets et 12 arêtes, dont 3 boucles.

Exemple 3.1

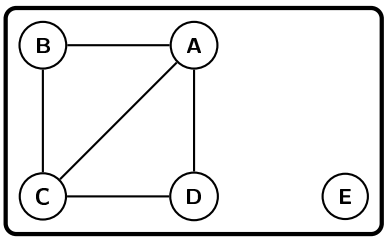
Dessinez le graphe non orienté $G = (V, E)$, où

$$V = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}\},$$
$$E = \big\{\{\mathsf{A}, \mathsf{B}\}, \{\mathsf{A}, \mathsf{C}\}, \{\mathsf{A}, \mathsf{D}\}, \{\mathsf{B}, \mathsf{C}\}, \{\mathsf{C}, \mathsf{D}\}\big\}.$$

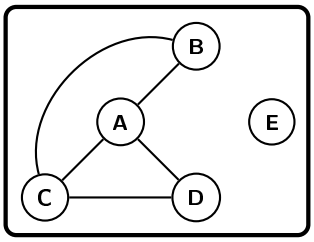
De plus, déterminez le degré de chacun des sommets.

Solution :

Il y a plusieurs façons de dessiner un graphe. En effet, on peut illustrer le graphe décrit plus haut comme



ou encore



Dans les deux cas, on voit que les arêtes respectives entre les sommets sont préservées ainsi que les degrés des sommets :

$$\deg(\mathsf{A}) = 3, \quad \deg(\mathsf{B}) = 2, \quad \deg(\mathsf{C}) = 3, \quad \deg(\mathsf{D}) = 2, \quad \deg(\mathsf{E}) = 0.$$

Théorème 3.1 : des poignées de mains

Soit $G = (V, E)$ un graphe non orienté. Alors

$$\sum_{v \in V} \deg(v) = 2|E|.$$

Exemple 3.2

Pour illustrer le théorème des poignées de mains, considérons le graphe non orienté de l'exemple 3.1 qui contient 5 arêtes :

$$\begin{aligned}\sum_{v \in V} \deg(v) &= \deg(\mathsf{A}) + \deg(\mathsf{B}) + \deg(\mathsf{C}) + \deg(\mathsf{D}) + \deg(\mathsf{E}) \\ &= 3 + 2 + 3 + 2 + 0 \\ &= 10 \\ &= 2|E|.\end{aligned}$$

Définition 3.2 : Graphe orienté

Un **graphe orienté** $G = (V, E)$ est un couple d'ensembles, où V est un ensemble de **sommets** et E un ensemble d'**arcs**. Chaque arc est associé à une *paire ordonnée* de sommets.

Si plusieurs arcs relient la même paire ordonnée de sommets, on les appelle des **arcs multiples**. Un arc qui relie un sommet à lui-même est appelé une **boucle**.

Soit $e \in E$ un arc associé à la paire de sommets (u, v). Le sommet u est appelé **sommet initial** et le sommet v **sommet terminal** de e. On dit que le sommet v est **adjacent** au sommet u. Le **degré sortant** d'un sommet v, noté $\deg^+(v)$, désigne le nombre d'arcs dont v est le sommet initial. Le **degré entrant** d'un sommet v, noté $\deg^-(v)$, désigne le nombre d'arcs dont v est le sommet terminal.

Note : Une boucle contribue pour 1 degré entrant et 1 degré sortant.

Notation : Lorsqu'il n'y a qu'un seul arc du sommet u au sommet v, on le dénote parfois par (u, v) ou encore plus simplement par $u \to v$.

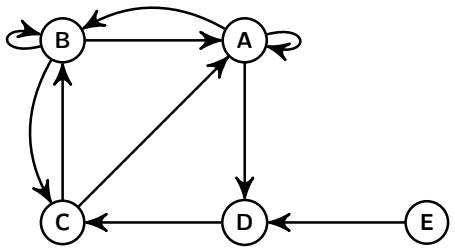
Exemple 3.3

Dessinez le graphe orienté $G = (V, E)$, où

$$V = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}\},$$
$$E = \{(\mathsf{A},\mathsf{A}), (\mathsf{A},\mathsf{B}), (\mathsf{A},\mathsf{D}), (\mathsf{B},\mathsf{B}), (\mathsf{B},\mathsf{A}), (\mathsf{B},\mathsf{C}), (\mathsf{C},\mathsf{A}), (\mathsf{C},\mathsf{B}), (\mathsf{D},\mathsf{C}), (\mathsf{E},\mathsf{D})\}.$$

De plus, déterminez les degrés entrants et sortants de chacun des sommets.

Solution :



$\deg^+(\mathsf{A}) = 3$ $\quad \deg^-(\mathsf{A}) = 3$
$\deg^+(\mathsf{B}) = 3$ $\quad \deg^-(\mathsf{B}) = 3$
$\deg^+(\mathsf{C}) = 2$ $\quad \deg^-(\mathsf{C}) = 2$
$\deg^+(\mathsf{D}) = 1$ $\quad \deg^-(\mathsf{D}) = 2$
$\deg^+(\mathsf{E}) = 1$ $\quad \deg^-(\mathsf{E}) = 0$

Théorème 3.2 : des coups de pieds

Soit $G = (V, E)$ un graphe orienté. Alors

$$\sum_{v \in V} \deg^+(v) = \sum_{v \in V} \deg^-(v) = |E|$$

Exemple 3.4

Pour illustrer le théorème des coups de pied, considérons le graphe orienté de l'exemple 3.3 qui contient 10 arcs :

$$\begin{aligned}\sum_{v\in V} \deg^+(v) &= \deg^+(\mathsf{A}) + \deg^+(\mathsf{B}) + \deg^+(\mathsf{C}) + \deg^+(\mathsf{D}) + \deg^+(\mathsf{E}) \\ &= 3+3+2+1+1 \\ &= 10 \\ &= |E|\end{aligned}$$

$$\begin{aligned}\sum_{v\in V} \deg^-(v) &= \deg^-(\mathsf{A}) + \deg^-(\mathsf{B}) + \deg^-(\mathsf{C}) + \deg^-(\mathsf{D}) + \deg^-(\mathsf{E}) \\ &= 3+3+2+2+0 \\ &= 10 \\ &= |E|.\end{aligned}$$

Dans certains théorèmes, définitions et exemples, nous utilisons le terme **graphe** pour désigner un graphe orienté ou non. Nous utilisons aussi parfois le terme graphe pour parler de graphe non orienté seulement, quand le contexte ne porte pas à confusion.

Définition 3.3 : Graphe simple

Un graphe qui ne contient aucune boucle ni arête (ou arc) multiple est qualifié de **simple**.

Un graphe peut donc être orienté ou non, et simple ou non. Le tableau 3.1 présente chacun des 4 types de graphes avec leurs caractéristiques et un exemple. Attention, certains auteurs utilisent le terme *graphe* pour désigner un *graphe simple*: il faut être vigilant avec les définitions quand on passe d'un livre à un autre.

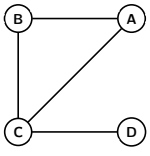
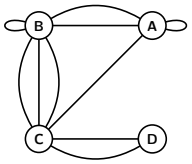
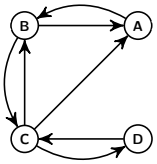
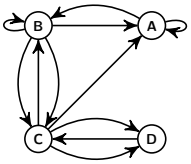
Graphe simple non orienté	**Graphe non orienté**	**Graphe simple orienté**	**Graphe orienté**
			
Arêtes multiples et boucles	**Arêtes multiples et boucles**	**Arcs multiples et boucles**	**Arcs multiples et boucles**
non permises	permises	non permis	permis

Tableau 3.1 Types de graphes

Plusieurs situations de la vie courante peuvent être modélisées par des graphes. Par exemple, le graphe de la figure 3.2 représente le trafic aérien mondial, où les sommets correspondent aux villes et les arêtes aux liens aériens. Le graphe de la figure 3.3 est un graphe partiel d'Internet, où les sommets correspondent à des adresses IP et les arêtes aux liens entre elles.

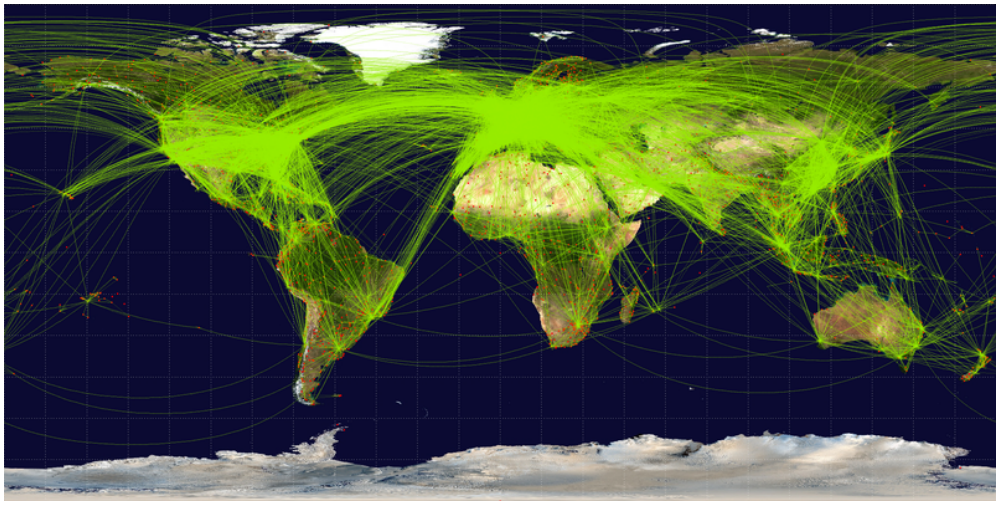


Figure 3.2 Carte du trafic aérien. Jpatokal (2009)

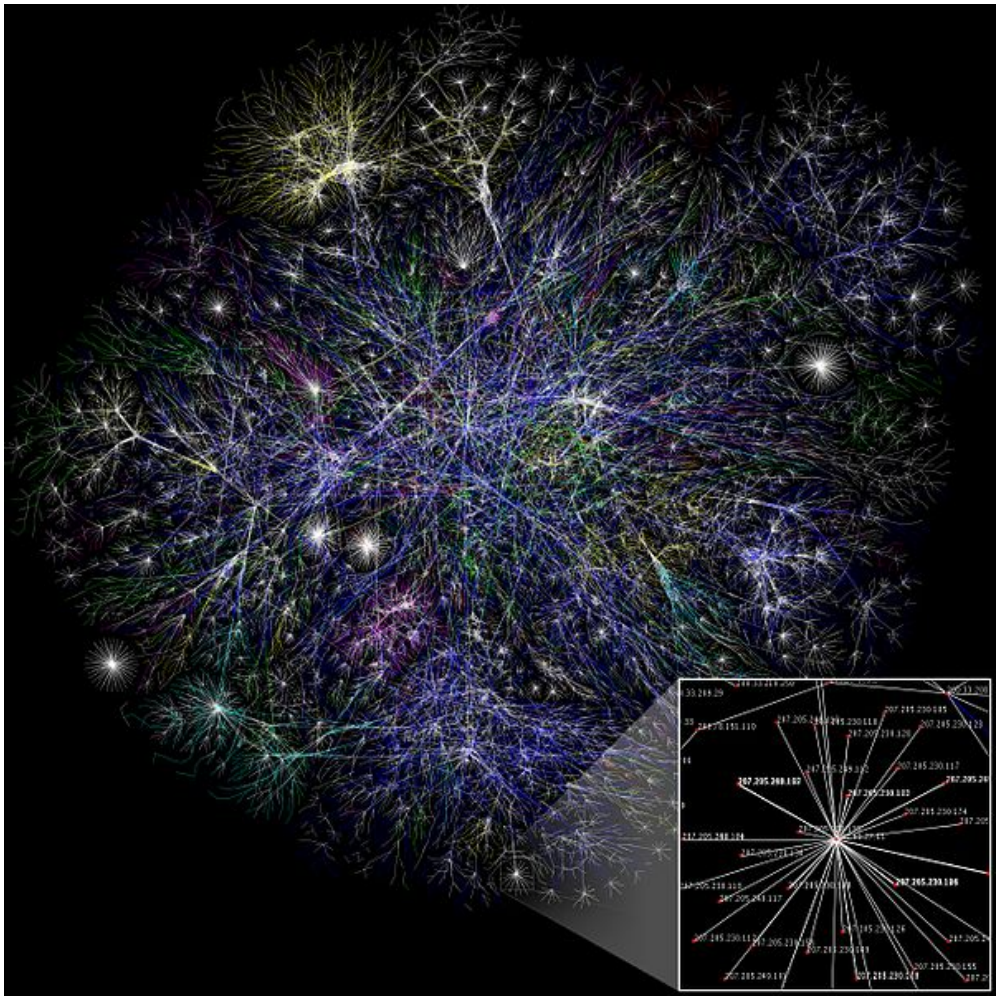


Figure 3.3 Graphe partiel d'Internet. The Opte Project (2006).

Exercices

3.1 Prouvez qu'un graphe non orienté possède toujours un nombre pair de sommets de degré impair.

3.2 Un graphe simple non orienté est dit **complet** si tous ses sommets sont adjacents. Quel est le nombre total d'arêtes dans le graphe complet

(a) K_5 à 5 sommets?

(b) K_n à n sommets?

Indice: il existe plusieurs façons de résoudre ce problème. Par exemple, on peut utiliser le théorème 3.1 ou encore, le théorème 5.7.

3.2 Représentation des graphes

Pour qu'un ordinateur puisse manipuler un graphe $G = (V, E)$, il faut une structure de données pour représenter les ensembles V et E. Dans cette section, nous donnons deux manières de représenter un graphe, soit par une liste d'adjacence ou une matrice d'adjacence.

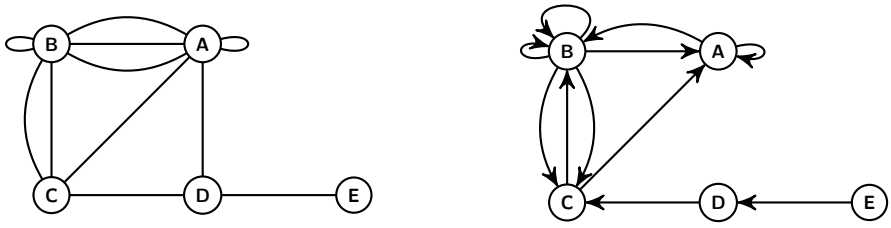
3.2.1 Représentation par listes d'adjacence

Un graphe est représenté par un tableau associatif qui, à chaque sommet, associe la liste des sommets auxquels il est adjacent. Autrement dit, la liste associée au sommet u contient le sommet v si et seulement s'il existe une arête (ou arc) de u à v.

Cette méthode est avantageuse dans le cas où le graphe compte un grand nombre n de sommets et où le nombre d'arêtes (ou d'arcs) est significativement plus petit que n^2.

Exemple 3.5

Représentez par une liste d'adjacence chacun des graphes suivants.



Solution :

sommet	liste d'adjacence
A	A,B,B,B,C,D
B	A,A,A,B,C,C
C	A,B,B,D
D	A,C,E
E	D

sommet initial	liste d'adjacence
A	A,B
B	A,B,B,C,C
C	A,B
D	C
E	D

3.2.2 Représentation par matrice d'adjacence

Un graphe est représenté par une matrice dont la case en ligne i, colonne j indique le nombre d'arêtes (ou d'arcs) allant du i^e sommet au j^e sommet.

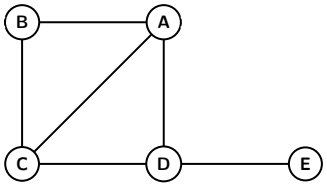
Définition 3.4 : Matrice d'adjacence

Soit G un graphe de n sommets $\{v_1, v_2, \dots v_n\}$. La **matrice d'adjacence** est la matrice carrée $M_{n\times n} = [m_{ij}]$ de dimension $n \times n$, où l'entrée

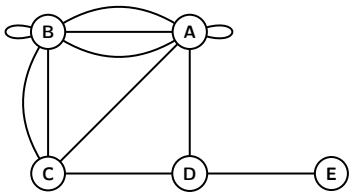
$$m_{ij} = \text{ nombre d'arêtes (ou d'arcs) du sommet } v_i \text{ au sommet } v_j.$$

Exemple 3.6

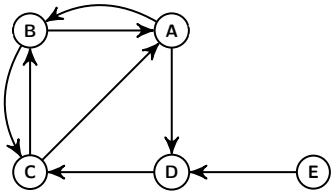
Considérez les quatre graphes suivants et leur matrice d'adjacence (en prenant l'ordre alphabétique sur les sommets). Pour chacun des graphes, déterminez s'il est simple ou non, orienté ou non.

(a) 

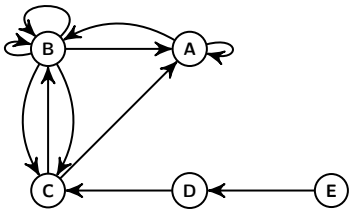
$$M = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

(b) 

$$M = \begin{pmatrix} 1 & 3 & 1 & 1 & 0 \\ 3 & 1 & 2 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

(c) 

$$M = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

(d) 

$$M = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 2 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Solution :

(a) Graphe simple non orienté. Dans un graphe non orienté, on remarque que la matrice d'adjacence est symétrique par rapport à la diagonale. De plus, dans un graphe simple, la diagonale ne contient que des 0.

(b) Graphe non orienté.

(c) Graphe simple orienté.

(d) Graphe orienté.

3.3 Chemins dans un graphe

3.3.1 Chemins, circuits, cycles

Définition 3.5 : Chemin, circuit, cycle, simple ou élémentaire

Soit $G = (V, E)$ un graphe orienté ou non. Un **chemin** de longueur n du sommet u au sommet v est une suite de n arêtes (arcs)

$$e_1, e_2, \ldots, e_n$$

telle qu'il existe une suite de sommets

$$\mathbf{u} = \mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n = \mathbf{v}$$

où, pour i allant de 1 à n, l'arête $e_i = \{\mathbf{v}_{i-1}, \mathbf{v}_i\}$ (ou l'arc $e_i = (\mathbf{v}_{i-1}, \mathbf{v}_i)$).

- S'il n'y a pas d'arêtes (arcs) multiples, on peut désigner ce chemin par $\mathbf{v}_0 - \mathbf{v}_1 - \ldots - \mathbf{v}_n$.
- Un **circuit**, aussi appelé **cycle**, est un chemin avec $\mathbf{v}_0 = \mathbf{v}_n$, pour $n > 0$. (On réserve parfois le terme *circuit* aux graphes orientés et le terme cycle aux graphes non orientés, mais nous les considérerons ici comme synonymes.)
- Un chemin, circuit ou cycle est dit **simple** s'il ne contient pas une même arête (arc) plus d'une fois.
- Un circuit ou cycle est dit **élémentaire** s'il ne contient pas un même sommet plus d'une fois (sauf le premier et le dernier). Un cycle élémentaire ne contient pas d'autre cycle.

Dans le graphe de la figure 3.4, la suite d'arêtes e_1, e_2, e_3, e_4, e_5 est un chemin allant du sommet u vers le sommet v. Comme le graphe est simple, on peut aussi désigner ce chemin par $\mathbf{v}_0 - \mathbf{v}_1 - \mathbf{v}_2 - \mathbf{v}_3 - \mathbf{v}_4 - \mathbf{v}_5$. La suite d'arêtes e_2, e_6, e_7, e_8 est un circuit simple et élémentaire. La suite d'arêtes e_1, e_2, e_2 (ou encore $\mathbf{v}_0 - \mathbf{v}_1 - \mathbf{v}_2 - \mathbf{v}_1$) est un chemin qui n'est pas simple, comme l'arête e_2 se répète.

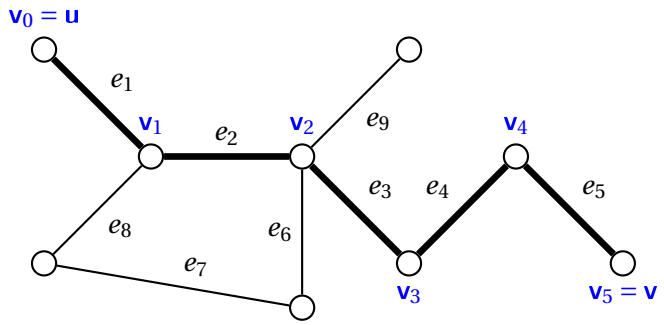
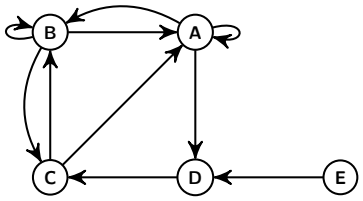


Figure 3.4 La suite d'arêtes e_1, e_2, e_3, e_4, e_5 est un chemin allant du sommet $\mathbf{u}$ vers le sommet $\mathbf{v}$.

Exemple 3.7

Soit le graphe orienté suivant.



Déterminez si les chemins suivants sont des circuits simples ou élémentaires. De plus, donnez leurs longueurs.

(a) A−B−A−B−C−B−A

(b) B−C−A−D−C−B

(c) D−C−B−A−D

Solution :

(a) Circuit de longueur 6 qui n'est ni simple, ni élémentaire.

(b) Circuit simple, mais pas élémentaire car il passe 2 fois par le sommet C, de longueur 5.

(c) Circuit simple et élémentaire de longueur 4.

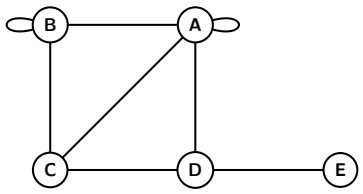
3.3.2 Dénombrement de chemins

Théorème 3.3

Soit G un graphe et M sa matrice d'adjacence (en respectant l'ordre $v_1, v_2, \ldots, v_n$ des sommets de G). Alors le nombre de chemins différents de longueur $k \in \mathbb{N}^*$ allant du sommet v_i au sommet v_j est donné par l'entrée (i, j) de la matrice M^k.

Exemple 3.8

Déterminez le nombre de chemins de longueur 4 allant du sommet **E** au sommet **B** dans le graphe suivant et dressez une liste de ces chemins.



Solution :

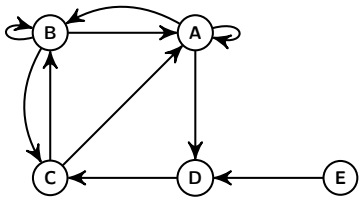
$$M = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \qquad M^4 = \begin{pmatrix} 39 & 31 & 30 & 23 & 8 \\ 31 & 26 & 23 & 20 & 5 \\ 30 & 23 & 24 & 16 & 7 \\ 23 & 20 & 16 & 18 & 3 \\ 8 & 5 & 7 & 3 & 3 \end{pmatrix}$$

Chemins de longueur 4 allant du sommet **E** au sommet **B** :

E–**D**–**A**–**A**–**B**
E–**D**–**A**–**B**–**B**
E–**D**–**A**–**C**–**B**
E–**D**–**C**–**A**–**B**
E–**D**–**C**–**B**–**B**

Exemple 3.9

Déterminez le nombre de chemins de longueur 3 allant du sommet **C** au sommet **A** dans le graphe suivant et dressez une liste de ces chemins.



Solution :

$$M = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \qquad M^3 = \begin{pmatrix} 6 & 6 & 3 & 2 & 0 \\ 7 & 7 & 4 & 3 & 0 \\ 5 & 5 & 3 & 2 & 0 \\ 2 & 2 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Chemins de longueur 3 allant du sommet **C** au sommet **A** :

C–**A**–**A**–**A**
C–**A**–**B**–**A**
C–**B**–**A**–**A**
C–**B**–**B**–**A**
C–**B**–**C**–**A**

Exercices

3.3 Les randonnées pédestres d'un parc national reliant différents sites (A, B, C, ...) à visiter sont représentées par un graphe. Les sommets correspondent aux différents sites et les arêtes aux randonnées d'une heure entre deux sites. La matrice d'adjacence du graphe, construite selon l'ordre alphabétique des sommets, est :

$$M = \begin{pmatrix} 0 & \mathbf{1} & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Ainsi, l'entrée $\mathbf{m_{12}} = \mathbf{1}$ de la matrice signifie qu'il y a une randonnée d'une heure entre le site A et le site B. Répondez aux questions suivantes en utilisant seulement des opérations matricielle sur M.

(a) Quel est le nombre de randonnées de 5 heures partant du site B ?

(b) Quel est le nombre de randonnées de 4 heures ou moins qui se terminent au site C ?

(c) Quel est le nombre de randonnées de 4 heures dont le site de départ est le même que le site d'arrivée ?

(d) Quelle est la durée minimale (en heures) d'une randonnée entre le site A et le site K ?

3.3.3 Chemins et circuits eulériens

Le problème des **sept ponts de Königsberg** consiste à déterminer si l'on peut prendre une marche dans les rues de la ville (voir figure 3.5) en passant une et une seule fois par chaque pont, en partant d'un point donné et en revenant à ce même point.

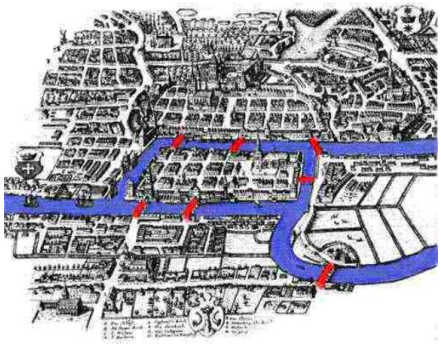


Figure 3.5 Ville de Königsberg.

Le mathématicien Euler a donné une solution à ce problème, qui est considérée comme le fondement même de la théorie des graphes. Dans sa version plus moderne, la solution repose sur l'existence de circuit eulérien dans un graphe. Nous présenterons donc la solution à ce problème après avoir introduit certaines notions importantes.

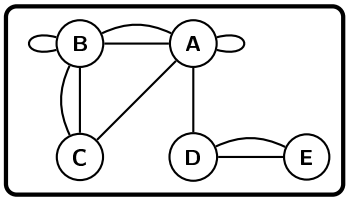
Définition 3.6 : Chemin et circuit eulériens

- Un **chemin eulérien** est un chemin simple qui contient toutes les arêtes.
- Un **circuit eulérien** est un circuit simple qui contient toutes les arêtes.

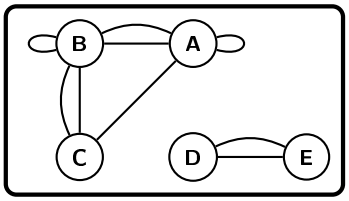
Définition 3.7 : Graphe connexe

Un graphe non orienté est **connexe** si pour toute paire de sommets (x, y), il existe un chemin allant de x à y.

Exemple 3.10



graphe connexe



graphe non connexe

Théorème 3.4

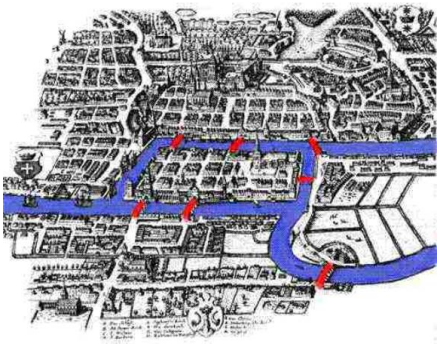
Un graphe connexe non orienté possède un circuit eulérien si et seulement si tous ses sommets sont de degré pair.

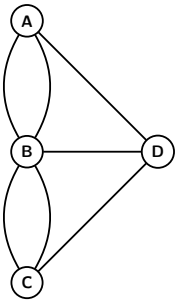
Théorème 3.5

Un graphe connexe non orienté possède un chemin eulérien, mais pas de circuit eulérien, si et seulement s'il possède exactement 2 sommets de degré impair.

Exemple 3.11

Nous pouvons maintenant revenir au problème des **sept ponts de Königsberg**. Est-il possible de prendre une marche dans les rues de la ville en passant une et une seule fois par chaque pont, en partant d'un point donné et en revenant à ce même point? La ville et ses ponts peuvent être modélisés par un graphe, où les ponts correspondent aux arêtes et les sommets aux différentes parties de la ville séparées par le cours d'eau:





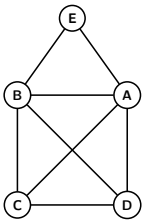
Le problème peut donc être reformulé comme suit: « Existe-t-il un circuit eulérien dans le graphe ci-dessus? »

Solution :

Le Théorème 3.4 nous permet de conclure que ce graphe ne possède aucun circuit eulérien, car il possède des sommets de degré impair. Il est donc impossible de trouver un itinéraire qui parcourt chaque pont de Königsberg une et une seule fois en revenant à son point de départ.

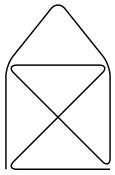
Exemple 3.12

Peut-on tracer cette maison sans lever le crayon et sans repasser sur un segment plus d'une fois?



Solution :

Comme ce graphe non orienté possède exactement 2 sommets de degré impair, les sommets C et D, le théorème 3.5 permet de conclure que ce graphe possède un chemin eulérien. En voici un: C−B−E−A−D−B−A−C−D. On peut donc tracer la maison sans lever le crayon et sans repasser sur un segment.



Algorithme de Hierholzer

L'**algorithme de Hierholzer** permet de construire un circuit eulérien dans un graphe non orienté, s'il existe. Le fait que cet algorithme est correct constitue une démonstration du Théorème 3.4. Étant donné un sommet v, on effectue le travail suivant :

À partir de v, avancer tant qu'il est possible de le faire, en supprimant toutes les arêtes empruntées.

Il est important de remarquer que si tous les sommets du graphe sont de degré pair, alors un tel chemin termine forcément à son point de départ. La fonction auxiliaire `chemin` formalise cette idée.

Algorithme 1 `chemin`

Entrées: un graphe non orienté G et v un sommet de G
Sortie: une liste de sommets décrivant un chemin dans G

```
1: C := liste vide
2: Ajouter v à C
3: tant que deg(v) > 0 faire
4:     Choisir une arête e incidente à v et appeler w le sommet à son autre extrémité
5:     Ajouter w à la fin de C
6:     Retirer l'arête e du graphe G
7:     v := w
8: fin tant que
9: retourner C
```

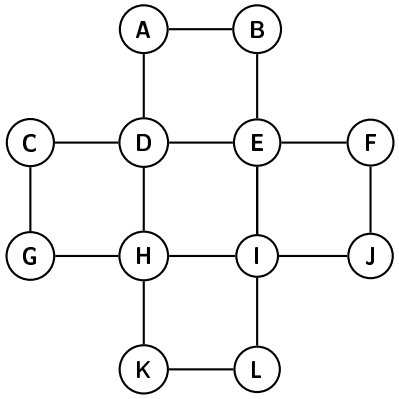
Algorithme 2 `Hierholzer`

Entrées: un graphe non orienté connexe G dont tous les sommets sont de degré pair
Sortie: un circuit eulérien de G

```
1: circuit := liste vide
2: Ajouter un sommet de G à circuit
3: i := 0
4: tant que G possède au moins une arête faire
5:     v := circuit[i]
6:     C := chemin(G, v)
7:     Insérer la liste C dans circuit, à la place de circuit[i]
8:     i := i + 1
9: fin tant que
10: retourner circuit
```

Exemple 3.13

Exécuter l'algorithme de Hierholzer sur le graphe suivant en utilisant la convention que lorsque qu'il faut choisir parmi plusieurs sommets, on utilise toujours le premier en ordre alphabétique.

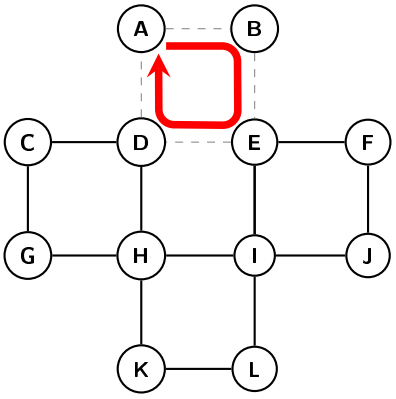


Solution :

À la ligne 2 de l'algorithme de Hierholzer, on choisit le sommet **A** comme sommet de départ puisqu'il s'agit du plus petit en ordre alphabétique. Ainsi, avant d'entrer dans la boucle **tant que**, la liste $circuit$ est [A].

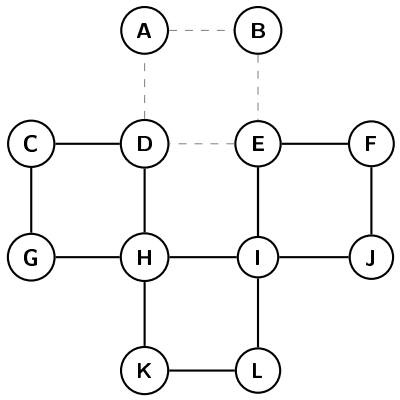
Afin d'effectuer une trace de l'algorithme, on donne l'état du graphe et de chacune des variables à chaque itération de la boucle **tant que**. On remarque que la variable i est incrémentée exactement une fois à chaque itération. On l'utilise donc pour identifier chacune des itérations.

- $i = 0$.
  Un chemin est construit à partir du sommet $circuit[i]$ = **A**. Les voisins de **A** étant les sommets **B** et **D**, le chemin débute avec l'arête **A**−**B** et celle-ci est retirée du graphe afin de s'assurer que le chemin constuit ne l'emprente pas une deuxième fois. Ensuite, à partir de **B** le chemin se poursuit à **E** puis à **D**. Les voisins du sommet **D** étant **A**, **C** et **H** (on rappelle que l'arête **E**−**D** a été retirée du graphe), le chemin se poursuit avec l'arête **D**−**A**. Le sommet **A** est alors de degré zéro et la fonction `chemin` termine et retourne la liste [A,B,E,D,A]. Finalement, à la ligne 7 de l'algorihtme de Hierholzer, la liste $circuit$ est modifiée. On remplace le sommet **A** par les sommets de la liste [A,B,E,D,A].
  Voici l'état des données après la première itération de la boucle **tant que**.



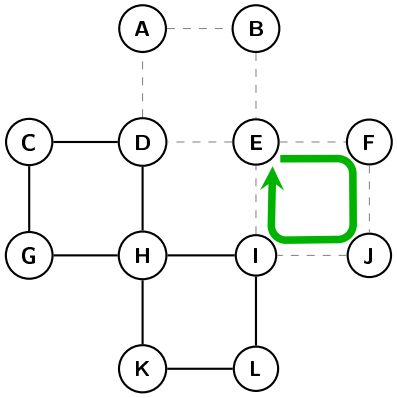
v = **A**,
$circuit$ = [A,B,E,D,A],

- $i = 1$.
  La fonction `chemin` est appelée à partir du sommet $circuit[1] = \mathsf{B}$. Ce sommet étant déjà de degré 0. La fonction `chemin` termine donc sans modifier le graphe et la liste retournée est simplement [B]. Ensuite, à la ligne 7, la modification effectuée à la liste $circuit$ n'a aucun effet puisque B est remplacé par B.



$v = \mathsf{A}$,
$circuit = [\mathsf{A},\underline{\mathsf{B}},\mathsf{E},\mathsf{D},\mathsf{A}]$,

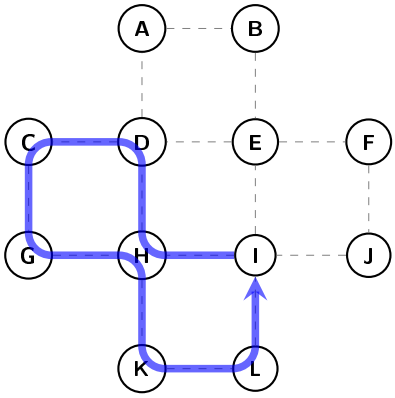
- $i = 2$.
  La fonction `chemin` est appelée à partir du sommet $circuit[2] = \mathsf{E}$. En respectant l'ordre alphabétique sur les sommets, le chemin construit est [E,F,J,I,E]. Ensuite, à la ligne 7, le sommet E est remplacé par ce chemin.



$v = \mathsf{A}$,
$circuit = [\mathsf{A},\mathsf{B},\underline{\mathsf{E},\mathsf{F},\mathsf{J},\mathsf{I},\mathsf{E}},\mathsf{D},\mathsf{A}]$,

- $i = 3$.
  Comme pour le cas $i = 1$, le sommet $circuit[3] = \mathsf{F}$ est de degré zéro. Le graphe et la liste $circuit$ restent inchangés.
- $i = 4$.
  Le sommet $circuit[4] = \mathsf{J}$ est de degré zéro.

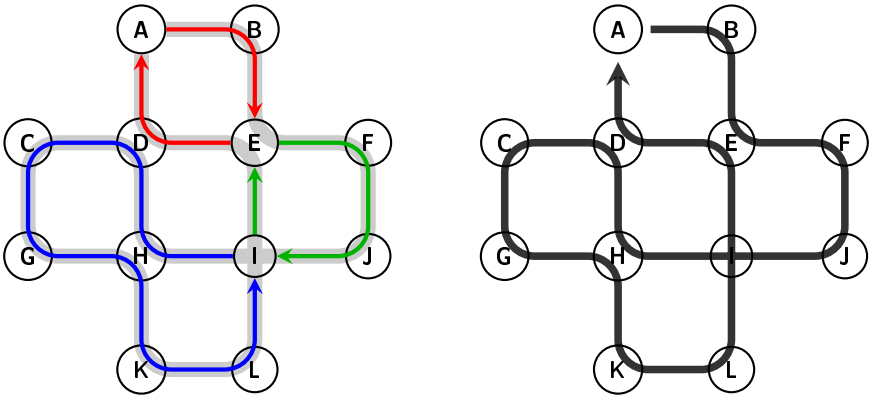
- $i = 5$.
  La fonction `chemin` est appelée à partir du sommet $circuit[5]$ = I et le chemin construit est [I,H,D,C,G,H,K,L,I]. Ensuite, à la ligne 7, le sommet I est remplacé par ce chemin.



$v = \mathsf{A}$,
$circuit = [\mathsf{A},\mathsf{B},\mathsf{E},\mathsf{F},\mathsf{J},\mathsf{I},\mathsf{H},\mathsf{D},\mathsf{C},\mathsf{G},\mathsf{H},\mathsf{K},\mathsf{L},\mathsf{I},\mathsf{E},\mathsf{D},\mathsf{A}]$.

- $i = 6,7,\ldots,16$.
  Toutes les arêtes ont été retirées du graphes. Il ne reste plus que des sommets de degré zéro. Aucune modification n'est apportée au données pendant ces itérations.

Le chemin eulérien calculé est donc : A–B–E–F–J–I–H–D–C–G–H–K–L–I–E–D–A.



3.3.4 Chemins et circuits hamiltoniens

Définition 3.8 : Chemins et circuits hamiltoniens

- Un **chemin hamiltonien** est un chemin qui passe exactement une fois par tous les sommets.
- Un **circuit hamiltonien** est un circuit formé par un chemin hamiltonien suivi d'une arête menant au sommet de départ.

Attention, les deux théorèmes suivants ne donnent pas des conditions nécessaires et suffisantes pour déterminer si un graphe non orienté possède un circuit hamiltonien. Si les hypothèses sont respectées, on peut affirmer qu'un tel circuit existe, mais dans le cas contraire, on ne peut rien conclure.

Théorème 3.6 : Théorème de Ore

Soit $G = (V, E)$ un graphe simple non orienté avec n sommets ($n \geq 3$). Si pour tous sommets non adjacents $u, v \in V$,

$$\deg(u) + \deg(v) \geq n$$

alors G possède un circuit hamiltonien.

Théorème 3.7 : Théorème de Dirac

Soit $G = (V, E)$ un graphe simple non orienté avec n sommets ($n \geq 3$). Si pour chaque sommet $v \in V$ on a

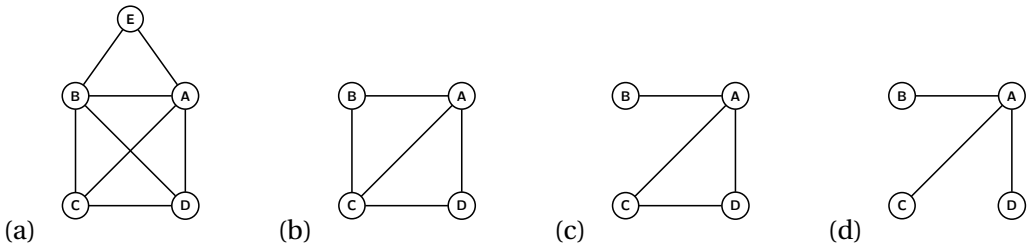
$$\deg(v) \geq \frac{n}{2},$$

alors G possède un circuit hamiltonien.

Le théorème de Dirac découle du théorème de Ore, bien qu'il ait été prouvé avant! En effet, si tous les sommets du graphe sont de degré plus grand ou égal à $\frac{n}{2}$, alors la somme des degrés de deux sommets non adjacents est forcément plus grande ou égale à $\frac{n}{2} + \frac{n}{2} = n$. Le théorème de Dirac est plus facile à vérifier, mais moins puissant que celui de Ore (voir exemple 3.14 (a)).

Exemple 3.14

Les graphes suivants possèdent-ils des circuits et chemins hamiltoniens? Est-ce que vous pouvez utiliser les Théorèmes 3.7 et 3.6 pour répondre à cette question?



Solution :

(a) A−D−C−B−E chemin hamiltonien ; A−D−C−B−E−A circuit hamiltonien.
Théorème 3.7 : non ; hypothèse n'est pas satisfaite puisque $\deg(E) = 2 \not\geq \frac{5}{2} = 2,5$.
Théorème 3.6 : oui ; hypothèse est satisfaite puisque $\deg(D) + \deg(E) = 5 \geq 5$ et $\deg(C) + \deg(E) = 5 \geq 5$.

(b) A−B−C−D chemin hamiltonien ; A−B−C−D−A circuit hamiltonien.
Théorème 3.7 : oui ; hypothèse est satisfaite puisque pour tout sommet v, $\deg(v) \geq \frac{4}{2} = 2$.
Théorème 3.6 : oui ; hypothèse est satisfaite puisque $\deg(B) + \deg(D) = 4 \geq 4$.

(c) B−A−D−C chemin hamiltonien ; pas de circuit hamiltonien.
Théorème 3.7 : non ; hypothèse n'est pas satisfaite puisque $\deg(B) = 1 \not\geq \frac{4}{2} = 2$.
Théorème 3.6 : non ; hypothèse n'est pas satisfaite puisque $\deg(B) + \deg(D) = 3 \not\geq 4$.

(d) pas de chemin hamiltonien ; pas de circuit hamiltonien.
Théorème 3.7 : non ; hypothèse n'est pas satisfaite puisque $\deg(B) = 1 \not\geq \frac{4}{2} = 2$.
Théorème 3.6 : non ; hypothèse n'est pas satisfaite puisque $\deg(B) + \deg(D) = 2 \not\geq 4$.

Exercices

3.4 Tracez les graphes non orientés demandés et donner un circuit eulérien et un circuit hamiltonien. S'il un tel circuit n'existe pas, donner un argument comme justification.

Note : Tous les graphes de cet exercice sont décrits à la page Wikipédia suivante :

https://en.wikipedia.org/wiki/Gallery_of_named_graphs

(a) Les graphes complets K_4, K_5 et K_6 (*complete graphs*).

(b) Les graphes bipartis complets $K_{3,3}$, $K_{2,4}$ et $K_{4,4}$ (*complete bipartite graphs*).

(c) Les graphes d'amitié F_2, F_3, F_4 (*friendship graphs*).

(d) ★ Le graphe de Peterson.

(e) ★ Les solides de Platon.

3.4 Problème du plus court chemin (algorithme de Dijkstra)

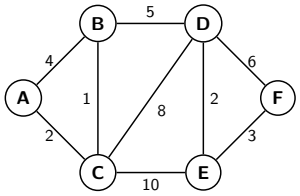
Plusieurs situations peuvent être modélisées à partir de graphes ayant un coût associé aux arêtes. Par exemple, si l'on veut représenter les distances qui relient les villes du Québec, les sommets du graphe sont identifiés par les villes et les arêtes par les routes qui les relient, ainsi que leurs distances. Comment trouver le chemin de distance minimale entre deux villes ?

Définition 3.9 : Graphe pondéré, distance, plus court chemin

Un **graphe pondéré** est un graphe qui a un poids (ou coût) associé à chacune de ses arêtes. La **distance** (ou **coût** ou **poids total**) d'un chemin dans un graphe pondéré est la somme de chacun des poids des arêtes de ce chemin. Le **plus court chemin** (ou **chemin de coût minimal** ou **chemin de poids minimal**) d'un sommet à un autre est celui de distance minimale.

Exemple 3.15

Le graphe suivant est un graphe pondéré.



Le chemin A−C−D−F allant du sommet A au sommet F est de distance $2+8+6=16$.

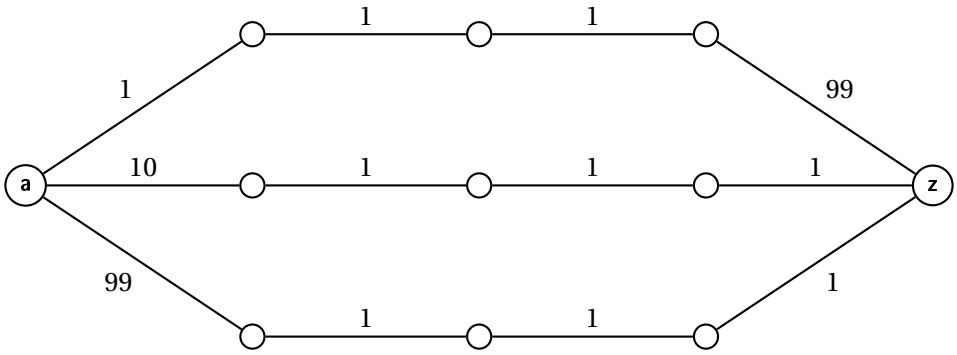
Le chemin A−C−B−D−E−F allant du sommet A au sommet F est de distance minimale $2+1+5+2+3=13$.

L'**algorithme de Dijkstra** donne la distance minimale d'un chemin allant d'un sommet à un autre dans un graphe pondéré de **poids positifs**.

Afin d'obtenir un chemin optimal, quelle que soit la structure du graphe considéré, l'algorithme se doit d'effectuer une exploration du graphe et de considérer plusieurs possibilités. Par contre, afin d'être le plus efficace possible, l'algorithme ne considère pas *toutes* les possibilités.

Exemple 3.16

On considère le graphe ci-dessous. Un simple coup d'oeil permet de constater que le chemin le moins coûteux de a à z est celui du milieu. Le principe de l'algorithme de Dijkstra est de se positionner au sommet de départ puis d'effectuer une exploration *intelligente* du graphe.



Si on considère les arêtes qui sortent du sommet a, on constate que le chemin du haut doit être exploré puisqu'il commence avec des arêtes peu coûteuses. Le chemin du milieu doit lui aussi être exploré puisqu'il constitue la solution optimale avec un coût de 13. Par contre, il est inutile d'explorer le chemin du bas plus loin que sa première arête puisque son coût sera forcément supérieur à 13.

Algorithme 3 `Dijkstra`

Entrées: $G = (V, E)$ un graphe simple pondéré connexe (orienté ou non orienté), $\omega : E \longrightarrow \mathbb{R}^+$ la fonction de pondération, $\mathsf{a} \in V$ le sommet de départ, $\mathsf{z} \in V$ le sommet d'arrivée.
Sortie: coût minimal d'un chemin allant du sommet a au sommet z

1: $S := \emptyset$ ▷ sommets dont le chemin optimal est connu
2: $L :=$ tableau indexé par les sommets ▷ coût du plus court chemin connu
3: $P :=$ tableau indexé par les sommets ▷ prédécesseur dans le plus court chemin connu
4: **pour tout** sommet $\mathsf{v} \in V$ **faire**
5: $L(\mathsf{v}) := \infty$ ▷ coût du plus court chemin connu de a et v
6: $P(\mathsf{v}) :=$ `null` ▷ v n'a pas encore de prédécesseur
7: **fin pour**
8: $L(\mathsf{a}) := 0$ ▷ a est le point de départ, il est à distance 0 de lui-même
9: **tant que** $\mathsf{z} \notin S$ **faire**
10: $\mathsf{u} :=$ sommet $\notin S$ avec $L(\mathsf{u})$ minimal
11: $S := S \cup \{\mathsf{u}\}$ ▷ le plus court chemin connu de a à u est optimal
12: **pour tout** sommets v tel que $(\mathsf{u}, \mathsf{v}) \in E$ et $\mathsf{v} \notin S$ **faire** ▷ les voisins de u qui ne sont pas dans S
13: **si** $L(\mathsf{u}) + \omega(\mathsf{u}, \mathsf{v}) < L(\mathsf{v})$ **alors** ▷ s'il est plus court d'aller à u puis prendre l'arc (ou l'arête) (u, v)
14: $P(\mathsf{v}) := \mathsf{u}$ ▷ nouveau meilleur chemin : on accède à v par le sommet u
15: $L(\mathsf{v}) := L(\mathsf{u}) + \omega(\mathsf{u}, \mathsf{v})$ ▷ coût de ce nouveau meilleur chemin
16: **fin si**
17: **fin pour**
18: **fin tant que**
19: **retourner** $L(\mathsf{z})$

Afin de déterminer un chemin de distance minimale allant du sommet a au sommet z, le principe de l'algorithme de Dijkstra consiste à séparer les sommets en deux sous-ensembles : ceux pour lesquels on connaît un chemin le plus court et ceux pour lesquels on ne le connaît pas encore. Plus précisément, on définit :

- S : l'ensemble des sommets u pour lesquels on connaît un chemin de distance minimale allant de a à u.

À l'initialisation, l'ensemble S est vide. À chaque itération de la boucle principale, un sommet est ajouté à l'ensemble S. Logiquement, si on veut calculer un plus court chemin allant de a à z, l'algorithme termine dès que le sommet z est ajouté à l'ensemble S.

En plus de cet ensemble, l'algorithme utilise deux tableaux indexés par les sommets :

- L : pour chaque sommet u, $L(\mathsf{u})$ est l'*étiquette* du sommet u. Il s'agit de la distance du plus court chemin **actuellement connu** allant du sommet a au sommet u.
- P : pour chaque sommet u, $P(\mathsf{u})$ est le *prédécesseur* de u. Il s'agit du sommet à partir duquel on accède à u dans le plus court chemin **actuellement connu** allant de a à u.

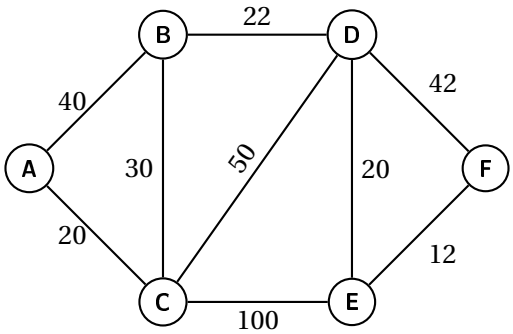
À l'initialisation, le tableau L associe la valeur ∞ à tous les sommets sauf pour le sommet de départ auquel on attribue la valeur zéro. Cela correspond au fait qu'au moment de l'initialisation, on ne connaît rien du graphe. Pour le tableau P, on affecte une valeur bidon à tous les sommets. Cette valeur bidon ne doit correspondre à aucun sommet. Dans le pseudo-code, on utilise le mot-clé `null` pour la représenter [1].

1. Si les sommets sont les entiers de 0 à $n-1$, on peut utiliser la valeur -1. Dans certains langages de programmation, il est possible d'utiliser des mots-clés comme `NULL` (C/C++), `null` (Java), `None` (Python), `Nothing` (VB), etc.

On peut montrer qu'une implémentation de l'algorithme de Dijkstra utilisant un tableau de booléen pour représenter l'ensemble S nécessite $O(n^2)$ opérations [2] pour calculer un chemin de poids minimal entre deux sommets d'un graphe à n sommets.

Exemple 3.17

Considérez le graphe pondéré ci-dessous, où les coûts sont en $.



(a) Trouvez le coût minimal d'un chemin allant du sommet **C** au sommet **E**. *Laissez une trace détaillée de l'algorithme*, en indiquant le sommet **u**, l'ensemble S, les étiquettes et les prédécesseurs de chacun des sommets à chaque passage dans la boucle **tant que** de la ligne 9.

(b) Donnez un chemin de coût minimal allant du sommet **C** au sommet **E**.

Solution :

(a) L'addition $L(\mathbf{u}) + \omega(\mathbf{u},\mathbf{v})$ est indiquée entre parenthèses. Si le résultat est inférieur à l'étiquette du sommet **v**, alors l'étiquette est mise à jour. Pour alléger la trace, on indique ici l'étiquette $L(\mathbf{v})$ et le prédécesseur $P(\mathbf{v})$ dans la même colonne en utilisant la convention L_P.

u	A	B	a = C	D	z = E	F	S
	∞	∞	0	∞	∞	∞	$\emptyset$
C	$(0+20=20)$ $20_{\mathbf{C}}$	$(0+30=30)$ $30_{\mathbf{C}}$		$(0+50=50)$ $50_{\mathbf{C}}$	$(0+100=100)$ $100_{\mathbf{C}}$	$(0+\infty=\infty)$ ∞	$\{\mathbf{C}\}$
A		$(20+40=60)$ $30_{\mathbf{C}}$		$(20+\infty=\infty)$ $50_{\mathbf{C}}$	$(20+\infty=\infty)$ $100_{\mathbf{C}}$	$(20+\infty=\infty)$ ∞	$\{\mathbf{A},\mathbf{C}\}$
B				$(30+22=52)$ $50_{\mathbf{C}}$	$(30+\infty=\infty)$ $100_{\mathbf{C}}$	$(30+\infty=\infty)$ ∞	$\{\mathbf{A},\mathbf{B},\mathbf{C}\}$
D					$(50+20=70)$ $70_{\mathbf{D}}$	$(50+42=92)$ $92_{\mathbf{D}}$	$\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}\}$
E							$\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D},\mathbf{E}\}$

Le **tant que** prend fin quand $\mathbf{z} \in S$ et retourne l'étiquette du sommet **z** (dans notre cas, $\mathbf{z} = \mathbf{E}$) : $L(\mathbf{z}) = 70$. Ainsi, le coût minimal pour aller de **C** à **E** est 70$.

(b) Pour retrouver un chemin de coût minimal, on remonte la chaîne des prédécesseurs à partir du sommet d'arrivée : $P(\mathbf{E}) = \mathbf{D}$ et $P(\mathbf{D}) = \mathbf{C}$. Ainsi, **C**−**D**−**E** est un chemin de coût minimal allant de **C** à **E**.

2. Cette notation est vue au chapitre sur la complexité des algorithmes.

Exemple 3.18

La trace de l'exemple précédent permet-elle de donner aussi le coût minimal pour aller :

(a) du sommet C au sommet B ?

(b) du sommet C au sommet F ?

(c) du sommet A au sommet E ?

Solution :

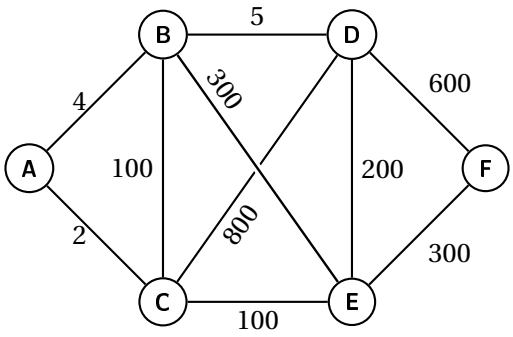
(a) Oui. Le sommet B a été visité ($u = B$ à la troisième étape), donc l'étiquette de B correspond bien au coût minimal pour aller de C à B : 30$.

(b) Non, car la trace s'est arrêtée sans que le sommet F n'ait été visité. L'étiquette de F ne correspond donc pas nécessairement au coût minimal pour aller de C à F. En fait, on remarque que le chemin C−D−E−F a un coût de 82$ tandis que $L(F) = 92$.

(c) Non, car cette trace correspond au sommet initial C et non A. Toutes les étiquettes correspondent à un coût pour un chemin issu de C.

Exemple 3.19

Considérez le graphe pondéré ci-dessous, où les poids correspondent à des temps de parcours en millisecondes.



(a) Trouvez le temps minimal pour aller du sommet A au sommet F.
Laissez une trace de l'algorithme.

(b) Donnez un chemin de temps minimal allant du sommet A au sommet F.

(c) La trace faite en (a) permet-elle aussi de donner le temps minimal pour aller de A à E ?

Solution :

(a) Trace de l'algorithme. Le détail des additions $L(\mathbf{u}) + \omega(\mathbf{u}, \mathbf{v})$ n'est pas indiqué.

u	a = A	B	C	D	E	z = F	S
	0	∞	∞	∞	∞	∞	$\emptyset$
A		4_{A}	2_{A}	∞	∞	∞	{A}
C		4_{A}		802_{C}	102_{C}	∞	{A, C}
B				9_{B}	102_{C}	∞	{A, B, C}
D					102_{C}	609_{D}	{A, B, C, D}
E						402_{E}	{A, B, C, D, E}
F							{A, B, C, D, E, F}

Le temps minimal pour aller de A à F est donc 402 millisecondes.

(b) Pour retrouver un chemin minimal, on remonte la chaîne des prédécesseurs à partir du sommet d'arrivée :

$$P(\mathsf{F}) = \mathsf{E}, \quad P(\mathsf{E}) = \mathsf{C}, \quad P(\mathsf{C}) = \mathsf{A}$$

ainsi, A – C – E – F est un chemin de coût minimal allant de A à F.

(c) Oui. Puisque le sommet E a été visité, la trace faite en (a) permet aussi de donner le temps minimal pour aller de A à E : 102 millisecondes.

3.4.1 Exemple : plan d'approvisionnement

En décembre, une usine doit planifier son approvisionnement en matière première (pierre) auprès de son fournisseur pour les 4 prochains mois. L'équipe de planification doit tenir compte de plusieurs facteurs. Si elle décide de passer une seule grosse commande pour diminuer le coût d'achat et de livraison, elle devra payer plus cher en frais d'entreposage. Au contraire, si elle décide de passer une petite commande à chaque mois, les coût d'entreposage seront faibles mais les coûts de livraison seront plus importants. De plus, les prix de la matière première et de la livraison varient dans l'année, tout comme les coûts d'entreposage, mais on suppose ici qu'ils sont connus à l'avance. Le cours *GOL455-Gestion des opérations, des flux et des stocks* aborde en détails la question de l'approvisionnement ; nous présentons ici une application de la théorie des graphes en simplifiant le problème.

mois	besoin (en tonnes)
J	20
F	30
M	40
A	30

Figure 3.6 Représentation des besoins en pierre d'une usine pour les prochains mois.

Dans le graphe orienté de la figure 3.7, les sommet correspondent à des mois de l'année : 0 - Décembre, 1 - Janvier, 2 - Février, etc. Le poids d'un arc allant du sommet i au sommet j correspond au coût engendré par une commande pour couvrir le besoin des mois $i+1$ à j, cette livraison arrivant au début de la période $i+1$. Le coût tient compte de tous les facteurs : coût unitaire, coût de livraison, coût d'entreposage, etc. Pour clarifier la situation, a quantité de la commande est indiquée entre parenthèses à côté du coût (ce n'est pas nécessaire, car cette information peut-être obtenue à partir du tableau des besoins).

Par exemple, considérons l'arc tireté et bleu allant du sommet **1** au sommet **4** : $c(1,4) = 5100\$$ est le coût engendré par une commande de 100 tonnes, livrée au début de février, couvrant les besoin de l'usine pour les mois de février à avril.

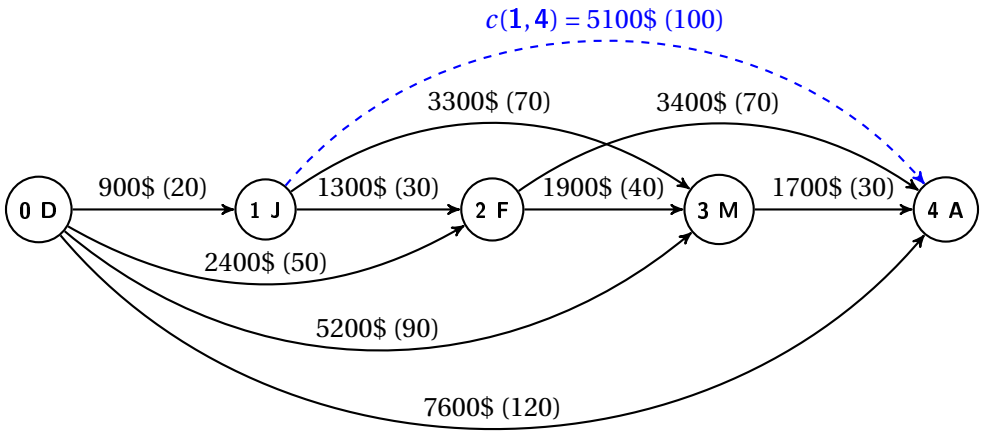


Figure 3.7 Représentation des coûts engendrés par des commandes pour différentes périodes. Exemple adapté des cours GOL455, section 3.2, ÉTS et GSO-6080, ULaval.

Sur le graphe, un chemin allant du sommet **0** au sommet **4** correspond donc a un plan d'approvisionnement. Par exemple, le chemin **0-1-2-3-4** correspond à passer 4 petites commandes, au coût total de 900$ + 1300$ + 1900$ + 1700$ = 5800$. Le chemin **0−1−4** correspond à passer une commande de 20 tonnes pour janvier, puis de 100 tonnes pour couvrir les besoins de février à avril, au coût total de

$$900\$ + 5100\$ = 6000\$.$$

Le chemin de coût minimal allant du sommet 0 au sommet 4 correspond au plan le moins cher. Pour l'obtenir, on peut utiliser l'algorithme de Dijkstra.

u	a = 0	1	2	3	z = 4
	0	∞	∞	∞	∞
0		(0 + 900 = 900) $900_{\mathbf{0}}$	(0 + 2400 = 2400) $2400_{\mathbf{0}}$	(0 + 5200 = 5200) $5200_{\mathbf{0}}$	(0 + 7600 = 7600) $7600_{\mathbf{0}}$
1			(900 + 1300 = 2200) $2200_{\mathbf{1}}$	(900 + 3300 = 4200) $4200_{\mathbf{1}}$	(900 + 5100 = 6000) $6000_{\mathbf{1}}$
2				(2200 + 1900 = 4100) $4100_{\mathbf{2}}$	(2200 + 3400 = 5600) $5600_{\mathbf{2}}$
3					(4100 + 1700 = 5800) $5600_{\mathbf{2}}$
4					

Figure 3.8 Trace de l'algorithme de Dijkstra : le coût minimal du chemin allant de **0** à **4** est de 5600$. Ce chemin est obtenu en remontant la chaîne des prédécesseurs (en indices) : **0−1−2−4**.

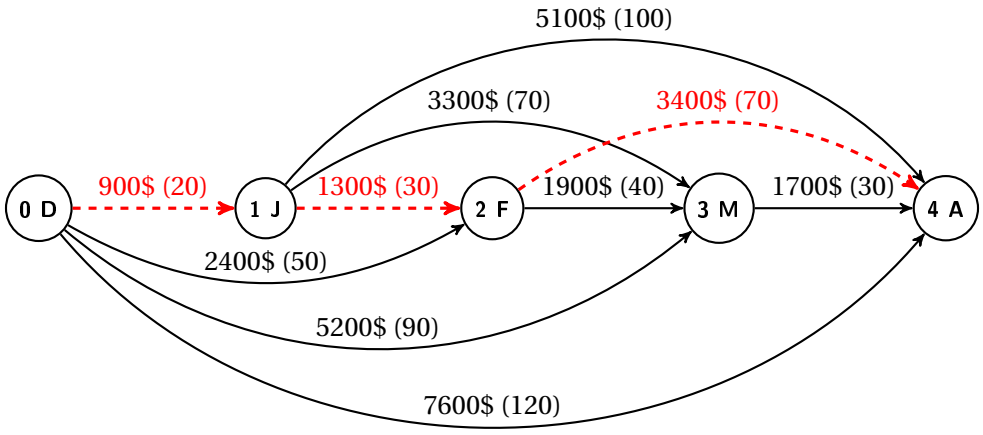
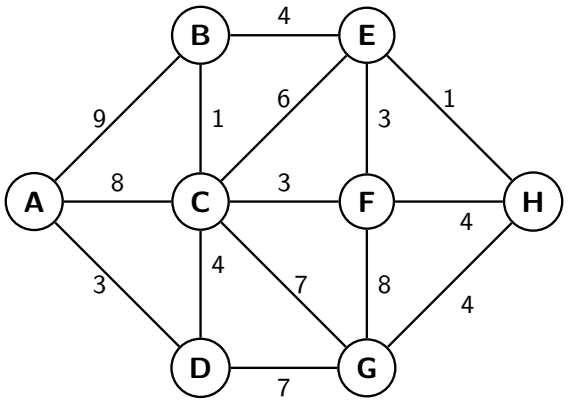


Figure 3.9 Le chemin de coût minimal allant du sommet **0** au sommet **4** est indiqué en tireté : **0−1−2−4** . Il correspond à recevoir 20 tonnes en janvier, 30 en février et 70 en mars (pour couvrir les besoins de mars et avril).

Exercices

3.5 Considérez le graphe suivant :



(a) Trouvez la distance minimale d'un chemin allant de **A** à **H**.

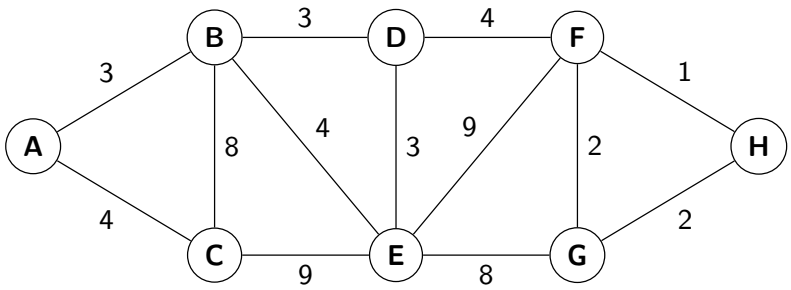
(b) Donnez explicitement ce chemin.

(c) Trouvez la distance minimale d'un chemin allant de **A** à **B**.

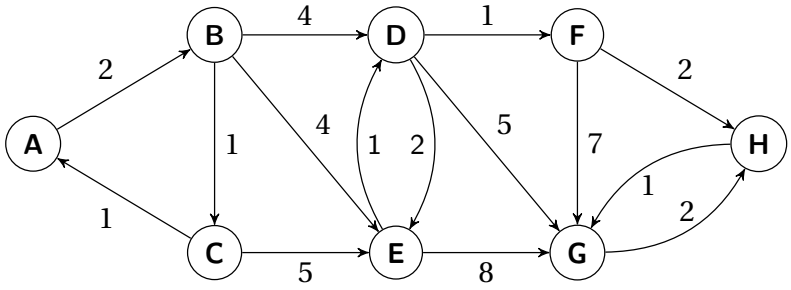
(d) Donnez explicitement ce chemin.

3.6 Pour chacun des graphes suivants, utilisez l'algorithme de Dijkstra afin de calculer la distance minimale d'un chemin de **C** à **G** et donner le chemin obtenu.

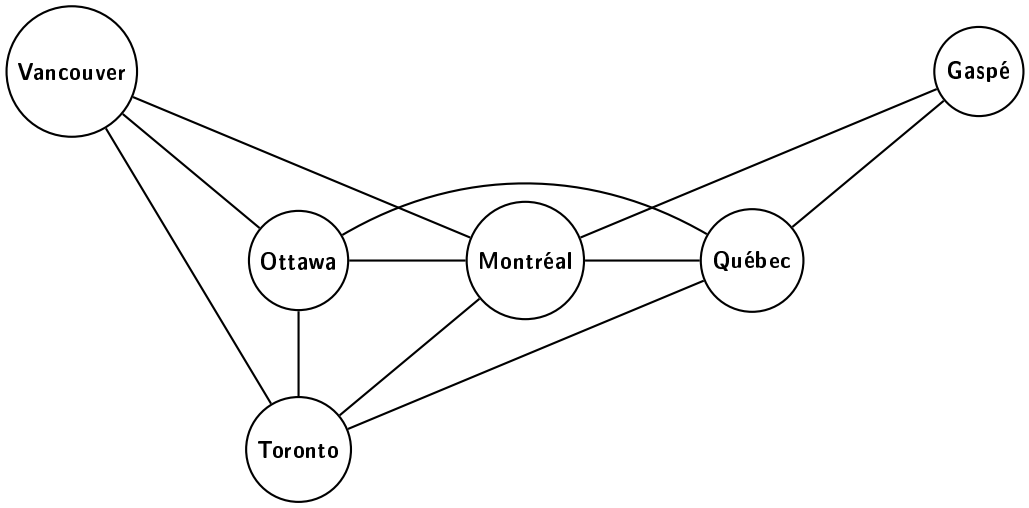
(a)



(b)



3.7 Exercice de révision du chapitre.
Considérez le graphe suivant représentant les liens aériens de la compagnie AéroÉTS entre certaines villes du Canada.



(a) Combien y a-t-il d'itinéraires possibles d'au plus deux escales de Gaspé à Vancouver?

(b) Combien y a-t-il d'itinéraires possibles ayant une et une seule escale partant de Montréal?

(c) Combien y a-t-il d'itinéraires possibles ayant exactement deux escales, avec Toronto comme ville de départ et d'arrivée?

(d) Combien y a-t-il d'itinéraires possibles ayant exactement sept escales, avec Toronto comme ville de départ et d'arrivée?

(e) Combien y a-t-il d'itinéraires possibles de quatre escales partant de Gaspé et revenant à Gaspé?

(f) S'agit-il d'un graphe simple et non orienté?

(g) Ce graphe vérifie-t-il les conditions du théorème 3.7? Et celles du théorème 3.6?

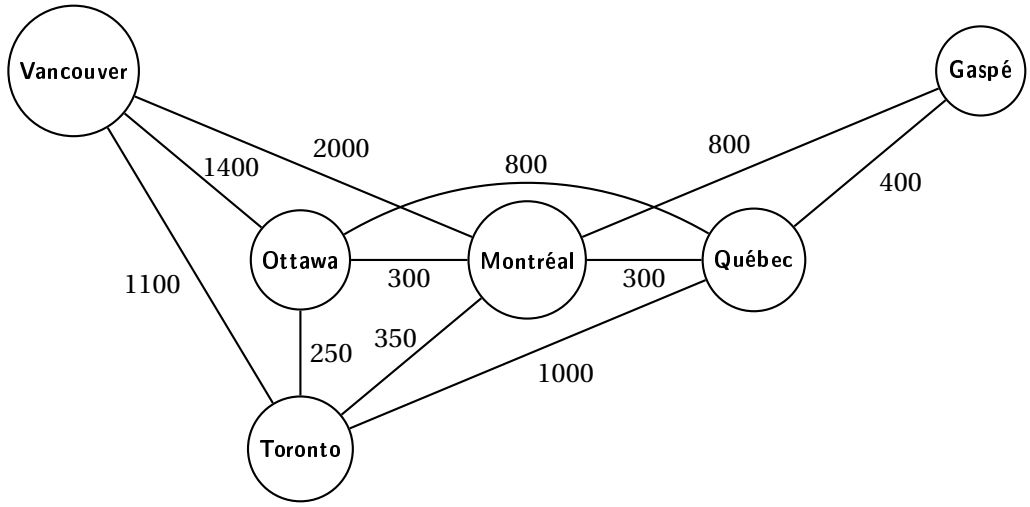
(h) Existe-t-il un circuit hamiltonien dans ce graphe? Si oui, donnez-en un.

(i) Existe-t-il un circuit eulérien dans ce graphe? Si oui, donnez-en un.

(j) Existe-t-il un chemin eulérien dans ce graphe? Si oui, donnez-en un.

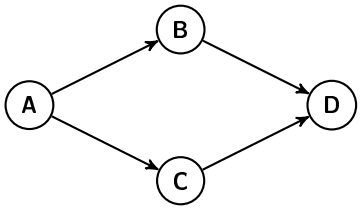
(k) Comment qualifie-t-on un graphe où chaque arête est associée à un poids?

(l) Étant donné les prix indiqués (en dollars), utilisez l'algorithme de Dijkstra afin de trouver un trajet de coût minimal entre Gaspé et Vancouver. Puis entre Toronto et Gaspé. Y a-t-il toujours le même nombre de lignes dans le tableau des étiquettes? Autrement dit, l'algorithme de Dijkstra exécute-t-il toujours le même nombre d'étapes pour obtenir le chemin de coût minimal entre deux sommets?



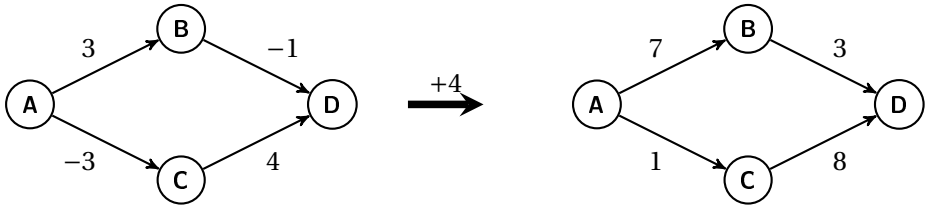
3.8 En entrée de l'algorithme de Dijkstra, on décrit la fonction de pondération comme étant $\omega : E \longrightarrow \mathbb{R}^+$. Le but de cet exercice est de comprendre pourquoi on impose que les pondérations ne soient pas négatives.

(a) Pondérez le graphe ci-dessous avec des valeurs positives et négatives de sorte que l'algorithme de Dijkstra ne produise pas un plus court chemin entre les sommets **A** et **D**.



(b) Effectuez une trace de l'algorithme de Dijkstra sur le graphe pondéré en (a) afin de montrer que le chemin calculé par l'algorithme n'est pas minimal.

(c) Afin de calculer un plus court chemin dans un graphe possédant des pondérations négatives, on pourrait être tenté d'additionner une même valeur à toutes les pondérations de sorte qu'elles soient toutes positives. Par exemple, dans le graphe ci-dessous, la plus petite pondération est -3. En additionnant 4 sur toutes les arêtes, on obtient un graphe avec des pondérations strictement positives.



Montrez que cette méthode ne fonctionne pas en général en fournissant un contre-exemple.

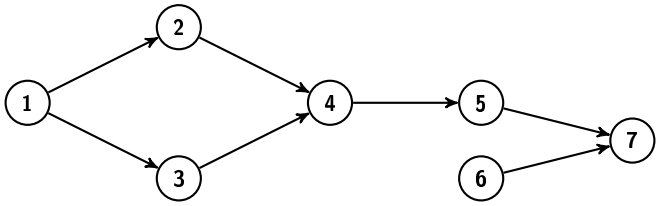
3.5 Tri topologique (algorithme de Kahn)

Un tri topologique est un ordonnancement des sommets d'un graphe orienté acyclique. Avant de le définir formellement, nous débutons avec un détour du côté de la gestion de projets.

Un projet d'envergure est habituellement découpé en plusieurs tâches. Parmi ces tâches, certaines sont préalables à d'autres. À titre d'illustration, on considère le cas d'une personne qui se rend à l'ÉTS pour assister à un cours de mathématiques. On identifie les tâches suivantes :

#	Description de la tâche	Tâche(s) préalable(s))
1	Se rendre à l'ÉTS	
2	Se procurer un café	1
3	Aller au toilettes	1
4	Se rendre à la salle de classe	2,3
5	Sortir son cahier de notes	4
6	Éteindre son téléphone	
7	Suivre le cours avec attention	5,6

Les contraintes sur l'ordre dans lequel ces tâches doivent être réalisées peuvent être modélisées par un graphe orienté où chaque tâche est représentée par un sommet.

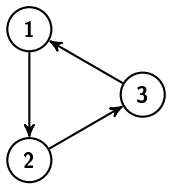


Dans ce graphe, un arc (u, v) signifie que la tâche u doit être réalisée avant la tâche v. Par contre, un tel arc ne signifie pas que la tâche doit être réalisée immédiatement avant. Ainsi, la tâche 2 peut être réalisée avant ou après la tâche, 3 mais les deux doivent être réalisées avant la tâche 4. De même, la tâche 6 peut être réalisée à tout moment, tant que ce n'est pas après le début de la tâche 7. Ainsi, il existe plusieurs ordonnancements valides, en voici quatre à titre d'exemples :

- 1, 2, 3, 4, 5, 6, 7.
- 1, 3, 2, 4, 5, 6, 7.
- 6, 1, 2, 3, 4, 5, 7.
- 1, 3, 6, 2, 4, 5, 7.

De manière générale, pour qu'un ordonnancement valide existe, il ne faut pas qu'une tâche soit préalable à une tâche qui lui est elle-même préalable. Voici un exemple d'une liste de tâches mal construite :

#	Description de la tâche	Tâche(s) préalable(s)
1	Acquérir une bonne expérience professionnelle	3
2	Obtenir un emploi stable	1
3	Conserver un emploi	2



Une telle dépendance circulaire est équivalente à la présence d'un cycle dans le graphe. Ainsi, pour qu'une liste de tâche soit réalisable, il faut que son graphe soit acyclique.

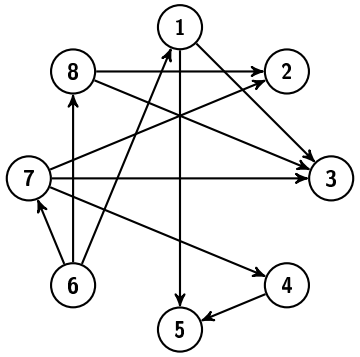
Définition 3.10

Étant donné $G = (V, E)$ un graphe orienté acyclique, un **tri topologique** est une liste dans laquelle chaque sommet de G apparait exactement une fois et pour tout arc $(u, v) \in E$, le sommet u apparait avant le sommet v.

De manière générale, donner un tri topologique est équivalent à disposer les sommets du graphe de sorte que tous les arcs pointent du même côté.

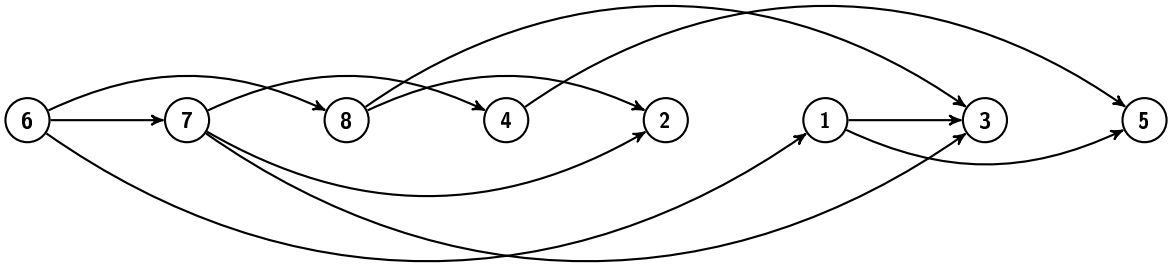
Exemple 3.20

Donnez un tri topologique pour le graphe suivant :



Solution :

La liste $6, 7, 8, 4, 2, 1, 3, 5$ est un tri topologique. En disposant les sommets dans cet ordre, le graphe devient :

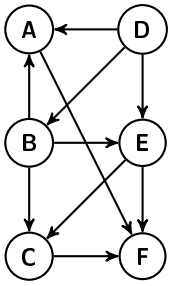


Comme le montre l'exemple précédent, déterminer un tri topologique peut vite devenir fastidieux si on ne procède pas de manière systématique. La section qui suit présente l'algorithme de *Kahn* qui permet de trouver facilement un tri topologique.

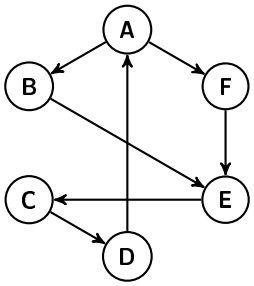
Exercices

3.9 Pour chacun des graphes suivants, donnez un tri topologique s'il en existe un et vérifiez sa validité en traçant le graphe avec les sommets positionnés dans cet ordre. S'il n'en existe pas, expliquez pourquoi.

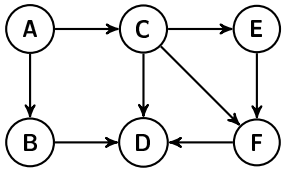
(a)



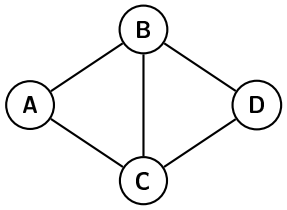
(b)



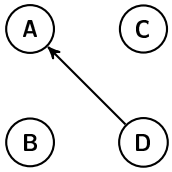
(c)



(d)



(e)



Algorithme de Kahn

Exprimé en termes d'ordonnancement de tâches, l'algorithme de Kahn est basé sur un principe fort simple : « *Si une tâche n'a pas de préalable, alors on peut commencer avec celle-ci. Une fois cette tâche terminée, on la retire de la liste des préalables et on recommence* ». En termes de graphes, cela revient à identifier un sommet ne possédant aucun arc entrant. Un tel sommet peut toujours être le premier dans un tri topologique. Ensuite, on supprime ce sommet ainsi que tous les arcs sortants de ce sommet et on recommence.

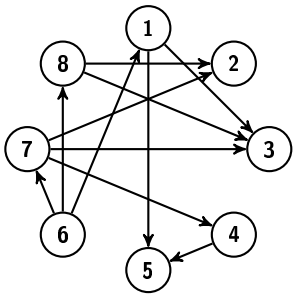
Algorithme 4 `Kahn`

Entrées: $G = (V, E)$ un graphe orienté et acyclique
Sortie: Un tri topologique des sommets de G

1: L := liste vide
2: **tant que** il reste des sommets **faire**
3: **u** := un sommet n'ayant aucun arc entrant.
4: Ajouter **u** à la fin de la liste L
5: Supprimer le sommet **u** et tous les arcs sortants de **u**
6: **fin tant que**
7: **retourner** L

Exemple 3.21

On reprend le graphe de l'exemple 3.20 et on le représente sous la forme d'une table. Dans cette table, pour chaque sommet **v**, on donne la liste des sommets **u** pour lesquels il existe un arc **u**→**v**. Afin d'illustrer clairement le déroulement de l'algorithme, on reproduit le graphe et la table à chaque itération de la boucle principale. En pratique, il n'est pas nécessaire d'en faire autant, on se contente de rayer les sommets de la colonne de droite de la table à mesure qu'ils sont traités.

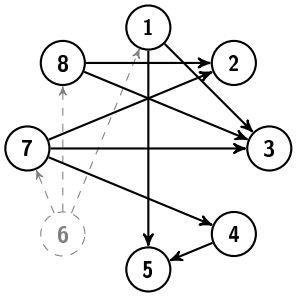


Sommet **v**	Sommet(s) **u** avec **u**→**v** ∈ E
1	6
2	7,8
3	1,7,8
4	7
5	1,4
6	
7	6
8	6

u =

$L = [\,]$

- **Itération 1**, seul le sommet 6 n'a aucun arc entrant, on pose donc **u** = 6.

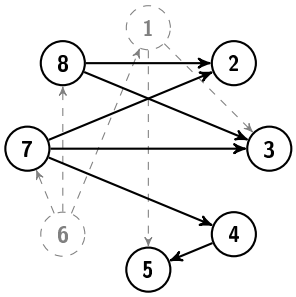


Sommet **v**	Sommet(s) **u** avec **u**→**v** ∈ E
1	
2	7,8
3	1,7,8
4	7
5	1,4
6	
7	
8	

u = 6

$L = [6]$

- **Itération 2**, les sommets 1, 7 et 8 n'ont aucun arc entrant. Par convention, on traite les sommets en ordre croissant, on commence donc par 1.

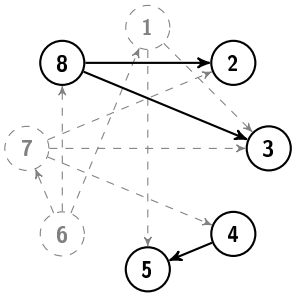


Sommet **v**	Sommet(s) **u** avec **u**→**v** ∈ E
1	
2	7,8
3	7,8
4	7
5	4
6	
7	
8	

u = 1

$L = [6,1]$

- **Itération 3**, on traite le sommet **7**.

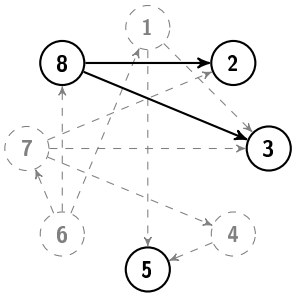


Sommet **v**	Sommet(s) **u** avec **u**→**v** ∈ E
1	
2	8
3	8
4	
5	4
6	
7	
8	

u = **7**

L = [**6**, **1**, **7**]

- **Itération 4**, on traite le sommet **4**.

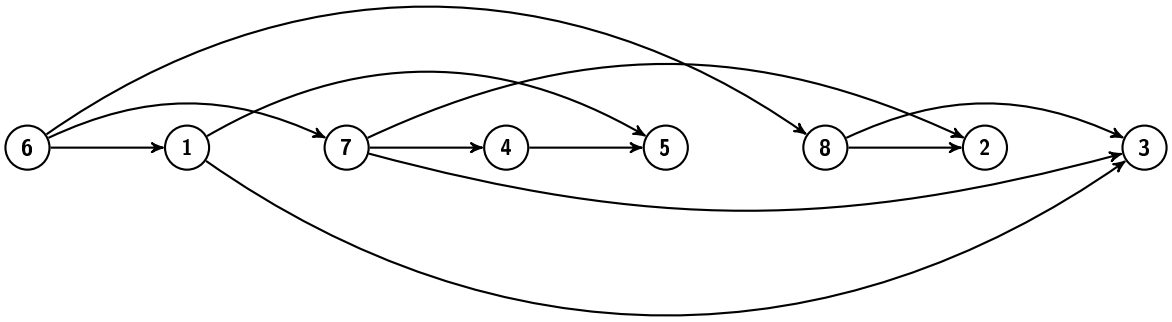


Sommet **v**	Sommet(s) **u** avec **u**→**v** ∈ E
1	
2	8
3	8
4	
5	
6	
7	
8	

u = **4**

L = [**6**,**1**,**7**,**4**]

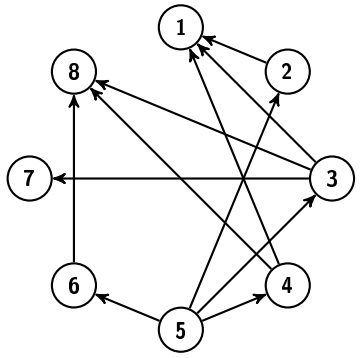
En continuant de la sorte, on traite ensuite les sommets **5**, **8**, **2** et **3**. Le tri topologique obtenu est: [**6**, **1**, **7**, **4**, **5**, **8**, **2**, **3**].



Exercices

3.10 Pour chacun des graphes suivants, utililez l'algorithme de Kahn afin de déterminer un tri topologique. Tracez ensuite le graphe en positionnant les sommets dans l'ordre correspondant.

(a) Le graphe suivant :



(b) Le graphe dont la matrice d'adjacence est :

$$M = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

(c) Le graphe dont la représentation par listes d'adjacences est :

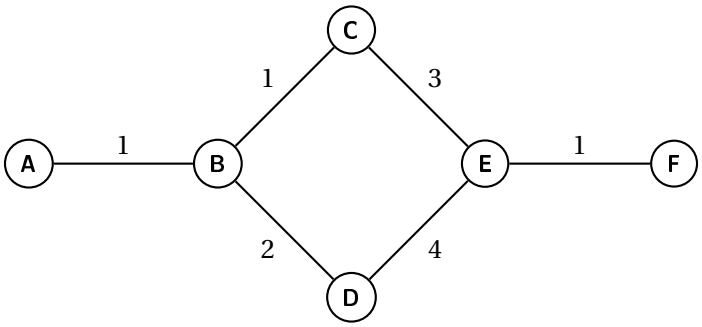
1	2, 3, 5, 6, 7
2	6, 7
3	2, 4, 5, 8
4	7
5	4
6	4, 5, 7
7	
8	2, 5, 6, 7

3.6 Chemin de poids maximal

À priori, le problème du chemin le plus long entre deux sommets d'un graphe est un problème mal posé, comme l'illustre l'exemple suivant.

Exemple 3.22

Quel est le chemin de poids maximal de A à F ?



On pourrait penser que le chemin A – B – D – E – F maximise le poids avec un total de 8, mais le chemin A – B – D – E – C – B – D – E – F a un poids de 18.

Il en va de même pour le chemin A – B – D – E – C – B – D – E – C – B – D – E – F qui a un poids total de 28. On remarque que chaque fois qu'on parcourt le circuit B – D – E – C – B, on ajoute 10 au poids du chemin. Comme il n'y a pas de limite au nombre de tours qu'on peut effectuer, il n'existe pas de chemin qui maximise le poids.

En effet, pour n'importe quel poids $p \in \mathbb{N}$, il existe un chemin dont le poids est strictement supérieur.

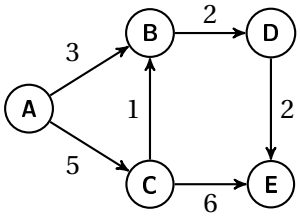
L'exemple ci-dessus montre donc que le problème consistant à déterminer un chemin de pondération maximal dans un graphe quelconque est sans intérêt. Il n'en demeure pas moins que si on impose certaines contraintes sur le graphe alors, le problème devient intéressant tant au point de vue théorique que pratique.

3.6.1 Chemin de poids maximal dans un graphe orienté acyclique

Contrairement au cas général, le problème du chemin de poids maximal dans un graphe orienté acyclique est un problème bien défini. En effet, la structure particulière d'un tel graphe impose qu'il n'existe qu'un nombre fini de chemins distincts dans le graphe et donc il en existe forcément un qui maximise le poids.

Exemple 3.23

Afin de déterminer un chemin de poids maximal dans un graphe orienté acyclique, il est possible de procéder par énumération de tous les chemins possibles. En effet, un tel graphe possède toujours un nombre fini de chemins distincts.



Chemin	Poids	Chemin	Poids
A – B	3	B – D	2
A – B – D	5	B – D – E	4
A – B – D – E	7	C – B	1
A – C	5	C – B – D	3
A – C – B	6	C – B – D – E	5
A – C – B – D	8	C – E	6
A – C – B – D – E	10	D – E	2
A – C – E	11		

On constate ainsi que le chemin qui maximise le poids est A – C – E.

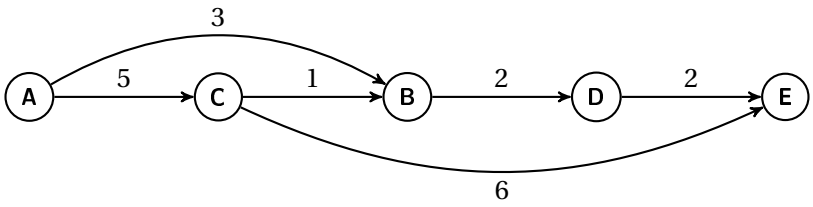
Dans l'exemple précédent, bien que le graphe utilisé compte seulement 5 sommets et 6 arcs, le graphe possède tout de même 15 chemins distincts. De manière générale, le nombre de chemins dans un graphe orienté acyclique à n sommets peut atteindre $2^n - n - 1$. Le fait que le nombre de chemins est donné par une fonction exponentielle rend cette méthode inutilisable en pratique. Quoique naïf, l'exemple 3.23 met en évidence deux faits :

- un sommet possédant une arête entrante ne peut pas être le point de départ d'un chemin maximal.
- un sommet possédant une arête sortante ne peut pas être le point d'arrivée d'un chemin maximal.

À la section 3.5, on a vu qu'un graphe orienté acyclique peut être représenté de telle sorte que tous les arcs pointent dans la même direction grâce au tri topologique.

Exemple 3.24

On considère le même graphe qu'à l'exemple 3.23 et on lui applique un tri topologique.



En considérant les sommets un par un et de gauche à droite, il est facile de déterminer le poids d'un chemin maximal terminant à ce sommet.

- Sommet A : aucun arc entrant, le poids d'un chemin maximal qui termine en A est donc 0.
- Sommet C : un seul arc entrant, le chemin maximal qui termine en C est donc A – C pour un poids de 5.
- Sommet B : deux arcs entrants. Il y a donc deux possibilités, soit faire A – B avec un poids de 3, soit se rendre à C avec un chemin de poids maximum puis prendre l'arc C – B pour un poids de $5 + 1 = 6$. Le chemin maximal qui termine en B est donc A – C – B pour un poids de 6.
- Sommet D : un seul arc entrant, le chemin maximal qui termine en D consiste donc à prendre le chemin maximal qui termine en B et y ajouter l'arc B – D pour un poids total de $6 + 2 = 8$.
- Sommet E : deux arcs entrants. Les deux possibilités sont : se rendre à C avec un chemin maximal et ajouter l'arc C – E pour un poids de $5 + 6 = 11$ ou se rendre à D avec un chemin maximal et ajouter l'arc D – E pour un poids de $8 + 2 = 10$.

On conclut que le chemin de poids maximal dans ce graphe orienté acyclique est $\mathsf{A} - \mathsf{C} - \mathsf{E}$ pour un poids de 11.

L'algorithme **CheminCritique** décrit la méthode illustrée à l'exemple 3.24 pour le calcul du chemin de poids maximal dans un graphe.

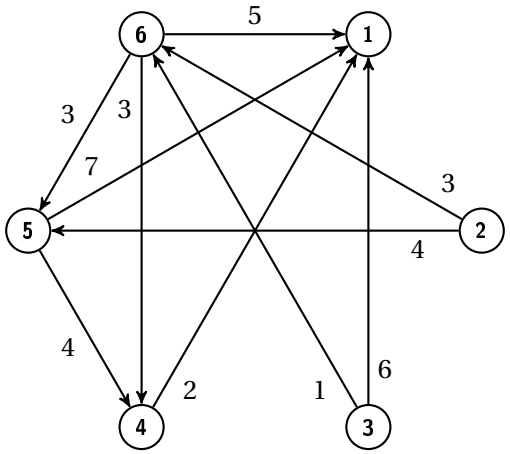
Algorithme 5 `CheminCritique`

Entrées: $G = (V, E)$ un graphe orienté pondéré acyclique et $\omega : E \longrightarrow \mathbb{R}^+$ la fonction de pondération
Sortie: Poids maximal d'un chemin dans G

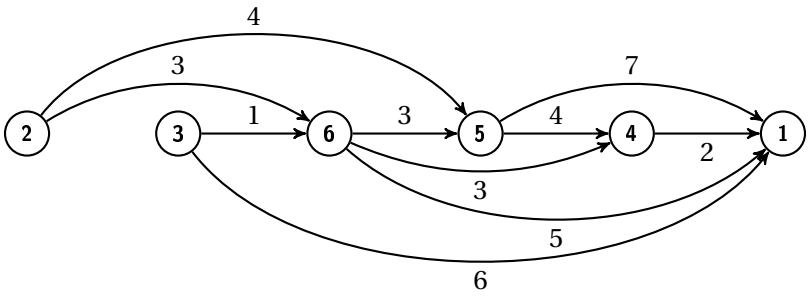
1: $T :=$ tri topologique des sommets de G
2: $L :=$ tableau associatif ▷ $L(\mathbf{v})$ est le poids d'un chemin maximal terminant en $\mathbf{v}$
3: **pour tout** $\mathbf{v} \in V$ dans l'ordre de T **faire**
4: **si** $\mathbf{v}$ possède au moins un arc entrant **alors**
5: $L(\mathbf{v}) := \max\{L(\mathbf{u} + \omega(\mathbf{u}, \mathbf{v})) \mid \mathbf{u} \in V \text{ et } \mathbf{u} \rightarrow \mathbf{v} \in E\}$
6: **sinon**
7: $L(\mathbf{v}) := 0$
8: **fin si**
9: **fin pour**
10: **retourner** $\max\{L(\mathbf{v}) \mid \mathbf{v} \in V\}$

Exemple 3.25

Effectuez une trace de l'algorithme **CheminCritique** dans le graphe G ci-dessous afin de déterminer le poids d'un chemin maximal.



Solution :
On commence par utiliser l'algorithme de Kahn afin de calculer un tri topologique. Cette étape est indépendante de la fonction de pondération. On obtient le tri topologique: [**2**, **3**, **6**, **5**, **4**, **1**]. On retrace le graphe en positionnant les sommets dans cet ordre.



- Initialisation :

$T = [\mathbf{2}, \mathbf{3}, \mathbf{6}, \mathbf{5}, \mathbf{4}, \mathbf{1}]$, $L =$

1	2	3	4	5	6

- Itération #1 : $\mathbf{v} = \mathbf{2}$.
Le sommet **2** n'a aucun arc entrant, le poids d'un chemin maximal terminal en **2** est zéro.

$L =$

1	2	3	4	5	6
	0				

- Itération #2 : $\mathbf{v} = \mathbf{3}$
Le sommet **3** n'a aucun arc entrant, le poids d'un chemin maximal terminal en **3** est zéro.

$L =$

1	2	3	4	5	6
	0	0			

- Itération #3 : $\mathbf{v} = \mathbf{6}$
Le sommet **6** possède deux arcs entrant, l'un depuis **2**, l'autre depuis **3**.
  - Pour $\mathbf{u} = \mathbf{2}$: $L(\mathbf{2}) + \omega(\mathbf{2},\mathbf{6}) = 0 + 3 = 3$
  - Pour $\mathbf{u} = \mathbf{3}$: $L(\mathbf{3}) + \omega(\mathbf{3},\mathbf{6}) = 0 + 1 = 1$

  On affecte la valeur 3 à $L(\mathbf{6})$.

$L =$

1	2	3	4	5	6
	0	0			3

- Itération #4 : $\mathbf{v} = \mathbf{5}$
Le sommet **5** possède deux arcs entrant, l'un depuis **2**, l'autre depuis **6**.
  - Pour $\mathbf{u} = \mathbf{2}$: $L(\mathbf{2}) + \omega(\mathbf{2},\mathbf{5}) = 0 + 4 = 4$
  - Pour $\mathbf{u} = \mathbf{6}$: $L(\mathbf{6}) + \omega(\mathbf{6},\mathbf{5}) = 3 + 3 = 6$

  On affecte la valeur 6 à $L(\mathbf{5})$.

$L =$

1	2	3	4	5	6
	0	0		6	3

- Itération #5 : $\mathbf{v} = \mathbf{4}$
  Le sommet **4** possède deux arcs entrant, l'un depuis **6**, l'autre depuis **5**.
  - Pour $\mathbf{u} = \mathbf{6}$: $L(\mathbf{6}) + \omega(\mathbf{6},\mathbf{4}) = 3 + 4 = 7$
  - Pour $\mathbf{u} = \mathbf{5}$: $L(\mathbf{5}) + \omega(\mathbf{5},\mathbf{4}) = 6 + 4 = 10$

  On affecte la valeur 10 à $L(\mathbf{4})$.

$L =$

1	2	3	4	5	6
	0	0	10	6	3

- Itération #6 : $\mathbf{v} = \mathbf{1}$
  Le sommet **1** possède quatre arcs entrant, depuis **3**, **6**, **5** et **4**.
  - Pour $\mathbf{u} = \mathbf{3}$: $L(\mathbf{3}) + \omega(\mathbf{3},\mathbf{1}) = 0 + 6 = 6$
  - Pour $\mathbf{u} = \mathbf{6}$: $L(\mathbf{6}) + \omega(\mathbf{6},\mathbf{1}) = 3 + 5 = 8$
  - Pour $\mathbf{u} = \mathbf{5}$: $L(\mathbf{5}) + \omega(\mathbf{5},\mathbf{1}) = 6 + 7 = 13$
  - Pour $\mathbf{u} = \mathbf{4}$: $L(\mathbf{4}) + \omega(\mathbf{4},\mathbf{1}) = 10 + 2 = 12$

  On affecte la valeur 13 à $L(\mathbf{1})$.

$L =$

1	2	3	4	5	6
13	0	0	10	6	3

La boucle principale est maintenant terminée. La dernière étape consiste à déterminer la valeur maximum présente dans le tableau L. Dans cet exemple, il s'agit de 13, ce qui correspond au poids maximum d'un chemin dans ce graphe.

Exemple 3.26

Durée minimale d'un projet

Le tableau suivant liste les étapes de la constructions d'une maison.

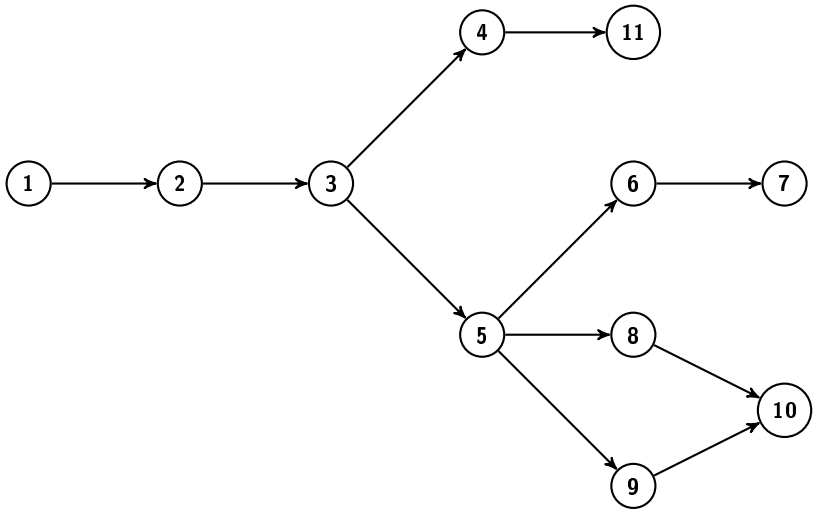
#	Description de la tâche	Durée (j)	Tâche(s) préalable(s)
1	Terrassement	5	
2	Fondations	3	**1**
3	Élévation des murs	2	**2**
4	Revêtement extérieur	3	**3**
5	Cloisonnement	2	**3**
6	Carrelage	3	**5**
7	Plomberie	2	**6**
8	Électricité	1	**5**
9	Isolation	1	**5**
10	Peinture	2	**8,9**
11	Éclairage extérieur	1	**4**

Déterminez la durée minimale requise pour compléter ce projet.

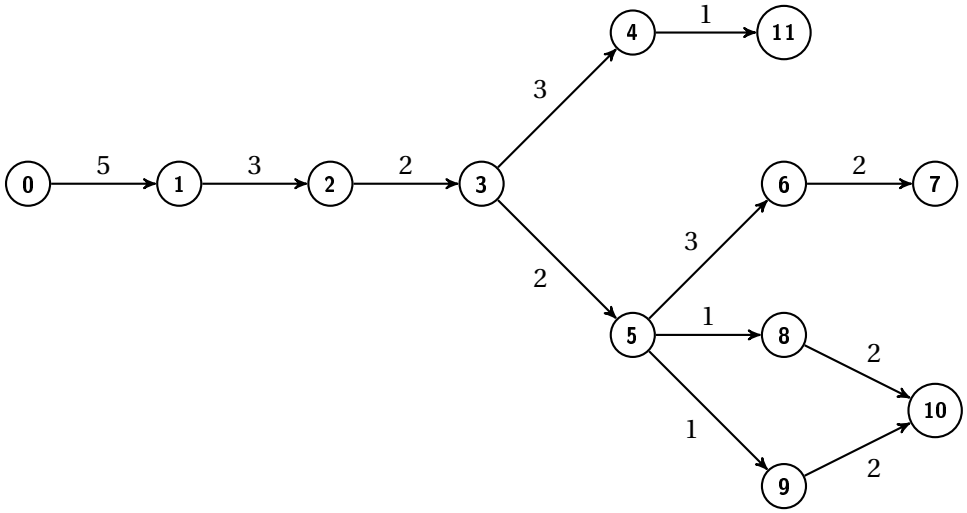
Solution :

Afin de déterminer une borne inférieure à la durée de ce projet, on fait l'hypothèse que si deux tâche n'ont pas de relation de dépendance l'une envers l'autre, alors elles seront réalisées simultanément. Par exemple, une fois la tâche « *5. Cloisonnement* » complétée, on suppose que les tâches « *8. Électricité* » et « *9. Isolation* » seront effectuées simultanément par deux équipes distinctes.

1. Construire le graphe de dépendance.
   On construit le graphe orienté G dont les sommets correspondent aux étapes du projet et les arcs décrivent les relations de dépendances.



2. Pondérer le graphe.
   Le graphe G est pondéré de sorte qu'une arête $\mathbf{u}-\mathbf{v}$ a comme poids la durée de la tâche v. De plus, afin de tenir compte de la durée des tâches n'ayant pas de tâche préalable, on crée un sommet $\mathbf{0}$ qui ne correspond à aucune tâche. Ce sommet peut être vu comme le point de départ du projet. Ensuite, pour chaque sommet u, autre que $\mathbf{0}$ n'ayant pas d'arc entrant, on ajout un arc $\mathbf{0}-\mathbf{u}$ avec comme pondération la durée de la tâche $\mathbf{u}$.



3. Calcul d'un chemin de poids maximal.
   On utilise l'algorithme du **Chemin critique** afin de déterminer le poids d'un chemin de pondération maximum dans ce graphe. Le poids de ce chemin correspond à la durée minimale du projet.
   Lorsque l'algorithme termine, l'état du tableau L est :

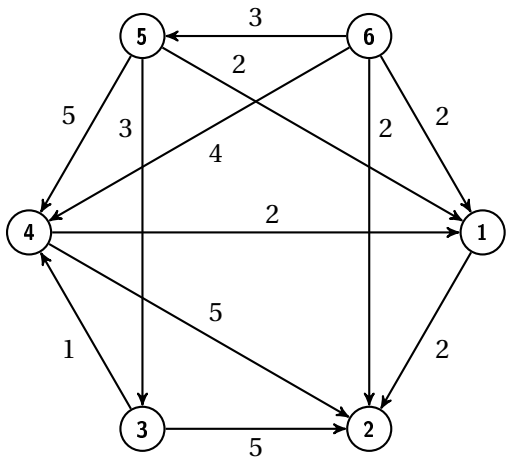
	0	1	2	3	4	5	6	7	8	9	10	11
$L =$	0	5	8	10	13	12	15	17	13	13	15	14

   La valeur retournée par l'algorithme est 17. On conclut que la durée minimale du projet est de 17 jours.

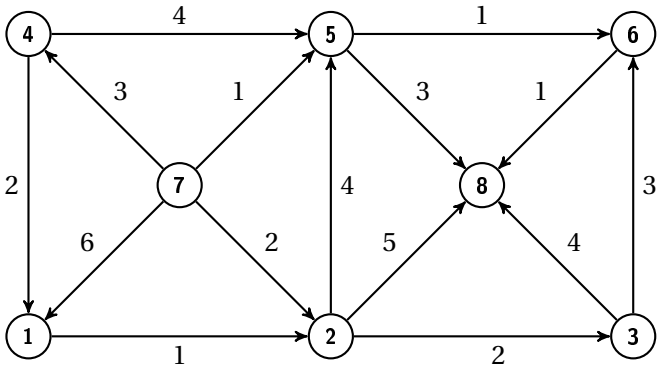
Exercices

3.11 Pour chacun des graphes suivants, utilisez l'algorithme **CheminCritique** afin de détermine le poids d'un chemin de pondération maximale. À chaque fois, spécifiez le tri topologique, l'état final du tableau L ainsi que le poids du chemin maximal.

(a) Le graphe G_1 :



(b) Le graphe G_2 :



3.12 Le tableau suivant liste les étapes nécessaires à la réalisation d'un projet. Déterminez la durée minimale que peut prendre la réalisation de toutes les tâches de ce projet.

Tâche	Durée (h)	Tâche(s) préalable(s)
1	9	**3,4,5**
2	2	**3,4,5**
3	3	
4	7	**6**
5	5	**6**
6	3	
7	6	**4,5**
8	4	**1,2,7**

Chapitre 4

Arbres

4.1 Définitions de base

Rappelons d'abord quelques définitions de théorie des graphes qui permettent de définir la notion d'arbre.

- On appelle **arêtes** les liaisons entre les sommets d'un graphe non orienté (définition 3.1).
- On appelle **arcs** les liaisons entre les sommets d'un graphe orienté (définition 3.2).
- On appelle **circuit** ou **cycle** un chemin qui commence et se termine au même sommet du graphe. Un cycle est dit **simple** s'il ne contient pas une même arête plus d'une fois (définition 3.5).

Définition 4.1 : Graphe acyclique

Un graphe $G = (V, E)$ est dit **acyclique** si, et seulement si, il ne comporte aucun cycle simple.

En d'autres mots, un graphe est dit acyclique s'il est impossible de trouver un chemin partant d'un sommet et y revenant sans passer deux fois par la même arête (ou le même arc).

Définition 4.2 : Arbre, feuille

Un graphe $G = (V, E)$ est un **arbre** si, et seulement si, il est non orienté, connexe et acyclique. Les sommets de degré 1 d'un arbre sont appelés **feuilles**. Un arbre est dit **fini** si son ensemble de sommets est de cardinalité finie.

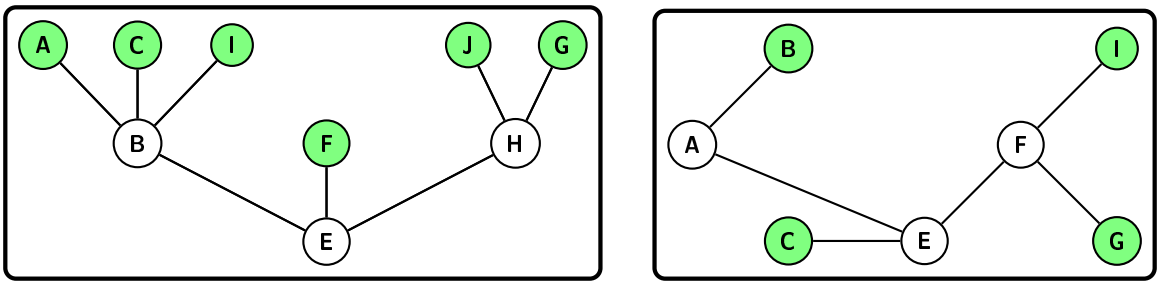
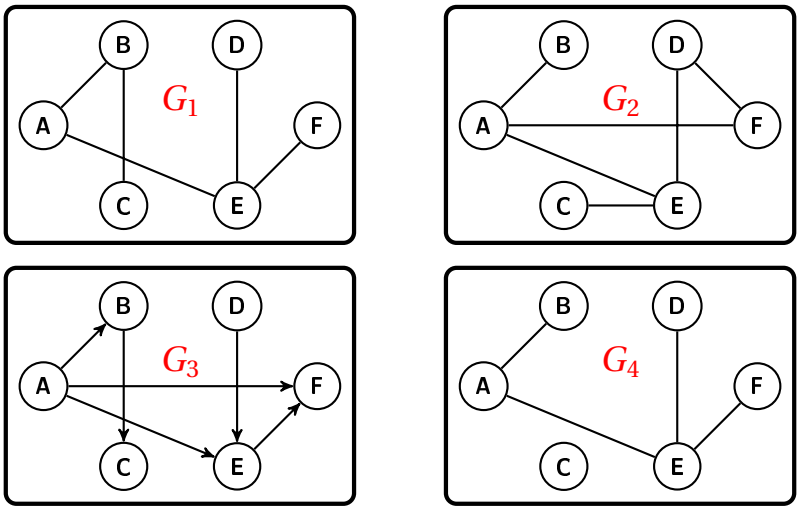


Figure 4.1 Deux exemples d'arbres dont les feuilles sont coloriées.

Exemple 4.1

Pour chacun des quatre graphes suivants, dites s'il est acyclique et s'il s'agit d'un arbre. S'il est un arbre, énumérez ses feuilles.



Solution :

G_1 est un graphe acyclique. De plus, il est et non orienté et connexe; c'est donc un arbre. Ses feuilles sont **C**, **D** et **F**.

G_2 n'est pas acyclique car il possède au moins un cycle simple: **A**−**F**−**D**−**E**−**A**. G_2 n'est donc pas un arbre.

G_3 est un graphe acyclique, mais il est orienté; ce n'est donc pas un arbre.

G_4 est un graphe acyclique et non orienté, mais ce n'est pas un arbre car il n'est pas connexe (par exemple, on ne peut pas trouver de chemin allant de **A** à **C**).

Si on choisit un sommet **r** quelconque dans un arbre $G = (V, E)$, il est possible d'enraciner l'arbre en **r**, c'est-à-dire d'orienter toutes les arêtes de sorte que, pour chaque sommet $\mathbf{v} \in V$, il existe un chemin allant de **r** à **v**.

Définition 4.3 : Arbre enraciné, racine, parent, enfant

Un **arbre enraciné en r** est un arbre $G = (V, E)$ où **r** est un sommet de G appelé **racine**.

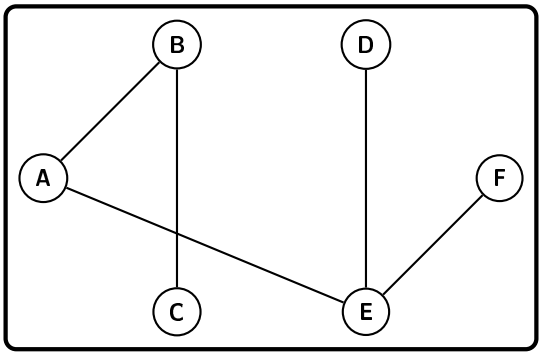
Les chemins allant de la racine **r** vers les feuilles induisent une orientation (on tracera ou non les flèches des arcs ainsi définis).

Chaque sommet $\mathbf{v} \in V$, sauf la racine, a un unique **parent p** : il s'agit du sommet précédent dans l'unique chemin allant de **r** à **v**. On dit alors que **v** est l'**enfant** de **p**.

Exemple 4.2

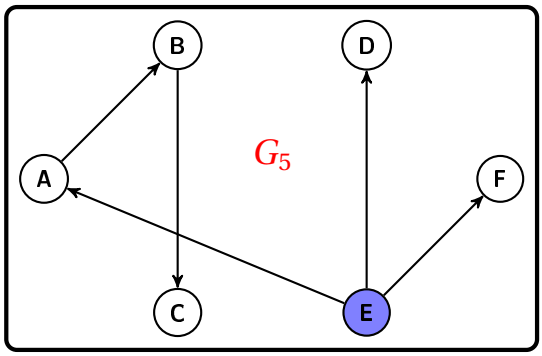
Enracinez l'arbre suivant en **E** puis en **A** afin de produire deux arbres enracinés. Coloriez la racine pour l'identifier.

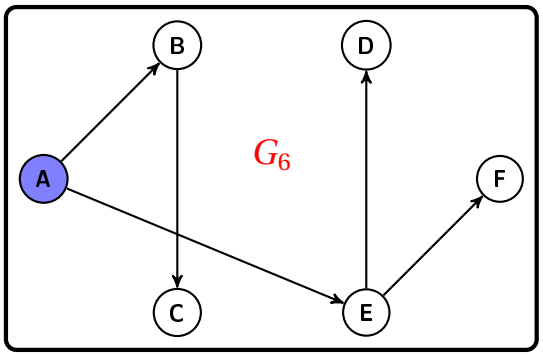
Redessinez ensuite les arbres enracinés de sorte que chaque parent soit situé plus haut que ses enfants, comme dans un arbre généalogique. Déterminez qui est le parent de **F** dans chacun des graphes enracinés.

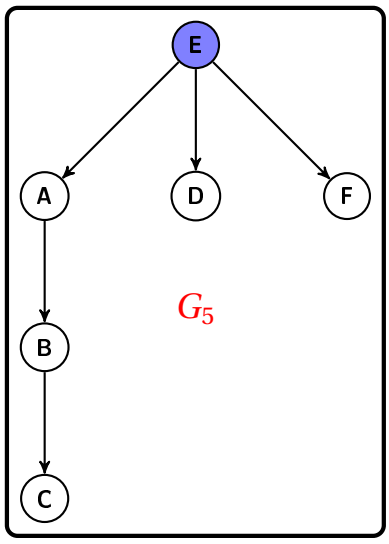


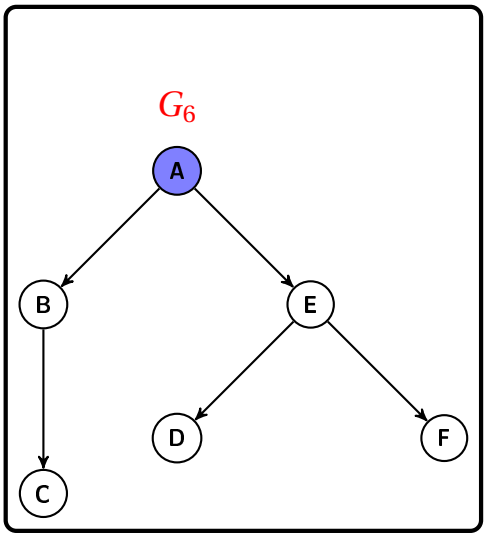
Solution :

G_5 est l'arbre enraciné en **E**. G_6 est l'arbre enraciné en **A**. Dans les deux cas, le parent de **F** est **E**.









Maths VS gestion, des définitions qui peuvent différer!

En gestion, on utilise souvent le concept **d'arbre de décision**. À la différence du concept d'arbre enraciné de la théorie des graphes, le concept d'arbre de décision utilisé en gestion permet qu'un enfant ait plus d'un parent. Par exemple, l'enfant indiqué par un astérisque dans la figure 4.3 a 4 parents; ce n'est donc pas un arbre enraciné selon notre définition. Par contre, l'arbre de décision de la figure 4.2, qui permet de décider si une opération déneigement sera déclenchée, est un arbre enraciné.

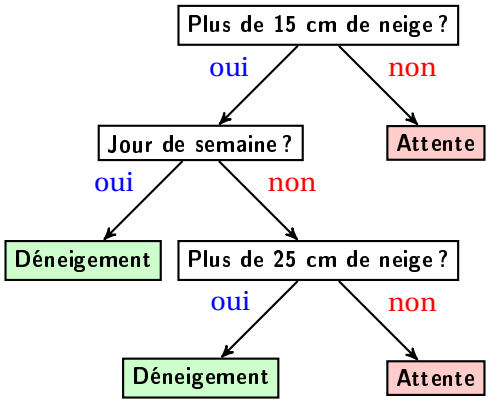


Figure 4.2 Exemple d'arbre de décision pour le déclenchement d'une opération de déneignement.

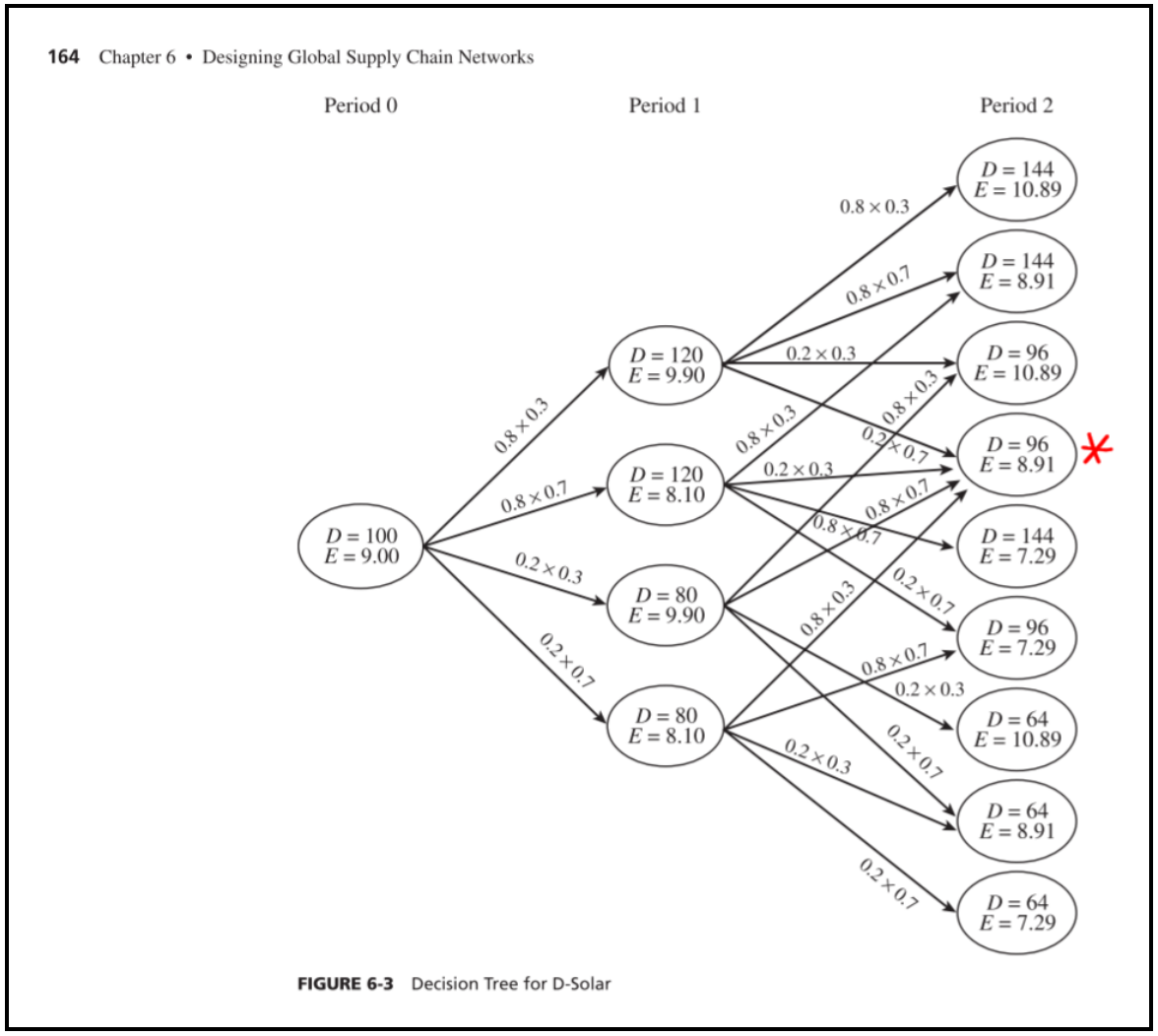


Figure 4.3 Exemple « d'arbre de décision » qui ne respecte pas notre définition d'arbre enraciné, ni celle d'arbre, tiré du livre Designing Global Supply Chain Networks, Fifth Edition, Sunil chopra et Peter Meindl.

Les arbres enracinés peuvent servir à expliciter l'ensemble des possibilités d'une situation donnée de façon systématique, afin de compter ce nombre de possibilités, de les analyser, ou pour choisir la plus avantageuse.

Exemple 4.3

Anouk et Xavier jouent quelques parties de leur jeu favori, un jeu qui n'admet pas de partie nulle. Le grand gagnant est celui qui gagne deux parties de suite ou un total de trois parties.

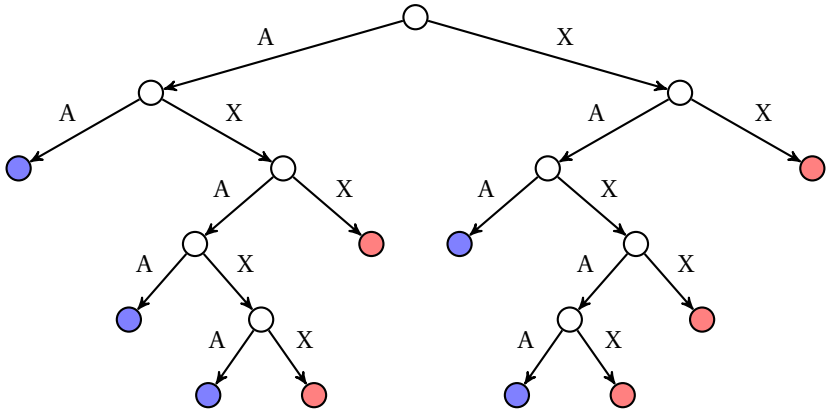
(a) Tracez un arbre enraciné qui illustre toutes les possibilités de victoires.

(b) Combien y a-t-il de possibilités ?

(c) Quel est le nombre maximal de parties qui sera joué ?

Solution :

(a)



(b) On voit sur l'arbre qu'il y a 10 possibilités pour le déroulement des parties: 5 où Anouk est gagnante (feuilles bleues) et 5 où Xavier est gagnant (feuilles rouges).

(c) Le nombre maximal de parties jouées sera 5.

4.1.1 Exemple: arbres syntaxiques

Considérons l'expression mathématique:

$$\sin(x+y)+2^{-a}+a^{-2}+\ln\left(|x-y|\right).$$

Cette expression est formée de plusieurs éléments:

- Des nombres: 2 et -2.
- Des variables: x, y et a.
- Des opérateurs: sin, ln, +, −, | |, $\widehat{}$.

(Note: on utilise le symbole $\widehat{}$ pour représenter l'exponentielle.)

Une telle expression peut être représentée par un arbre enraciné comme celui de la figure 4.4.

Remarques:

- Plus précisément, le schéma représente l'expression

$$\sin(x+y)+\left(2^{-a}+\left(a^{-2}+\ln\left(|x-y|\right)\right)\right).$$

  L'associativité de l'addition nous permet d'enlever les parenthèses.
- Les 4 opérateurs unaires sont identifiés en jaune. Ils sont faciles à reconnaître puisqu'ils sont appliqués à une seule sous-expression. La fonction récursive devra donc retourner le nombre 4 si elle reçoit l'expression $\sin(x+y)+2^{-a}+a^{-2}+\ln\left(|x-y|\right)$ en entrée.
- Attention: le symbole « − » est utilisé pour différents usages (il y a d'ailleurs souvent deux touches « moins » sur une calculatrice).
  - Dans l'expression $x-y$, le symbole « − » désigne la soustraction, un opérateur binaire.
  - Dans l'expression $-a$, le symbole « − » désigne l'opérateur unaire « opposé » et il est appliqué à la variable a.
  - L'expression -2 désigne simplement le nombre $-2 \in \mathbb{R}$ (sans opérateur).

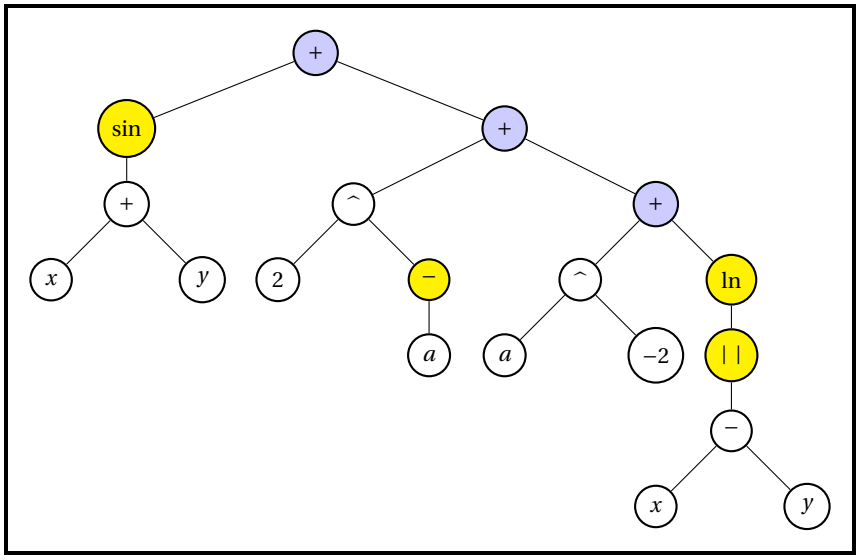


Figure 4.4 Arbre syntaxique représentant l'expression $\sin(x+y)+\left(2^{-a}+\left(a^{-2}+\ln\left(|x-y|\right)\right)\right)$

- Les règles de priorité des opérations font en sorte que les trois opérateurs identifiés en bleu sont tous au même niveau. Le fait d'avoir placé le premier opérateur + au sommet de la hiérarchie est purement arbitraire. On pourrait commencer avec le second + ou le troisième. Le schéma serait différent, mais l'expression représentée serait égale.

4.2 Quelques théorèmes sur les arbres

Théorème 4.1

Tout arbre fini ayant plus d'un sommet possède au moins une feuille, c'est-à-dire au moins un sommet de degré 1.

▷ ***Démonstration*** En guise de démonstration, décrivons un processus pour trouver une feuille d'un arbre T possédant au moins 2 sommets. Ce processus est illustré à la figure 4.5. Choisissons un sommet $\mathbf{v}_1$ au hasard. Si ce sommet $\mathbf{v}_1$ est de degré 1, alors c'est une feuille et la recherche est terminée. Sinon, le sommet $\mathbf{v}_1$ est incident à au moins 2 arêtes: choisissons-en une et notons-la e_1, puis notons $\mathbf{v}_2$ l'autre sommet adjacent à e_1. Si $\mathbf{v}_2$ est de degré 1, alors la recherche est terminée. Sinon, continuons le chemin: notons e_2 une arête incidente à $\mathbf{v}_2$ qui est différente de e_1, et $\mathbf{v}_3$ le sommet adjacent à e_2. Et ainsi de suite. Le chemin $e_1, e_2, e_4, \ldots, e_m$ ainsi créé ne repasse jamais par le même sommet puisque T est acyclique par définition. Et puisque le nombre de sommets de T est fini, le processus de création du chemin doit s'arrêter. Le sommet terminal de la dernière arête est forcément un sommet de degré 1 (sinon le processus de création du chemin aurait continué). ◁

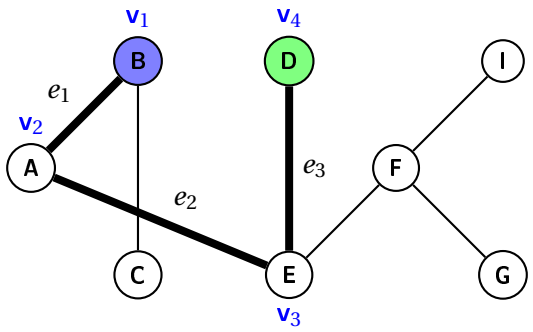


Figure 4.5 Recherche d'une feuille à partir du sommet **B**. Ici, le processus génère le chemin $e_1, e_2, e_3 =$ **B**−**A**−**E**−**D** qui se termine au sommet **D**. Le sommet **D** est bel et bien une feuille de l'arbre.

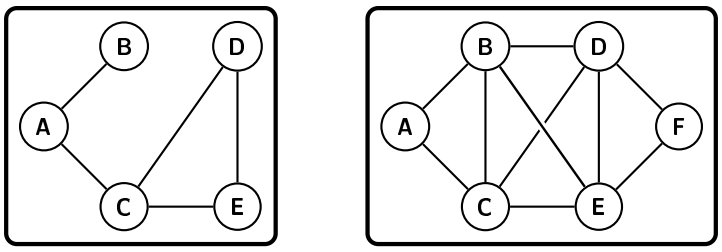
Définition 4.4 : Sous-graphe

Soit $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ deux graphes orientés ou deux graphes non orientés. On dit que G_1 est un **sous graphe** de G_2 si et seulement si l'ensemble des sommets de G_1 est un sous-ensemble de celui de G_2 et que l'ensemble des arêtes ou arcs de G_1 est un sous-ensemble de celui de G_2. On notera ceci $G_1 \subseteq G_2$, ou $G_2 \supseteq G_1$. Autrement dit,

$$G_1 \subseteq G_2 \iff V_1 \subseteq V_2 \quad \text{et} \quad E_1 \subseteq E_2.$$

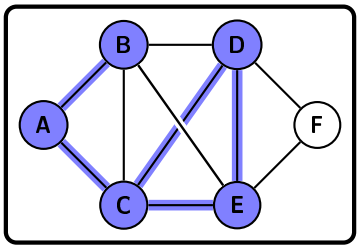
Exemple 4.4

Le graphe de gauche est-il un sous-graphe de celui de droite ?



Solution :

Oui. On peut aussi illustrer le graphe et son sous-graphe en une seule figure, en coloriant les arêtes et sommets du sous-graphe.



Le lemme suivant stipule que lorsqu'on retire une feuille à un arbre, alors le graphe obtenu est lui aussi un arbre (voir figure 4.6).

Lemme 4.1

Soient $G = (V, E)$ un arbre à $n \geq 2$ sommets, $\mathbf{u} \in V$ une feuille de G et e l'unique arête incidente à $\mathbf{u}$. Le sous-graphe $G' = (V \setminus \{\mathbf{u}\}, E \setminus \{e\})$ est aussi un arbre.

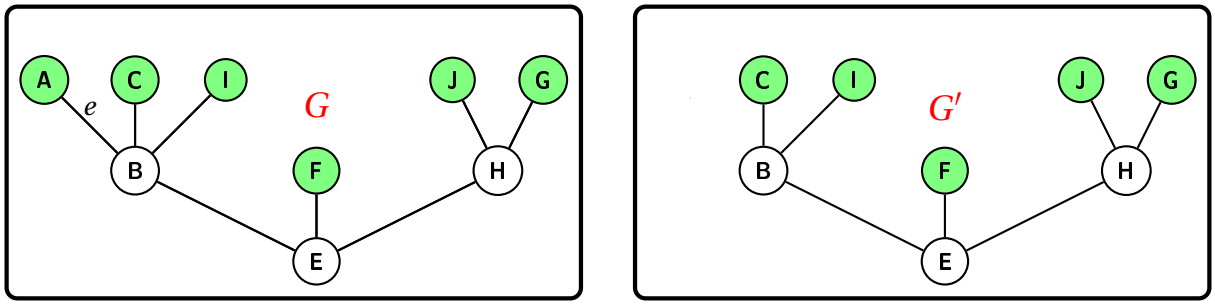


Figure 4.6 Illustration du lemme 4.1. Si on retire la feuille **A** et l'arête e de l'arbre G, le sous-graphe obtenu est lui aussi un arbre. On le note $G' = (V \setminus \{\mathbf{A}\}, E \setminus \{e\})$.

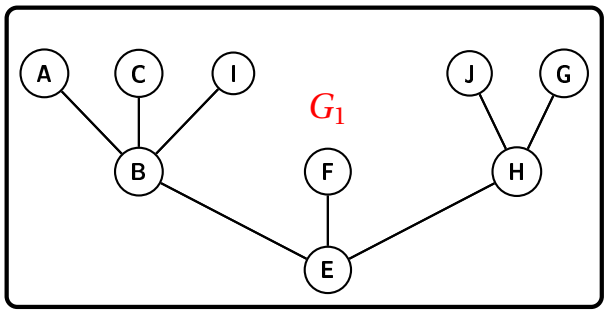
▷ ***Démonstration*** Tout d'abord, le théorème 4.1 garantit l'existence d'une feuille dans le graphe G. Il reste à montrer que le graphe $G' = (V \setminus \{\mathbf{u}\}, E \setminus \{e\})$ est acyclique et connexe.

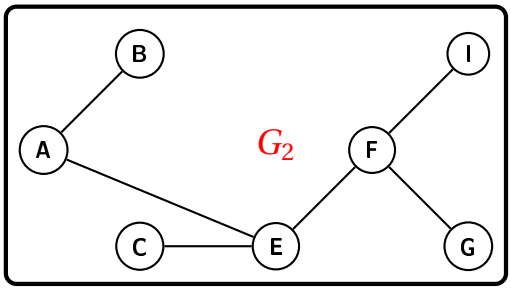
On montre que G' est acyclique par contradiction. Supposons qu'il existe un cycle simple dans G'. Comme tous les sommets et toutes les arêtes de G' sont aussi dans G, le cycle est donc présent dans le graphe G. Il y a contradiction puisque l'énoncé du théorème stipule que G est acyclique.

On montre que G' est connexe par une preuve directe. Soient $\mathbf{s}_1$ et $\mathbf{s}_2$ deux sommets quelconques de G'. Ces deux sommets sont aussi des sommets de G et comme G est connexe, il existe un chemin simple de $\mathbf{s}_1$ à $\mathbf{s}_2$ dans G, on appelle ce chemin C. Pour montrer que le chemin C est aussi présent dans le graphe G', il suffit de montrer qu'il ne passe pas par le sommet $\mathbf{u}$ ni par l'arête e. Comme le sommet $\mathbf{u}$ est une feuille, il ne peut pas être un sommet intermédiaire dans un chemin simple. De plus, comme $\mathbf{s}_1$ et $\mathbf{s}_2$ sont des sommets de G', on a $\mathbf{s}_1 \neq \mathbf{u}$ et $\mathbf{s}_2 \neq \mathbf{u}$. Le sommet $\mathbf{u}$ n'est donc ni le point de départ, ni le point d'arrivée, ni un sommet intermédiaire du chemin C. Il ne fait donc pas partie du chemin C. Finalement, comme l'arête e est adjacente à $\mathbf{u}$, elle ne peut pas faire partie du chemin C. ◁

Exemple 4.5

Pour chacun des deux arbres suivants, comptez le nombre de sommets et le nombre d'arêtes.





Solution :

L'arbre G_1 possède 9 sommets et 8 arêtes.

L'arbre G_2 possède 7 sommets et 6 arêtes.

À l'exemple précédent, dans les deux arbres observés, la différence entre le nombre de sommets et le nombre d'arêtes d'un arbre est de 1. Est-ce un hasard ou une propriété commune à tous les arbres ? Le théorème suivant répond à cette question.

Théorème 4.2

Pour tout nombre $n \in \mathbb{N}^*$, tout arbre ayant n sommets possède exactement $n-1$ arêtes.

Autrement dit, si $G = (V, E)$ est un arbre ayant au moins 1 sommet, alors

$$|V| - 1 = |E|.$$

▷ ***Démonstration*** Cette preuve repose sur l'observation suivante : un graphe à un seul sommet ne possède aucune arête. Cette observation montre déjà que le théorème est vrai pour $n = 1$.

Soit G un graphe à n sommets, avec $n \geq 2$. Le théorème 4.1 garanti que G possède au moins une feuille. De plus, le lemme 4.1 stipule que le sous-graphe obtenu en retirant une feuille de G est lui aussi un arbre.

Soient $\mathbf{u}_1$ une feuille de G et G_1 le sous-graphe obtenu en retirant cette feuille et son unique arête de G. Comme G_1 est un arbre, s'il possède au moins deux sommets, alors on peut répéter le processus et définir G_2 un sous-graphe de G_1 obtenu en lui retirant une feuille. En répétant ce processus tant que le sous-graphe obtenu possède au moins deux sommets, on génère ainsi une suite de sous-graphes :

$$G \supseteq G_1 \supseteq G_2 \supseteq \ldots \supseteq G_{n-1}.$$

Dans cette suite, chaque G_i est un arbre obtenu en retirant une feuille au graphe précédent. Cette construction impose donc que le graphe G_i possède exactement i sommets et i arêtes de moins que le graphe G. En particulier, le graphe G_{n-1} possède exactement 1 sommet et donc aucune arête.

Finalement, comme le graphe G_{n-1} a été obtenu à partir du graphe G en lui retirant exactement $n-1$ sommets et $n-1$ arêtes. Il en découle que le nombre d'arêtes dans le graphe G est exactement $n-1$. ◁

Théorème 4.3

Soit $G = (V, E)$ un graphe connexe et C un circuit simple de G.

Si une des arêtes du circuit C est retirée, alors le sous-graphe obtenu demeure connexe.

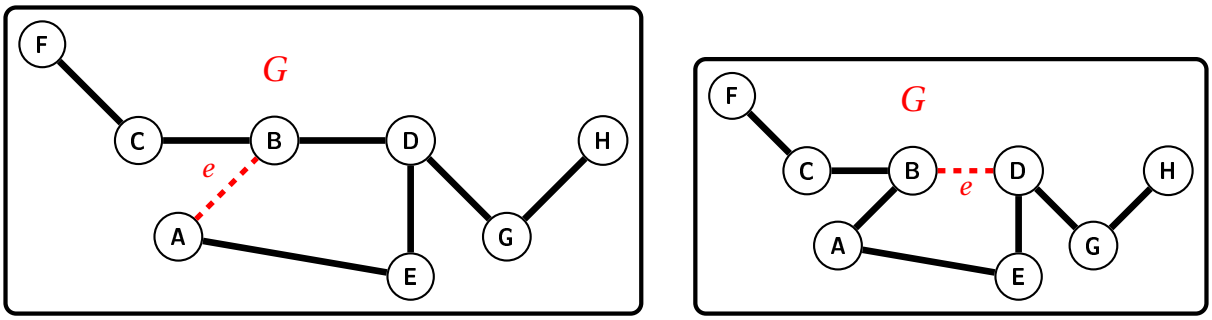


Figure 4.7 Illustration du théorème 4.3. Le graphe connexe G possède le circuit simple **A**–**B**–**D**–**E**–**A**. Si une arête e est retirée du circuit, alors le sous-graphe obtenu demeure connexe.

▷ ***Démonstration*** Soit $G = (V, E)$ un graphe connexe, C un circuit simple de G et e une arête du circuit C. Soit $G_1 = (V_1, E_1)$ le sous-graphe de G obtenu en retirant l'arête e :

$$V_1 = V \qquad \text{et} \qquad E_1 = E - \{e\}.$$

On doit montrer que le graphe G_1 est connexe lui aussi, c'est-à-dire que, pour toute paire de sommets s_1 et s_2, il existe un chemin allant de l'un à l'autre par des arêtes de E_1.

Soit donc s_1 et s_2 deux sommets du graphe $G_1 = (V_1, E_1)$. Puisque $V_1 = V$, les sommets appartiennent aussi au graphe G. Puisque G est connexe, il existe un chemin dans G allant de s_1 à s_2 : $e_1, e_2, e_3, \ldots, e_n$, où $e_i \in E$.

Si toutes les arêtes e_i du chemin sont dans E_1, alors l'objectif est atteint.

Sinon, c'est que le chemin passe par l'arête e. Il faut donc trouver un chemin alternatif reliant les sommets s_1 et s_2 qui ne passe pas par e. Ceci est toujours possible, car e fait partie du circuit. Notons v_1 et v_2 les sommets reliés par e dans l'ordre de parcours du chemin. Si l'arête e est retirée du graphe G et que l'on note G_1 le sous-graphe obtenu, les sommets v_1 et v_2 pourront encore être reliés en empruntant les autres arêtes du circuit.

Ainsi, en demeurant toujours dans le sous-graphe G_1 tel qu'illustré à la figure 4.8, on a un chemin qui va de s_1 à v_1, un chemin alternatif qui va de v_1 à v_2 et un chemin qui va de v_2 à s_2. On a donc un chemin allant de s_1 à s_2 et l'objectif est atteint.

◁

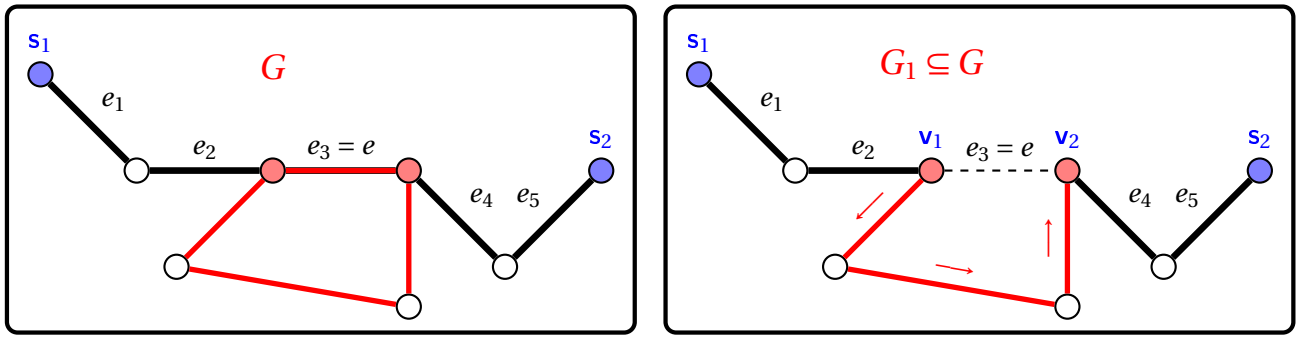


Figure 4.8 Illustration de la preuve du théorème 4.3.

4.3 Arbre couvrant d'un graphe

Définition 4.5 : Arbre couvrant

Soit $G_2 = (V_2, E_2)$ un graphe et $G_1 = (V_1, E_1)$ un sous graphe de G_2. On dit que G_1 est un **arbre couvrant** de G_2 si et seulement si G_1 est un arbre et l'ensemble des sommets de G_1 est égal à celui de G_2 : $V_1 = V_2$. Voir figure 4.9 et 4.10.

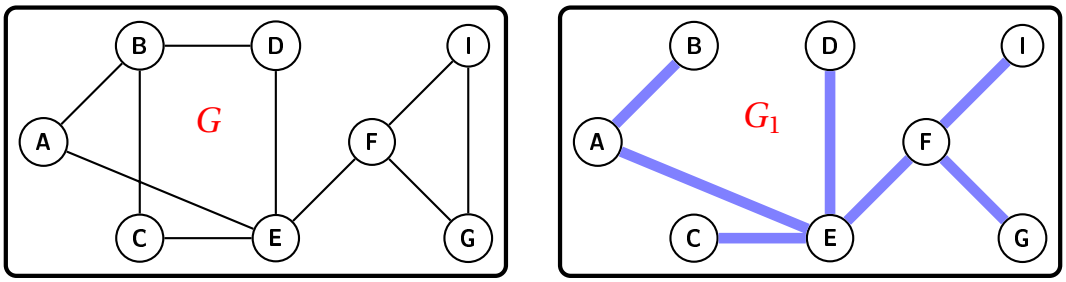


Figure 4.9 Graphe connexe G et un de ses arbres couvrants G_1.

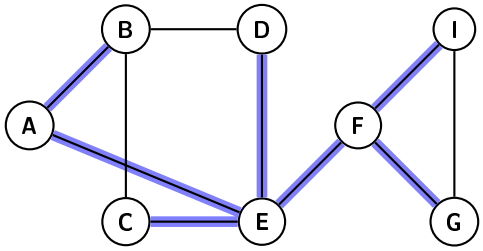


Figure 4.10 Graphe connexe et arbre couvrant (en gras).

Théorème 4.4

Pour tout nombre $n \in \mathbb{N}^*$, tout graphe connexe ayant n sommets et $n-1$ arêtes est un arbre.

Autrement dit, si $G = (V, E)$ est un graphe connexe ayant au moins 1 sommet et si

$$|V| - 1 = |E|,$$

alors G est un arbre.

▷ ***Démonstration*** Soit $G = (V, E)$ un graphe connexe ayant au moins 1 sommet et tel que $|V| - 1 = |E|$. On doit montrer que G est un arbre.

Si G a un seul sommet, alors G ne possède aucune arête car $|E| = |V| - 1 = 1 - 1$ (voir figure 4.11). Le graphe G est donc connexe et acyclique : c'est un arbre.

Figure 4.11 Graphe ayant un seul sommet et aucune arête : $|V| = 1$ et $|E| = 0$.

Procédons par contradiction pour le cas où G possède au moins 2 sommets : supposons que G n'est pas un arbre et montrons que cela entraîne une contradiction.

Hypothèse : en plus d'être connexe et tel que $|V| - 1 = |E|$, le graphe G **possède au moins 2 sommets et n'est pas un arbre.**

Si le graphe connexe G n'est pas un arbre, cela implique qu'il possède au moins un circuit simple. Retirons n'importe quelle arête de ce circuit : le sous-graphe obtenu, notons-le $G_1 = (V_1, E_1)$, est encore connexe (par théorème 4.3). Voir figure 4.12. De plus, on a

$$V_1 = V \quad \text{et} \quad |E_1| = |E| - 1.$$

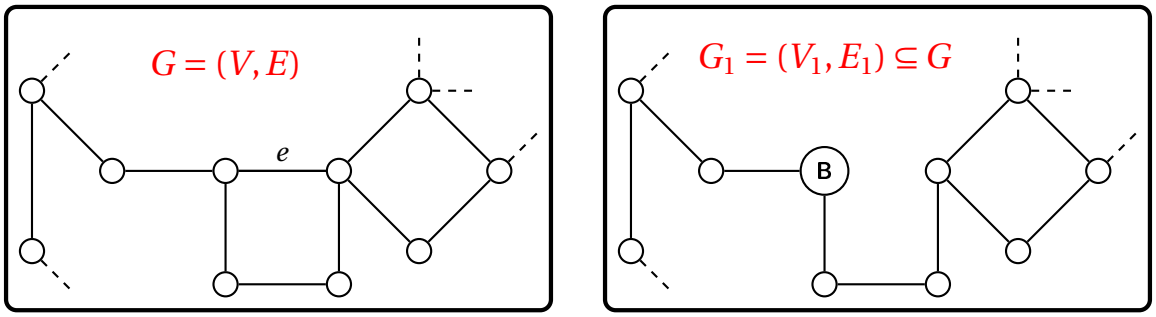


Figure 4.12 Portions des graphes connexes G et G_1 de la preuve du théorème 4.4.

Si G_1 n'est pas un arbre, cela implique qu'il possède au moins un circuit simple et que nous pouvons encore lui retirer une arête pour obtenir un sous-graphe connexe $G_2 = (V_2, E_2)$. On a

$$V_2 = V \quad \text{et} \quad |E_2| = |E_1| - 1 = |E| - 2.$$

Continuons ainsi à retirer des arêtes tant qu'il reste des cycles simples, pour générer une suite de sous-graphes :

$$G \supseteq G_1 \supseteq G_2 \supseteq \ldots \supseteq G_m.$$

Le nombre de sommets des graphes G, G_1, $G_2, \ldots, G_m$ est toujours le même: n. Le nombre d'arêtes diminue d'un à chaque sous-graphe et ceux-ci sont tous connexes. Le processus doit s'arrêter: si ce n'était pas le cas, on aurait éventuellement un sous-graphe connexe sans arêtes! Ainsi, disons après m étapes, on obtient éventuellement un sous-graphe connexe qui ne possède pas de cycle simple: $G_m = (V_m, E_m)$. Par définition, G_m est un arbre.

Puisque G_m est un arbre, son nombre d'arêtes doit être un de moins que son nombre de sommets (par le théorème 4.2):

$$|E_m| = |V_m| - 1 = |V| - 1.$$

Or le sous-graphe G_m a été obtenu en retirant m arêtes de G, avec $m \geq 1$. Et par hypothèse sur G, on a $|E| = |V| - 1$, ce qui entraîne que

$$|E_m| = |E| - m = |V| - 1 - m.$$

Ainsi

$$|V| - 1 - m = |V| - 1$$

et donc

$$m = 0.$$

Mais ceci est impossible, car $m \geq 1$!

La contradiction ($m = 0$ et $m \geq 1$) nous force à rejeter l'hypothèse et à conclure qu'un graphe G ayant une arête de moins que le nombre de sommets est forcément un arbre. ◁

Théorème 4.5

Soit G un graphe ayant n sommets, avec $n \geq 2$. Si le nombre d'arêtes du graphe G est inférieur à $n-1$, alors G n'est pas connexe.

▷ ***Démonstration*** Soit G un graphe ayant n sommets, avec $n \geq 2$. Procédons par l'absurde pour démontrer que G n'est pas connexe.

Hypothèse: supposons que G est connexe. Alors soit G est un arbre ou ne l'est pas. Montrons que les deux cas entraînent une contradiction.

Si G est un arbre, cela contredit le théorème 4.2 stipulant que son nombre d'arêtes doit être $n-1$.

Si G n'est pas un arbre, alors G possède un circuit et, en empruntant la technique utilisée dans la démonstration du théorème 4.4, on peut engendrer une suite de sous-graphes connexes qui mène éventuellement à un arbre ayant moins de $n-1$ arêtes. Cela contredit le théorème 4.2 . ◁

Théorème 4.6

Pour chaque graphe connexe, il existe au moins un arbre couvrant.

De plus, si G_1 et G_2 sont deux arbres couvrants du graphe G, alors G_1 et G_2 possèdent le même nombre d'arêtes.

▷ ***Démonstration*** Soit G un graphe connexe ayant n sommets, avec $n \geq 2$. (Les cas $n = 0$ et $n = 1$ sont simples et remis en exercice). Le nombre d'arêtes du graphe connexe G doit être supérieur ou égal à $n-1$ (contraposée du théorème 4.5) :

$$|E| \geq n-1.$$

Si $|E| = n-1$, alors G est un arbre (par le théorème 4.2) et il est donc son propre arbre couvrant.

Si $|E| > n-1$, alors le graphe connexe G n'est pas un arbre : il contient donc au moins un circuit simple. On peut retirer n'importe quelle arête de ce circuit pour obtenir un sous-graphe $G_1 = (V_1, E_1)$ connexe (par théorème 4.3) tel que

$$V_1 = V \quad \text{et} \quad |E_1| = |E| - 1.$$

On continue ainsi à retirer des arêtes situées dans un circuit simple, comme dans la démonstration du théorème 4.4, pour obtenir une suite de sous-graphes connexes comportant une arête en moins à chaque étape, mais toujours les n sommets de G (voir figure 4.13). Lorsque le nombre d'arêtes du sous-graphe arrive à $n-1$, on a forcément un arbre. C'est un arbre couvrant de G. Ceci termine la démonstration du premier énoncé.

Par ailleurs, si G_1 et G_2 sont deux arbres couvrants du graphe $G = (V, E)$, alors G_1 et G_2 possèdent $|V| - 1$ arêtes par le théorème 4.2 : ils ont donc le même nombre d'arêtes. ◁

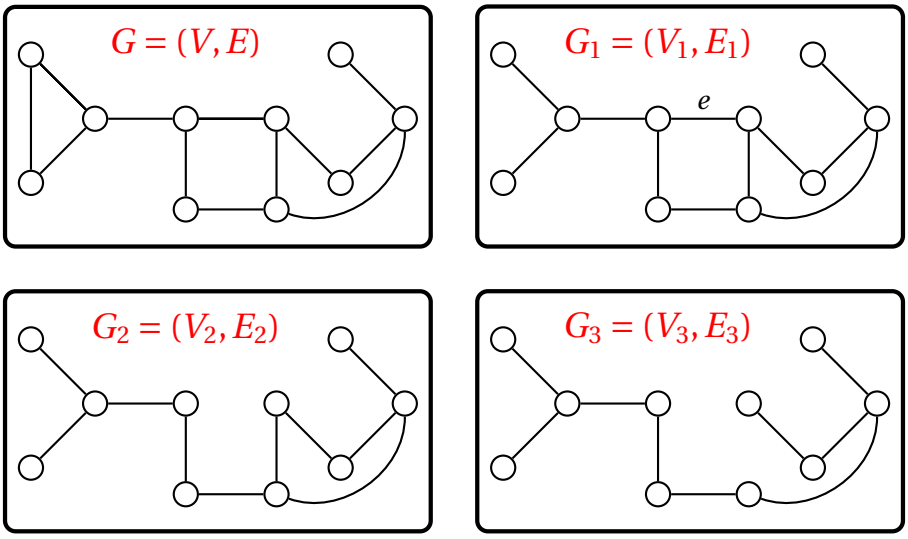
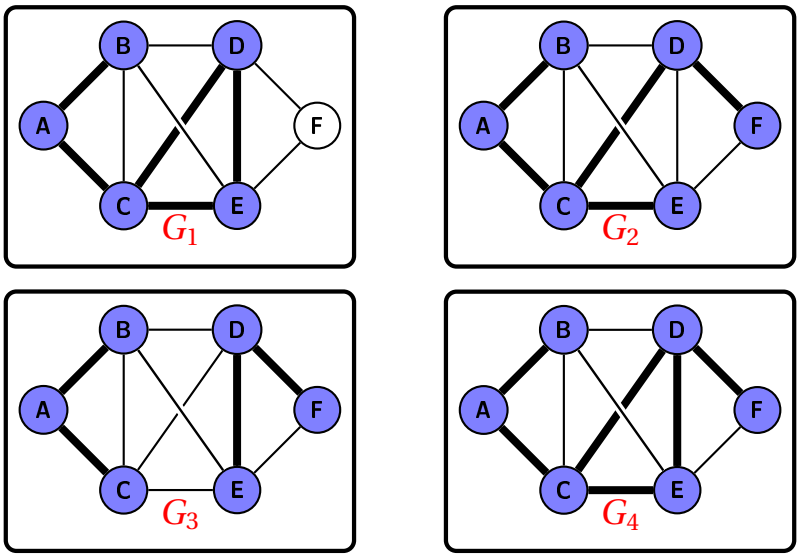


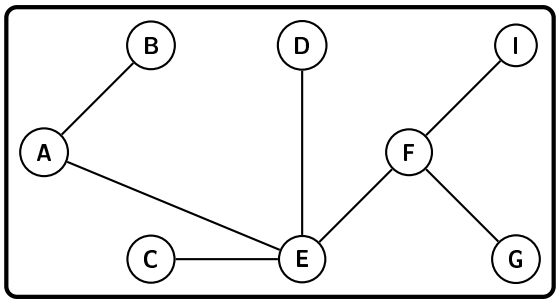
Figure 4.13 Illustration de la preuve du théorème 4.6 pour $n = 10$. Le graphe connexe G a 10 sommets et 12 arêtes. On lui retire successivement une arête située dans un cycle. Lorsque le nombre d'arêtes atteint 9 $(n-1)$, le sous-graphe obtenu est un arbre couvrant de G.

Exercices

4.1 Pour chacun des graphes G_1 à G_4, dites si le sous-graphe illustré en gras est un arbre couvrant du graphe de départ. Si ce n'est pas le cas, donnez au moins un critère de la définition 4.5 qui n'est pas satisfait.



4.2 Enracinez l'arbre suivant en **A** puis en **D** afin de produire deux arbres enracinés. Coloriez la racine pour l'identifier. De plus, déterminez qui est le parent et les enfants de **E** dans chacun des graphes enracinés.



4.3 Existe-t-il un graphe G possédant les caractéristiques suivantes? Si oui, dessinez-en un, sinon, dites pourquoi.

(a) G est un arbre ayant 5 sommets et 5 arêtes.

(b) G est un graphe acyclique ayant 5 sommets et 3 arêtes.

(c) G est un arbre ayant 5 sommets et 4 arêtes.

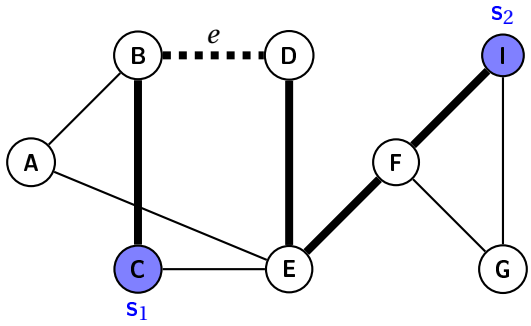
(d) G est un arbre ayant 5 sommets et 7 arêtes.

(e) G est un graphe ayant 3 sommets, 2 arêtes et n'est pas un arbre.

(f) G est un graphe simple ayant 3 sommets, 2 arêtes et G n'est pas un arbre.

(g) G est un arbre n'ayant aucune arête.

4.4 Les sommets $s_1 = C$ et $s_2 = I$ sont reliés par le chemin C–B–D–E–F–I. En utilisant la méthode décrite dans la démonstration du théorème 4.3, donnez le chemin alternatif reliant C et I si l'arête e est retirée du graphe. Il ne s'agit pas de trouver n'importe quel chemin par observation du graphe!



4.5 Utilisez différentes techniques de preuves afin de montrer l'équivalence entre trois définitions courantes des arbres :

(A) un graphe connexe et acyclique (notre définition d'arbre) ;

(B) un graphe vide (aucun sommet) ou un graphe connexe dont le nombre d'arêtes est $n-1$, où n désigne le nombre de sommets. ;

(C) un graphe où il existe un unique chemin simple entre chaque paire de sommets.

Vous devez donc montrer que si un graphe G satisfait l'une des 3 définitions, alors il satisfait aussi les 2 autres.

4.3.1 Arbre couvrant de poids minimal (algorithme de Prim)

Définition 4.6 : Arbre couvrant de poids minimal

Soit G un graphe pondéré connexe. Un **arbre couvrant de poids minimal** de G, appelé aussi arbre couvrant minimal de G, est un sous-graphe de G qui est un arbre couvrant de G dont la somme des poids des arêtes est la plus faible parmi tous les arbres couvrant de G.

Pour un graphe pondéré connexe G donné, il peut y avoir *plusieurs* arbres couvrants dont le poids est le même et est minimal. Par exemple, il suffit de considéré un graphe où toutes les arêtes ont le même poids, disons 1. Si le graphe G a n sommets, tout arbre couvrant aura $n-1$ arêtes et son poids sera de $n-1$.

Il existe plus d'un algorithme pour trouver un arbre couvrant minimal dans un graphe connexe. Ils ne fournissent pas nécessairement le même arbre, mais des arbres de poids égaux. Nous avons choisi de présenter l'algorithme de Prim, car son fonctionnement ressemble beaucoup à celui de l'algorithme de Dijkstra.

L'idée de cet algorithme est de commencer avec un arbre T constitué d'un seul sommet et d'aucune arête, comme à la figure 4.14.

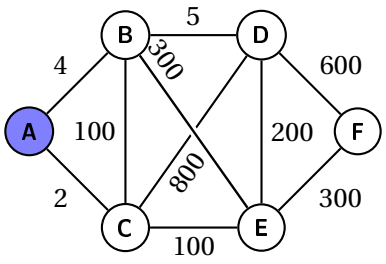


Figure 4.14 Au départ, T possède un seul sommet et aucune arête. C'est donc un arbre. On lui ajoutera ensuite le sommet le plus « proche », **C**, ainsi que l'arête qui relie **C** à T dont le poids est minimal : **A**–**C**.

On recherche alors un sommet **u** qui est « le plus proche » de l'arbre T, en considérant les poids comme des distances. On ajoute ce sommet **u** à T, ainsi que l'arête qui relie **u** à T dont le poids est minimal, tel qu'illustré à la figure 4.15. On démontre que T demeure un arbre à l'aide du théorème 4.4 : en effet, la construction garantit que T est connexe et que son nombre d'arêtes est un de moins que son nombre de sommets.

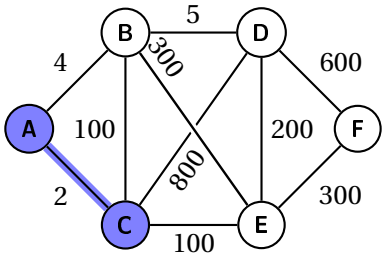


Figure 4.15 Après une étape, T demeure connexe, possède deux sommets et une arête : c'est donc un arbre. On lui ajoutera ensuite le sommet le plus « proche », **B**, ainsi que l'arête qui relie **B** à T dont le poids est minimal, **A**–**B**.

On continue ainsi jusqu'à ce que tous les sommets du graphes G soit ajoutés à ceux de l'arbre T, ce qui se produit après l'ajout de $n-1$ arêtes et $n-1$ sommets. T est alors un arbre couvrant de G. Il n'est pas évident pour le moment que T est un arbre couvrant *minimal*. Nous y reviendrons.

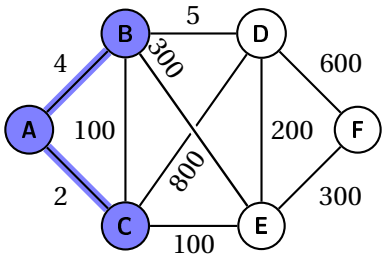


Figure 4.16 Après 2 étapes, T possède 3 sommets et 2 arêtes et demeure un arbre. On lui ajoutera ensuite le sommet le plus « proche », **D**, ainsi que l'arête qui relie **D** à T dont le poids est minimal, **B**–**D**.

Algorithme 6 `Prim`

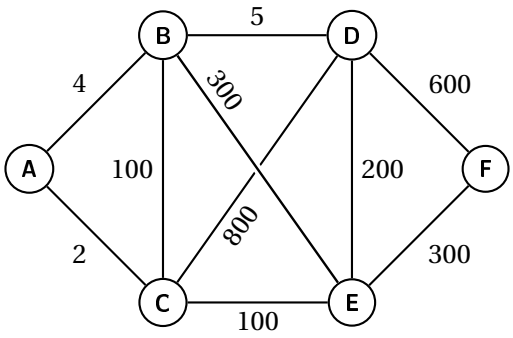
Entrées: $G = (V, E)$ un graphe simple pondéré connexe non orienté, $\omega : E \longrightarrow \mathbb{R}^+$ la fonction de pondération

Sortie: $T = (S, E_T)$, un arbre couvrant de poids minimal pour le graphe pondéré G.

1: $L :=$ tableau indexé par les sommets $\mathbf{v} \in V$ ▷ $L(\mathbf{v})$: poids minimal d'une arête $\mathbf{u}-\mathbf{v}$, avec $\mathbf{u} \in S$
2: $P :=$ tableau indexé par les sommets $\mathbf{v} \in V$ ▷ $P(\mathbf{v})$: sommet $\mathbf{u} \in S$ minimisant le poids de $\mathbf{u}-\mathbf{v}$
3: $\mathbf{a} :=$ un sommet de G (imposé ou choisi au hasard)
4: $E_T := \emptyset$ ▷ ensemble des arêtes de l'arbre T
5: $S := \emptyset$ ▷ ensemble des sommets de l'arbre T
6: **pour tout** sommet $\mathbf{v} \in V$ **faire**
7: $L(\mathbf{v}) := \infty$ ▷ car on ne peut pas encore atteindre $\mathbf{v}$ à partir d'un sommet de S
8: $P(\mathbf{v}) :=$ `null` ▷ $\mathbf{v}$ n'a pas encore de prédécesseur
9: **fin pour**
10: $L(\mathbf{a}) := 0$ ▷ pour que $\mathbf{a}$ soit le premier sommet ajouté à S
11: **tant que** $S \neq V$ **faire** ▷ tant que l'arbre T ne couvre pas tous les sommets du graphe G
12: $\mathbf{u} :=$ sommet $\notin S$ tel que $L(\mathbf{u})$ est minimal
13: $S := S \cup \{\mathbf{u}\}$ ▷ ajouter $\mathbf{u}$ aux sommets de T
14: $E_t := E_t \cup \{P(\mathbf{u})-\mathbf{u}\}$ ▷ ajouter l'arête allant de S à $\mathbf{u}$ aux arêtes de T (si $\mathbf{u} = \mathbf{a}$, ne rien ajouter car $P(\mathbf{a}) =$ `null`)
15: **pour tout** $\mathbf{v}$ tel que $\mathbf{u}-\mathbf{v} \in E$ et $\mathbf{v} \notin S$ **faire**
16: **si** $\omega(\mathbf{u}, \mathbf{v}) < L(\mathbf{v})$ **alors** ▷ si l'arête $\mathbf{u}-\mathbf{v}$ est moins lourde que les autres arêtes reliant $\mathbf{v}$ et S
17: $L(\mathbf{v}) := \omega(\mathbf{u}, \mathbf{v})$ ▷ on met à jour le poids minimal pour atteindre $\mathbf{v}$
18: $P(\mathbf{v}) := \mathbf{u}$ ▷ le prédécesseur du sommet $\mathbf{v}$ dans l'arbre T est $\mathbf{u}$
19: **fin si**
20: **fin pour**
21: **fin tant que**
22: **retourner** $T = (S, E_T)$

Exemple 4.6

Considérez le graphe pondéré ci-dessous.



(a) Utilisez l'algorithme de Prim avec **A** comme sommet de départ pour obtenir un arbre couvrant de poid minimal. Laissez une trace de l'algorithme.

(b) Donnez le poids de cet arbre couvrant.

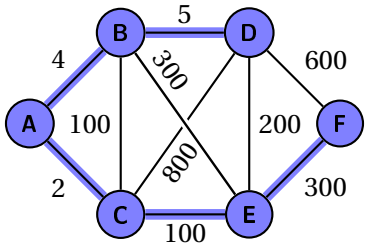
(c) Donnez l'état des tableaux L et P à la fin de l'algorithme.

Solution :

(a) Trace de l'algorithme.

u	a = A	B	C	D	E	F	sommet ajouté à S	arête ajoutée à E_T
	0	∞	∞	∞	∞	∞		
A		4_A	2_A	∞	∞	∞	A	
C		4_A		800_C	100_C	∞	C	A – C
B				5_B	100_C	∞	B	A – B
D					100_C	600_D	D	B – D
E						300_E	E	C – E
F							F	E – F

L'arbre couvrant minimal est représenté en gras sur le graphe pondéré.

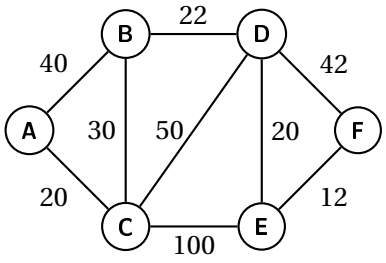


(b) Le poids total de cet arbre couvrant est 411.

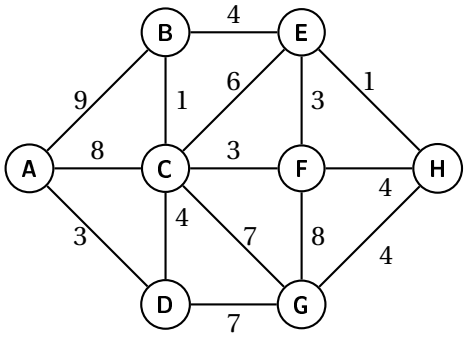
(c) $L = [0,4,2,5,100,300]$ et $P = [\texttt{null}, \mathsf{A}, \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{E}]$.

Exercices

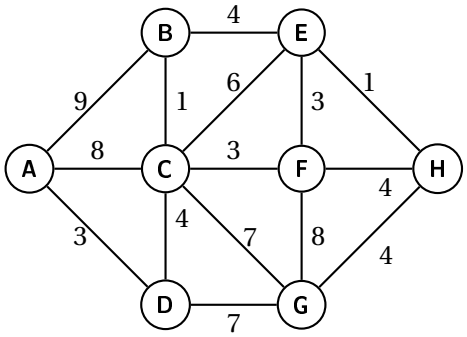
4.6 Utilisez l'algorithme de Prim pour trouver un arbre couvrant minimal du graphe ci-dessous, en prenant le sommet **D** comme point de départ. Donnez le poids de cet arbre. Laissez une **trace** de l'algorithme. De plus, dites quelle a été la deuxième arête ajoutée à l'arbre.



4.7 Utilisez l'algorithme de Prim pour trouver un arbre couvrant minimal du graphe ci-dessous, en prenant le sommet **E** comme point de départ. Donnez le poids de cet arbre. De plus, dites quelle a été la dernière arête ajoutée à l'arbre.

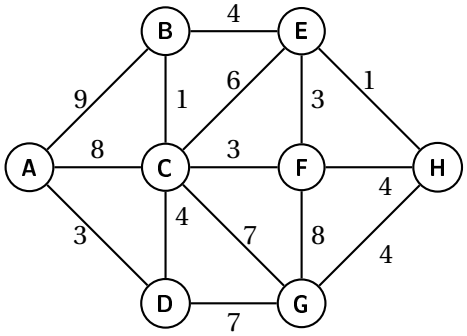


4.8 Considérez le graphe pondéré ci-dessous. Utilisez l'algorithme de Dijkstra modifié [1], en laissant une trace, pour obtenir l'**arbre des chemins de poids minimaux issus du sommet A**. Il est constitué de l'ensemble des chemins allant de **A** à **v**, pour chacun des autres sommets **v** du graphe. L'arbre obtenu est-il un arbre couvrant de poids minimal? Justifiez.



4.9 Considérez le graphe pondéré ci-dessous.

- Utilisez l'algorithme de Dijkstra modifié (voir exercice 4.8) pour obtenir l'arbre des chemins de poids minimaux issus du sommet **E**.
- Tracez l'arbre obtenu sous la forme d'arbre enraciné en **E**, comme un arbre généalogique.
- Donnez l'état des tableaux L et P à la fin de l'algorithme.
- L'arbre obtenu est-il un arbre couvrant de poids minimal? Justifiez.



1. Remplacez la ligne 9 par **tant que** $S \neq V$ **faire** pour éviter que l'algorithme ne s'arrête avant d'avoir visité tous les sommets.

4.10 ★ À partir des tableaux L et P à la fin de l'algorithme de Disktra modifié, donnez l'arbre des chemins de poids minimaux issus de **E** ainsi que le poids total de cet arbre. *Vous n'avez pas besoin de connaître le graphe G en question, seulement de savoir que ses 10 sommets sont étiquetés* **A** *à* **J**.

$L = [13,4,5,9,0,3,5,1,9,6]$ et $P = [\mathsf{B},\mathsf{E},\mathsf{B},\mathsf{C},\texttt{null},\mathsf{E},\mathsf{H},\mathsf{E},\mathsf{B},\mathsf{H}]$

Chapitre 5

Introduction à la complexité des algorithmes

En informatique, on considère habituellement qu'un algorithme est une suite d'opérations permettant de résoudre un problème. Cette suite d'opérations peut être *implémentée* de sorte à obtenir un programme qui résout ce problème de manière effective.

En général, il existe plusieurs algorithmes pour résoudre un problème donné. Par exemple, étant donné deux entiers positifs x et y, pour calculer x modulo y, on peut :

- **Algorithme A.** Soustraire y à x jusqu'à obtenir une valeur dans l'intervalle $[0, y[$. Cette valeur est la solution.
- **Algorithme B.** Effectuer la division de x par y avec la méthode du *crochet*[1]. Le reste de la division est la solution.

Exemple 5.1 (à compléter en classe)
Calculez 63 modulo 5 en utilisant d'abord l'algorithme A (soustractions successives), puis l'algorithme B (division par crochet).

Solution :

Même si dans certains cas l'algorithme A est plus rapide que le B, dans la grande majorité des cas c'est l'algorithme B qui est le plus rapide. En effet, il est évident que pour calculer 100 000 mod 7, l'algorithme A est beaucoup plus lent.

1. Il s'agit de la méthode habituellement enseignée au primaire.

Dans le cas de cet exemple, il est facile de déterminer quel algorithme est le plus efficace, mais de manière générale *comment comparer l'efficacité de deux algorithmes ?*

Cette simple question a donné naissance à un domaine d'études appelé la **théorie de la complexité**, dont voici quelques principes clés :

- le **temps** est mesuré en **nombre d'opérations**,
- on s'intéresse à la **croissance** du nombre d'opérations en fonction de la taille de l'entrée,
- pour une taille de données fixée, on s'intéresse toujours au **pire cas** possible.

Ce chapitre d'introduction à la théorie de la complexité se divise en deux parties. Nous verrons d'abord comment représenter la complexité d'un algorithme à l'aide d'une fonction $f(n)$, où n correspond à la taille des données à traiter et $f(n)$ correspond aux nombres d'opérations qui seront effectuées par l'algorithme. Comparer la rapidité de deux algorithmes reviendra donc à comparer deux fonctions.

À la deuxième section, nous introduirons les notations grand-O et grand-Θ. Celles-ci sont en quelque sorte analogues aux symboles $\leq$ et $=$ car elles permettent de comparer « *l'ordre de grandeur* » des fonctions. Cela nous permettra de comparer les algorithmes afin d'identifier lesquels sont plus efficaces pour traiter des données de grande taille.

5.1 Mesurer un temps de calcul à l'aide d'une fonction

Vous souvenez-vous comment additionner et multiplier des nombres à la main ? L'algorithme enseigné au primaire pour additionner deux nombres va comme suit (expliqué ici à un humain et non à une machine) :

1. superposer les 2 nombres de façon à ce que les chiffres des unités soient alignés
2. tracer une ligne horizontale sous le nombre du bas
3. débuter par la colonne des unités (colonne de droite) :
   (a) additionner les 2 chiffres de la colonne
   (b) inscrire le chiffre des unités du résultat en dessous en notant la retenue au-dessus de la colonne située juste à gauche s'il y a lieu
4. passer à la colonne suivante
   (a) additionner les chiffres de la colonne (possibilité de 3 chiffres, car retenue possible)
   (b) inscrire le chiffre des unités du résultat en dessous en notant la retenue au-dessus de la colonne située juste à gauche s'il y a lieu
5. répéter le processus jusqu'à la dernière colonne (colonne la plus à gauche)
6. abaisser la dernière retenue potentielle, c'est-à-dire réécrire ce chiffre sous la ligne horizontale
7. lire le résultat sous la ligne horizontale

Exemple 5.2 (à compléter en classe)
Calculez 3186 + 5916 en utilisant l'algorithme décrit ci-dessus. De plus, déterminez le nombre d'additions simples de 2 chiffres que l'on doit-on effectuer pour obtenir le résultat.

Solution :

Exemple 5.3 (à compléter en classe)
Si les 2 nombres à additionner ont chacun 10 chiffres, combien d'additions simples de 2 chiffres doit-on effectuer pour obtenir le résultat ?

Et si les nombres ont 100 chiffres chacun ?

De façon plus générale, si les 2 nombres à additionner ont chacun n chiffres, combien d'additions simples de 2 chiffres doit-on effectuer pour obtenir le résultat ?

Solution :

Exemple 5.4 (à compléter en classe)
Considérons maintenant l'algorithme de multiplication de 2 nombres de n chiffres enseigné au primaire, similaire à celui exposé ci-dessus pour l'addition. Combien cet algorithme nécessite-t-il de multiplications simples de 2 chiffres ?

Solution :

Le **temps d'exécution** d'un algorithme donné dépend principalement de

- la machine utilisée : langage, système d'exploitation, compilateur, vitesse de calcul de la machine, etc.
- les données auxquelles l'algorithme est appliqué.

Nous allons utiliser une mesure plus abstraite, qui ne dépend pas de la machine ni des données elles-mêmes, mais plutôt de la *taille* des données. Par exemple, le temps d'exécution d'un algorithme de tri dépend de la longueur de la liste à trier. Le temps d'exécution d'une multiplication de deux nombres dépend du nombre de chiffres de ces nombres. Nous utiliserons donc une fonction pour décrire la complexité d'un algorithme (voir figure 5.1).

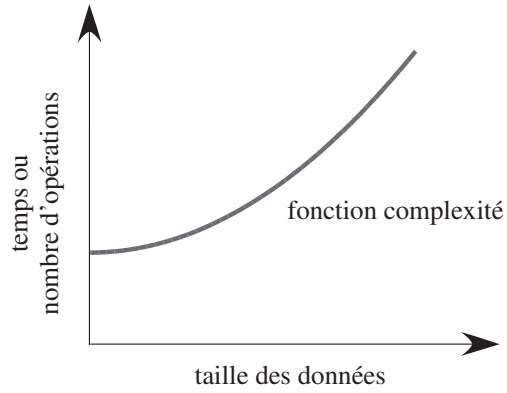


Figure 5.1 On utilise une fonction pour décrire la complexité d'un algorithme.

Exemple 5.5 (à compléter en classe)

Considérez l'algorithme de fouille séquentielle dans une liste quelconque (fonction **recherchelin** à la figure 5.2) et l'algorithme de fouille dichotomique **dans une liste ordonnée en ordre croissant** (fonction **recherchebin** à la figure 5.2). Nous choisissons ici d'étudier la complexité dans *le pire des cas*.

(a) Pour chacun des algorithmes, trouvez le nombre de comparaisons requises pour traiter une liste de taille n. Considérez autant les comparaisons de nombres entiers que les comparaisons d'éléments de la liste avec l'élément recherché. Lequel des deux algorithmes requiert au plus $f(n) = 2n+2$ comparaisons pour traiter une liste de taille n? Et lequel en nécessite au plus $g(n) = 2\log_2(n)+2$?

(b) Lequel des deux algorithmes sera plus efficace pour traiter de grandes listes de données préalablement ordonnées en ordre croissant?

```
Define recherchelin(t,x)=
Func
© t est un tableau, i.e. une liste, indexé par 1, 2, ..., n.
© x est l'élément recherché dans le tableau t.
© Retourne l'indice, i.e. la position, de l'élément x s'il est dans le tableau t
© et retourne 0 si x n'est pas dans le tableau.
Local n,i,position
n:=dim(t)
i:=1
While i≤n
    If x=t[i]:Exit
    i:=i+1
EndWhile
If i≤n Then
  position:=i
Else
  position:=0
EndIf
Return position
EndFunc
```

```
Define recherchebin(t,x)=
Func
© t est un tableau, i.e. une liste, ORDONNÉ indexé par 1, 2, ..., n.
© x est l'élément recherché dans le tableau t.
© Retourne l'indice, i.e. la position, de l'élément x s'il est dans le tableau t
© et retourne 0 si x n'est pas dans le tableau.
Local n,a,b,position,m
n:=dim(t)
a:=1
b:=n+1
While b−a>1
    m:=floor((a+b)/2)
    If t[m]≤x Then
      a:=m
    Else
      b:=m
    EndIf
EndWhile
If x=t[a] Then
  position:=a
Else
  position:=0
EndIf
Return position
EndFunc
```

Figure 5.2 Fouille séquentielle et fouille dichotomique en TI Nspire.

Exemple 5.6 (à compléter en classe)
Considérez l'algorithme A qui nécessite $f(n)$ opérations d'un certain type pour résoudre un problème de taille n et l'algorithme B qui en nécessite $g(n)$. Utilisez vos connaissances mathématiques préalables afin de déterminez lequel, selon vous, sera plus efficace pour résoudre les gros problèmes si

(a) $f(n) = n^3 + 3$ et $g(n) = 25 + n^2$;

(b) $f(n) = 2^n + 4n^3$ et $g(n) = 10n^4$.

5.2 Notation grand-O et grand-Θ

Il y a plusieurs facteurs qui influencent la vitesse à laquelle un programme exécute une tâche précise :

- l'ordinateur sur lequel il est lancé ;
- le système d'exploitation ;
- la taille des données à traiter pour cette tâche ;
- l'utilisation de la mémoire ;
- etc.

Nous nous concentrerons ici uniquement sur la fonction de complexité d'un algorithme, c'est-à-dire le **nombre d'opérations** requises par l'algorithme pour traiter un problème. Si la taille des données à traiter double, par exemple pour trier une liste de 1000 ou de 2000 entrées, le nombre d'opérations requises demeurera-t-il constant ? Doublera-t-il ? Sera-t-il mis au carré ? Au cube ?

Concrètement, il peut s'avérer fastidieux de compter précisément toutes les opérations effectuées lors de l'exécution d'un algorithme. Afin de nous simplifier la tâche dans le cadre de ce chapitre, **l'opération la plus significative** sera sélectionnée et nous compterons uniquement le nombre de fois que cette opération est effectuée.

Le rappel de la méthode d'additions présentée à la section 5.1 nous a conduits au fait que le nombre d'additions de chiffres requises pour additionner deux nombres de n chiffres est au plus $f(n) = 2n - 1$. L'addition de deux nombres de 100 chiffres requiert donc au plus 199 additions, tandis que l'addition de deux nombres de 200 chiffres requiert au plus 399 additions. Quand la taille des données augmente, le nombre d'opérations requises pour les traiter (les additionner) ne demeure pas constant. Il augmente environ du même facteur.

Pour des algorithmes plus compliqués, il est souvent difficile et même impossible de déterminer précisément la fonction de complexité. En fait, pour comparer deux algorithmes, il n'est pas toujours nécessaire de connaître exactement le nombre d'opérations effectuées par chacun d'eux : bien souvent, il suffit de connaître leur ordre de grandeur. Il nous faut donc un outil qui permet de déterminer et de comparer ces ordres de grandeur sur les fonctions de complexités. Cet outil existe ; il s'agit de la notation grand-O. La notation grand-O regroupe en sous-ensemble les fonctions dont la croissance est comparable. On peut ensuite les trier, de la croissance le plus lente à la croissance la plus rapide. Chacun des sous-ensembles est identifié par son représentant le plus simple possible.

- Les fonctions constantes, ou qui sont bornées par une fonction constante, sont regroupées en un sous-ensemble qui est identifié par $O(1)$, car la fonction constante la plus « simple » possible est $f(n) = 1$. Les fonctions de complexité de l'ensemble $O(1)$ correspondent à des algorithmes qui ne prennent pas plus de temps pour traiter un problème dont la taille est grande ou petite. La figure 5.3 illustre 3 fonctions de l'ensemble $O(1)$.
- Les fonctions qui sont bornées par un polynôme de degré 1 forment le sous-ensemble $O(n)$. La figure 5.4 illustre 3 fonctions de l'ensemble $O(n)$.
- Les fonctions qui sont bornées par un polynôme de degré 2 forment le sous-ensemble $O(n^2)$.

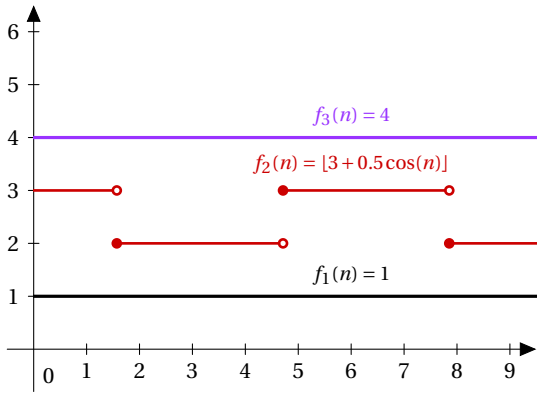


Figure 5.3 Les fonctions qui sont bornées par une fonction constante forment le sous-ensemble qui est identifié par $O(1)$. Par exemple, on a $f_1 \in O(1)$, $f_2 \in O(1)$, $f_3 \in O(1)$.

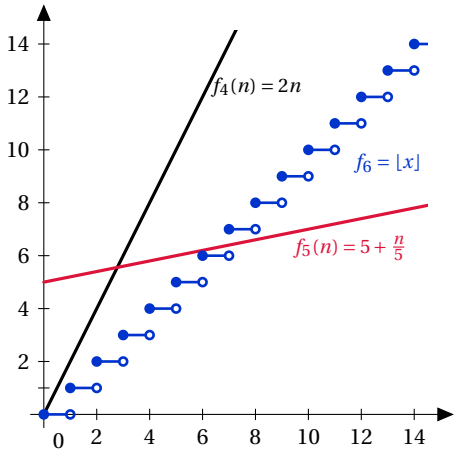


Figure 5.4 Les fonctions qui sont bornées par un polynôme de degré 1 forment le sous-ensemble qui est identifié par $O(n)$. Par exemple, on a $f_4 \in O(n)$, $f_5 \in O(n)$, $f_6 \in O(n)$.

Définition 5.1 : Grand-O, facteur, témoin

Soit f et g deux fonctions de $\mathbb{R}$ vers $\mathbb{R}$ ou de $\mathbb{N}$ vers $\mathbb{R}$. On dit que $f(x)$ est **grand-O** de $g(x)$, que l'on note $f(x) \in O(g(x))$, si, en valeur absolue, f est éventuellement dépassée par un multiple de g. Ainsi,

$$f(x) \in O(g(x)) \iff \exists C \in \mathbb{R}, \exists k \in \mathbb{R}, \forall x \in \mathrm{dom}(f),\ x > k \rightarrow |f(x)| \leq C|g(x)|$$

Cela signifie donc que la fonction $|f|$ est bornée par un multiple de la fonction $|g|$ à partir d'une certaine valeur k du domaine (voir figure 5.5).

La borne k, appelée ***seuil***, permet d'ignorer le comportement des fonctions pour les données de petite taille (dans ces cas, la complexité de l'algorithme est souvent dominée par des opérations d'initialisation qui deviennent négligeables pour des données plus grandes).

La constante C, appelée ***facteur***, permet de faire abstraction de la vitesse de la machine utilisée.

Les nombres k et C sont appelés ***témoins*** de la relation $f(x) \in O(g(x))$.

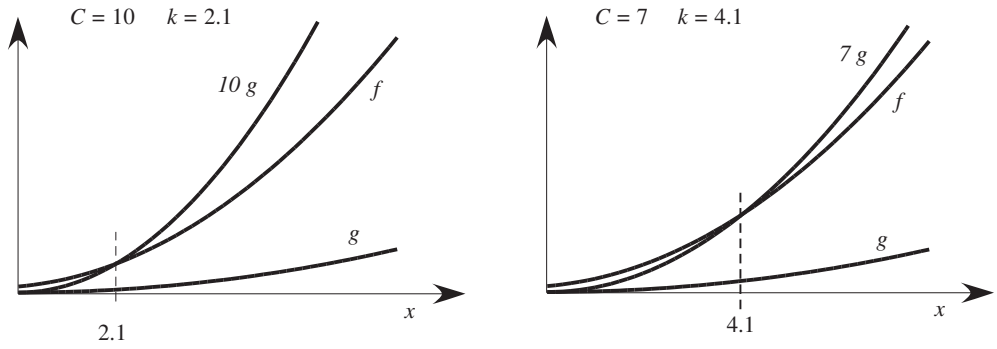


Figure 5.5 Pour montrer que $f(x) \in O(g(x))$, il suffit de trouver une paire de témoins C et k, mais il en existe une infinité. Nous en présentons deux paires ici. Le but est de montrer qu'il existe un multiple de g qui dépasse f pour toutes les valeurs de x à partir d'un certain seuil.

Notons que la définition repose sur une inégalité. Ainsi, les sous-ensembles de fonctions définis par cette notation sont imbriqués. Par exemple, on a

$$O(1) \subseteq O(n) \subseteq O(n^2).$$

Définition 5.2 : Grand-Θ

Soit f et g deux fonctions de $\mathbb{R}$ vers $\mathbb{R}$ ou de $\mathbb{N}$ vers $\mathbb{R}$. La notation « **grand thêta** » est utilisée pour désigner que chacune des 2 fonctions est grand-O de l'autre. On dit alors que les fonctions sont **du même ordre**.

$$f(x) \in O(g(x)) \ \text{ et } \ g(x) \in O(f(x)) \longleftrightarrow f(x) \in \boldsymbol{\Theta(g(x))}$$

Exemple 5.7

Considérez les deux fonctions de $\mathbb{R}$ vers $\mathbb{R}$ suivantes :

$$f(x) = 5x^2 + 6x + 9 \qquad \text{et} \qquad g(x) = x^2.$$

Montrez que $f(x) \in \Theta(g(x))$ en fournissant les témoins obtenus algébriquement et non par l'observation d'un graphique.

Solution :

Nous devons donc démontrer que $f(x) \in O(g(x))$ et que $g(x) \in O(f(x))$. Notons d'abord que ces deux fonctions prennent des valeurs positives lorsque $x \geq 0$. Nous pouvons donc laisser tomber les valeurs absolues dans les inégalités à vérifier.

Pour montrer que $f(x) \in O(g(x))$, nous devons montrer que $f(x)$ est inférieure ou égale à un multiple de $g(x)$ à partir d'une certaine valeur du domaine. On sait que

$$\begin{aligned} x \geq 3 \quad &\rightarrow \quad 9 \leq x^2; \\ x \geq 6 \quad &\rightarrow \quad 6x \leq x^2. \end{aligned}$$

En additionnant ces inégalités et en en additionnant $5x^2$ de chaque côté, on obtient

$$x \geq 6 \quad \rightarrow \quad 5x^2 + 6x + 9 \leq 5x^2 + x^2 + x^2 = 7x^2.$$

Nous avons donc trouvé une paire de témoins de la relation $f(x) \in O(g(x))$: $C = 7$ et $k = 6$. En effet,

$$\forall x \in \mathbb{R},\ x \geq 6 \rightarrow f(x) \leq 7g(x).$$

Le graphique 5.6 illustre la situation.

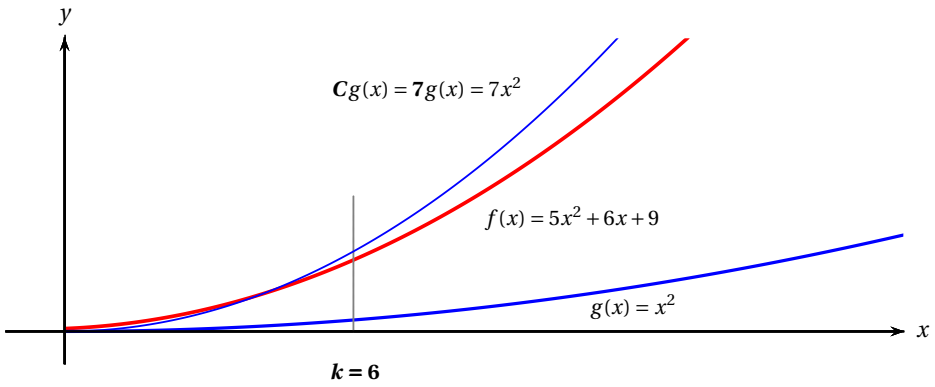


Figure 5.6 Illustration de la relation $f(x) \in O(g(x))$, avec les témoins $k = 6$ et $C = 7$.

Pour montrer que $g(x) \in O(f(x))$, nous devons montrer que $g(x)$ est inférieure ou égale à un multiple de $f(x)$ à partir d'une certaine valeur du domaine. Or, nous n'avons même pas à multiplier f par une constante pour dépasser g. On sait que

$$x \geq 0 \quad \rightarrow \quad x^2 \leq 5x^2 + 6x + 9.$$

Nous avons donc trouvé une paire de témoins de la relation $g(x) \in O(f(x))$: $C = 1$ et $k = 0$. En effet,

$$\forall x \in \mathbb{R},\ x \geq 0 \rightarrow g(x) \leq 1f(x).$$

En général, il est laborieux de prouver la validité des témoins k et C d'une relation grand-O et, dans la pratique, les valeurs de ces témoins n'ont pas d'importance : l'important est de savoir que $f(x)$ est $O(g(x))$. Les théorèmes suivants permettent d'établir une relation grand-O tout en évitant la recherche de témoins.

Théorème 5.1

Soit f et g deux fonctions de $\mathbb{R}$ vers $\mathbb{R}$ ou de $\mathbb{N}$ vers $\mathbb{R}$.

- Si
$$\lim_{x\to\infty}\left|\frac{f(x)}{g(x)}\right| = b \quad \text{avec} \quad 0 < b < \infty$$
alors les fonctions f et g sont du **même ordre de grandeur** :
$f(x) \in O(g(x))$ et $g(x) \in O(f(x))$.
- Si
$$\lim_{x\to\infty}\left|\frac{f(x)}{g(x)}\right| = 0$$
alors les fonctions f et g **ne** sont **pas du même ordre de grandeur** :
$f(x) \in O(g(x))$ mais $g(x) \notin O(f(x))$. La fonction f est donc négligeable face à la fonction g quand x tend vers l'infini.
- Si
$$\lim_{x\to\infty}\left|\frac{f(x)}{g(x)}\right| = \infty$$
alors les fonctions f et g **ne** sont **pas du même ordre de grandeur** :
$g(x) \in O(f(x))$ mais $f(x) \notin O(g(x))$. La fonction g est donc négligeable face à la fonction f quand x tend vers l'infini.

Exemple 5.8

Soit $f(x) = 7x^2$, $g(x) = x^3$ et $h(x) = x^2$. Est-ce que $f(x) \in O(g(x))$? Est-ce que $g(x) \in O(f(x))$?

Est-ce que $f(x) \in O(h(x))$? Est-ce que $h(x) \in O(f(x))$?

Solution :

Calculons la limite
$$\begin{aligned}\lim_{x\to\infty}\left|\frac{f(x)}{g(x)}\right| &= \lim_{x\to\infty}\frac{7x^2}{x^3}\\ &= \lim_{x\to\infty}\frac{7}{x}\\ &= 0.\end{aligned}$$

Par le théorème 5.1, les fonctions f et g ne sont donc pas du même ordre :

$$7x^2 \in O(x^3) \text{ mais } x^3 \notin O(7x^2).$$

Calculons la limite
$$\begin{aligned}\lim_{x\to\infty}\left|\frac{f(x)}{h(x)}\right| &= \lim_{x\to\infty}\frac{7x^2}{x^2}\\ &= \lim_{x\to\infty}7\\ &= 7\end{aligned}$$

Par le théorème 5.1, les fonctions f et h sont du même ordre :

$$7x^2 \in O(x^2) \text{ et } x^2 \in O(7x^2).$$

Notez que l'on souhaite habituellement trouver **le plus petit sous-ensemble** de fonctions auxquelles appartient la fonction de complexité que l'on étudie, disons $f(x) = 7x^2$. Pour cela, il faut trouver la fonction aussi simple que possible et ayant la complexité la moins grande qui est du même ordre de grandeur que la fonction f. Ainsi, même s'il est vrai que

$$7x^2 \in O(x^3),$$

cette proposition nous renseigne moins que celle-ci :

$$7x^2 \in O(x^2).$$

Exemple 5.9

Soit $f(x) = 5x^2 + 2x + 7$ et $g(x) = x^2$. Est-ce que $f(x) \in O(g(x))$? Est-ce que $g(x) \in O(f(x))$?

Solution :

$$\begin{aligned} \lim_{x\to\infty} \left|\frac{f(x)}{g(x)}\right| &= \lim_{x\to\infty} \frac{5x^2+2x+7}{x^2} \\ &= \lim_{x\to\infty} \left(5 + \frac{2}{x} + \frac{7}{x^2}\right) \\ &= 5 \end{aligned}$$

Ainsi, par le théorème 5.1, les fonctions f et g sont du même ordre de grandeur :

$$5x^2 + 2x + 7 \in O(x^2) \qquad \text{et} \qquad x^2 \in O(5x^2 + 2x + 7).$$

Exemple 5.10

Soit $f(n) = \log_2(n)$ et $g(n) = n$. Est-ce que $f(n) \in O(g(n))$? Est-ce que $g(n) \in O(f(n))$?

Solution :

$$\begin{aligned} \lim_{x\to\infty} \left|\frac{f(n)}{g(n)}\right| &= \lim_{n\to\infty} \frac{\log_2(n)}{n} && \text{qui est de la forme } \frac{\infty}{\infty} \\ &= \lim_{n\to\infty} \frac{\frac{1}{n\ln(2)}}{1} && \text{par la règle de l'Hopital} \\ &= 0 \end{aligned}$$

donc, par le théorème 5.1, les fonctions f et g ne sont pas du même ordre de grandeur :

$$\log_2(n) \in O(n) \qquad \text{mais} \qquad n \notin O(\log_2(n)).$$

La même démarche fonctionne pour un logarithme de n'importe quelle base.

Exemple 5.11

Soit $f(n) = \log_2(n)$ et $g(n) = \log(n)$. Est-ce que $f(n) \in O(g(n))$? Est-ce que $g(n) \in O(f(n))$?

Solution :

Grâce à la loi des logarithmes (changement de base), on a

$$\log_2(n) \quad = \quad \frac{\log_{10}(n)}{\log_{10}(2)}.$$

Donc

$$\lim_{x\to\infty} \left| \frac{f(n)}{g(n)} \right| \quad = \quad \frac{1}{\log_{10}(2)}.$$

Ainsi, par le théorème 5.1, les fonctions f et g sont du même ordre de grandeur :

$$\log_2(n) \in O(\log_{10}(n)) \qquad \text{et} \qquad \log_{10}(n) \in O(\log_2(n)).$$

Théorème 5.2

Dans la liste de fonctions ci-dessous, chaque fonction est grand-O des fonctions qui sont situées plus à droite, mais n'est pas grand-O des fonctions situées plus à gauche.

$$1,\ \log(n),\ \sqrt{n},\ n,\ n\log(n),\ n^2,\ n^3,\ n^4,\ \ldots,\ 2^n,\ 3^n,\ 4^n,\ \ldots,\ n!,\ n^n$$

Autrement dit :

$$O(1) \subset O(\log(n)) \subset O(\sqrt{n}) \subset O(n) \subset O(n\log(n)) \subset O(n^2) \subset O(n^3) \subset O(n^4)$$

$$\subset \cdots \subset O(2^n) \subset O(3^n) \subset O(4^n) \subset \cdots \subset O(n!) \subset O(n^n).$$

La figure 5.7 illustre l'imbrication de quelques-uns des sous-ensembles du théorème 5.2. La figure 5.8 illustre que les sous-ensembles $\Theta(1), \Theta(n^2), \Theta(2^n), \ldots, \Theta(n^n)$ sont disjoints, contrairement aux sous-ensembles $O(1), O(n^2), O(2^n), \ldots, O(n^n)$ qui sont emboîtés.

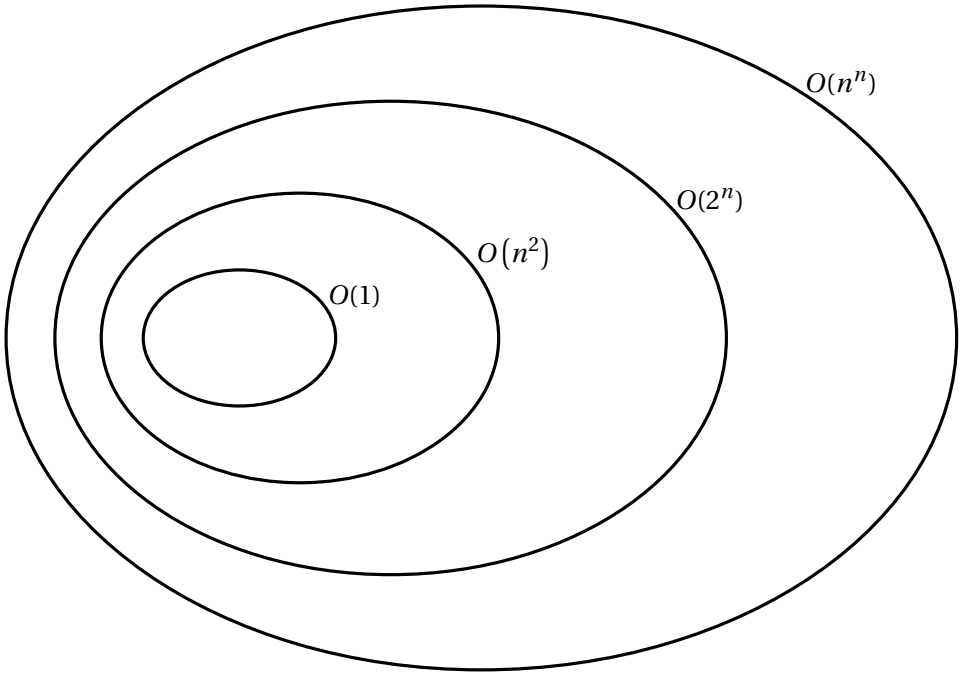


Figure 5.7 Diagramme de Venn de quatre sous-ensembles de fonctions pour la notation grand-O.

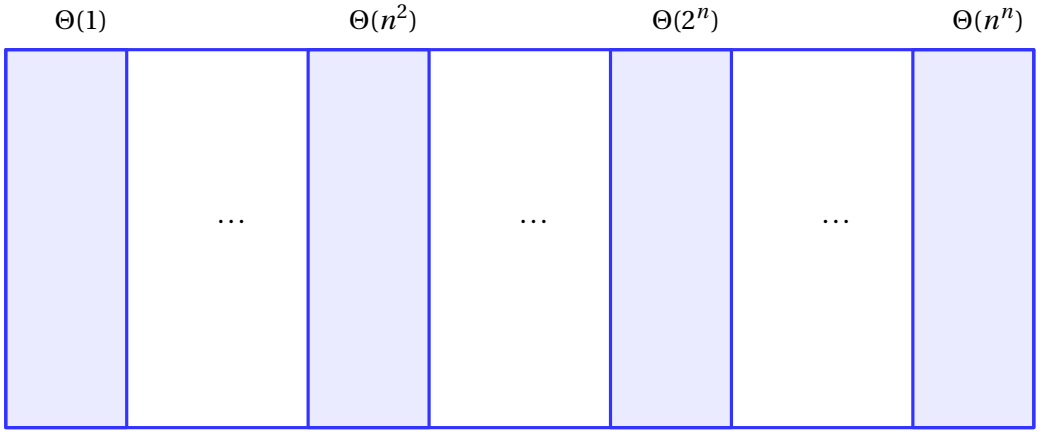


Figure 5.8 Diagramme de Venn illustrant que les sous-ensembles $\Theta(1), \Theta(n^2), \Theta(2^n)$ et $\Theta(n^n)$ sont disjoints, contrairement aux sous-ensembles $O(1), O(n^2), O(2^n)$ et $O(n^n)$ qui sont emboîtés. Le grand rectangle désigne l'ensemble $O(n^n)$.

Les graphes de la figure 5.9 illustrent la croissance des fonctions de la liste du théorème 5.2.

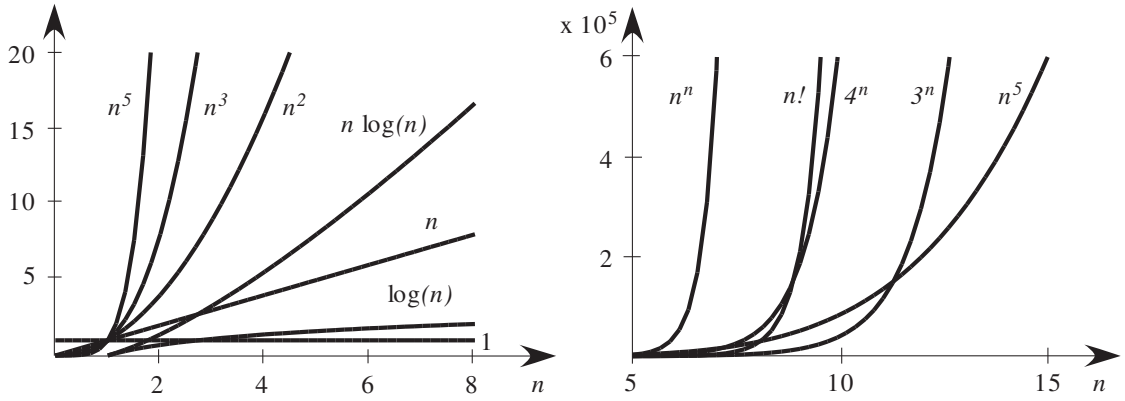


Figure 5.9 Illustration de la liste de fonctions du théorème 5.2.

Par exemple, la liste du théorème 5.2 nous informe que la fonction 2^n croît plus vite que toutes les puissances de n (comme n^3, n^4 ou n^{10}) puisqu'elle est située plus à droite. Elle finira donc par les dépasser éventuellement comme l'illustre la figure 5.10.

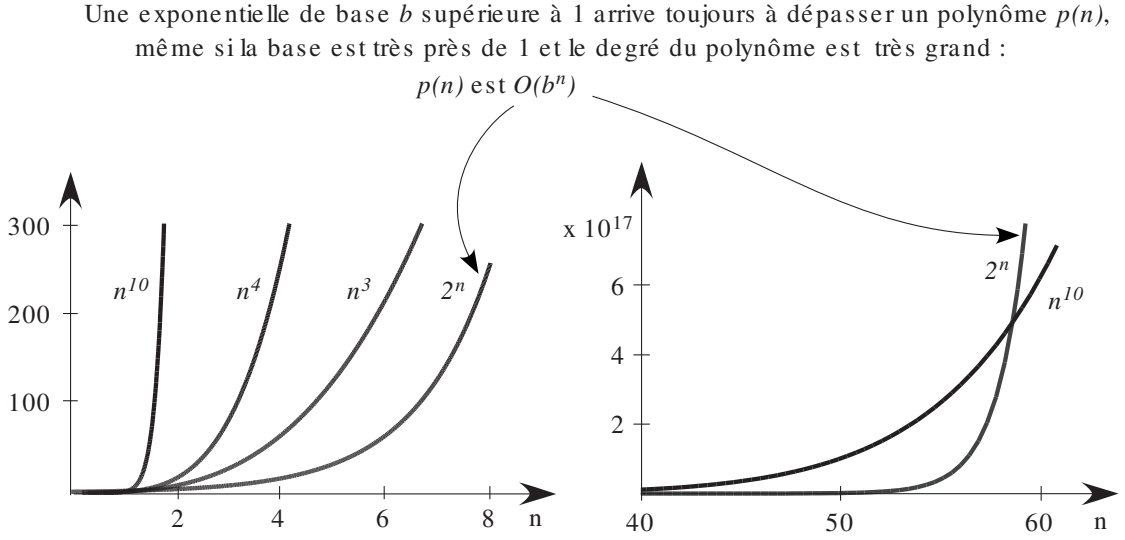


Figure 5.10 Illustration de la liste de fonctions du théorème 5.2.

5.2.1 Grand-O d'une fonction composée

Théorème 5.3 : Grand-O de la somme

Quand on additionne des fonctions, la somme appartient au plus grand des ensembles grand-O de chacune des fonctions.

Ainsi, si $f_1 \in O(g_1)$ et $f_2 \in O(g_2)$ et si $O(g_1) \subseteq O(g_2)$ alors

$$(f_1 + f_2) \in O(g_2).$$

Théorème 5.4 : Grand-O du produit

Quand on multiplie des fonctions, le produit appartient au grand-O du produit des représentants des ensembles grand-O de chacune des fonctions.

Ainsi, si $f_1 \in O(g_1)$ et $f_2 \in O(g_2)$ alors

$$f_1 f_2 \in O(g_1 g_2).$$

La figure 5.11 illustre les théorèmes du grand-O de la somme et du produit de fonctions.

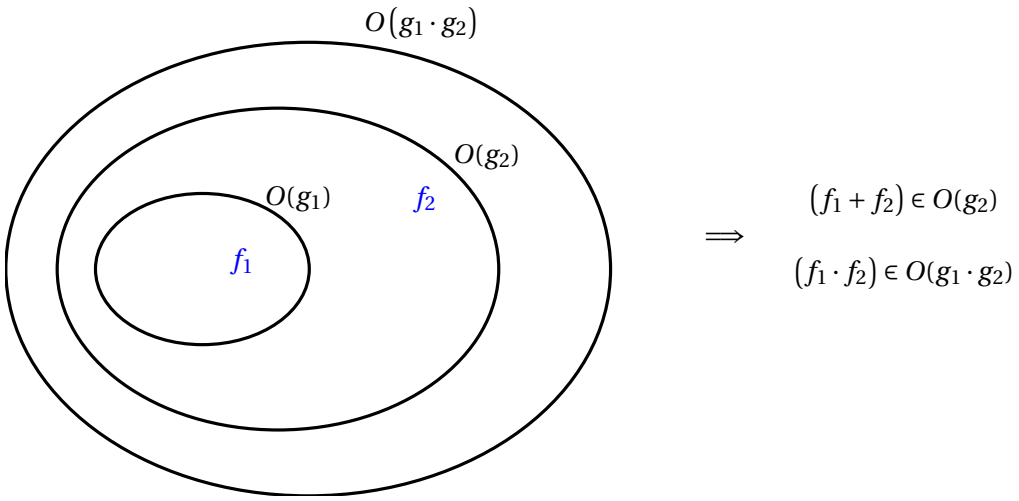


Figure 5.11 Illustration des théorèmes du grand-O de la somme et du produit.

Théorème 5.5 : Grand-Θ d'un polynôme

Si $p(n)$ est un polynôme de degré d alors $p(n) \in \Theta(n^d)$.

Exemple 5.12

Soit

$$f(n) = 3\log(n) + 7n^5 + 4n.$$

Trouvez la fonction $g(n)$ *la plus simple possible* telle que $f(n) \in O(g(n))$.

Solution :
En utilisant le théorème 5.1 ou le théorème 5.2, on a que

$$f_1(n) = 3\log(n) \in O\big(\log(n)\big).$$

En utilisant le théorème 5.5, on a que

$$f_2(n) = 7n^5 + 4n \in \Theta(n^5)$$

car f_2 est un polynôme de degré 5. Étant donné que $\Theta\left(n^5\right) \subset O\left(n^5\right)$, on obtient

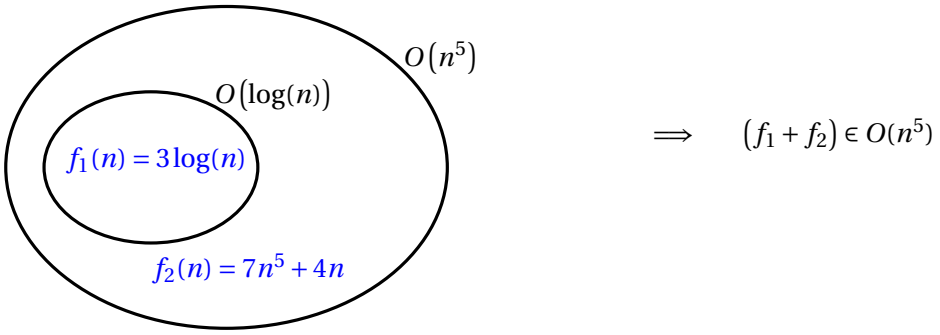
$$f_2(n) = 7n^5 + 4n \in O(n^5).$$

De plus, comme $O\big(\log(n)\big) \subset O(n^5)$, le théorème de la somme nous dit que

$$f(n) = f_1(n) + f_2(n) \in O(n^5).$$

Autrement dit, la fonction recherchée est

$$g(n) = n^5.$$



Exemple 5.13
Soit

$$f(n) = 3\log(n) \cdot \left(7n^5 + 4n\right).$$

Trouvez la fonction $g(n)$ *la plus simple possible* telle que $f(n) \in O(g(n))$.

Solution :
Par l'exemple 5.12, on sait que

$$f_1(n) = 3\log(n) \in O\big(\log(n)\big)$$

et

$$f_2(n) = 7n^5 + 4n \in O(n^5).$$

Donc, par le théorème du produit, on a

$$f(n) = f_1(n) \cdot f_2(n) \in O\left(n^5 \log(n)\right).$$

Autrement dit, la fonction recherchée est

$$g(n) = n^5 \log(n).$$

Exemple 5.14

Soit

$$f(x) = \sqrt{7}x^6 + 7x^5 + \pi x^3 - 194x^2 - 2112$$

Trouvez la fonction $g(x)$ *la plus simple possible* telle que $f(x) \in O(g(x))$.

Solution :

La fonction f est un polynôme de degré 6. Ainsi, par le théorème 5.5,

$$\sqrt{7}x^6 + 7x^5 + \pi x^3 - 194x^2 - 2112 \in O(x^6).$$

Autrement dit, la fonction recherchée est

$$g(x) = x^6.$$

Exemple 5.15

Soit

$$f(x) = x^2 + 5^x$$

Trouvez la fonction $g(x)$ *la plus simple possible* telle que $f(x) \in O(g(x))$.

Solution :

La fonction f est une somme. On peut traduire le théorème de la somme ainsi : *c'est la fonction dont le grand-O domine les autres qui l'emporte.* Dans le cas de f, quelle fonction domine l'autre ? Allons voir la liste du théorème 5.2 : c'est l'exponentielle 5^x qui domine la puissance x^2, au sens où $x^2 \in O(5^x)$. Ainsi,

$$x^2 + 5^x \in O(5^x)$$

La fonction recherchée est donc $g(x) = 5^x$.

Exemple 5.16

Soit

$$f(n) = (14n + 3)\log(n) + 3n^2$$

Trouvez la fonction $g(n)$ *la plus simple possible* telle que $f(n) \in O(g(n))$.

Solution :

On sait que,

$$14n + 3 \in O(n)$$

car c'est un polynôme de degré 1. Donc, par le théorème du produit,

$$(14n + 3)\log(n) \in O(n\log(n))$$

De plus, on a $3n^2 \in O(n^2)$ et en regardant la liste du théorème 5.2, on voit que n^2 domine $n\log(n)$. Ainsi, d'après le théorème sur la somme,

$$(14n + 3)\log(n) + 3n^2 \in O(n^2)$$

La fonction cherchée est donc $g(n) = n^2$.

Exercices

5.1 Pour chaque fonction ci-dessous, déterminez la fonction $g(n)$ *la plus simple possible* telle que $f(n) \in O(g(n))$.

(a) $f(n) = 50\sqrt{n} + 600$

(b) $f(n) = 2n^2 + 50\sqrt{n} + 600$

(c) $f(n) = 20n + 3\log(n)$

(d) $f(n) = \left(n^2 + 1\right)\left(n + 3\log(n)\right)$

(e) $f(n) = \left(n^2 + 3n^4\right)\left(\log(n) + 5\right)$

(f) $f(n) = 2^n + 5^5$

(g) $f(n) = 2^n + 5^n$

(h) $f(n) = 20! + n!$

5.2 Pour chaque fonction ci-dessous, déterminez la fonction $g(n)$ *la plus simple possible* telle que $f(n) \in O(g(n))$.

(a) $f(n) = 2 + n + \dfrac{1}{10}n^2$

(b) $f(n) = (n^2 + 1)(n + 3)$

(c) $f(n) = (n + 1)(n + 2)(n + 3)(n + 4)$

(d) $f(n) = (1 + n^2)(1 + n)^4$

(e) $f(n) = 4n^2 + 100n + 10$

(f) $f(n) = \sqrt{n} + \sqrt{n^3} + \sqrt[3]{n} + \sqrt[3]{n^5} + \sqrt[5]{n^4}$

(g) $f(n) = \sqrt[10]{n} + \log_2(n)$

(h) $f(n) = 2^n + n^2 + \pi$

(i) $f(n) = 100!$

(j) $f(n) = 2^{n+2} + 3^{n+3} + 4^{n+4}$

(k) $f(n) = 2^{3n+1} + 3^{2n+2}$

5.3 Afin de résoudre un problème, vous avez le choix entre deux algorithmes. Le premier résout un problème de taille n avec un nombre d'opérations égal à $f(n) = 2n^{\frac{3}{2}} + 5n + 10$ alors que le deuxième produit le même résultat en $g(n) = 5n\log_5(n^2)$ opérations.

(a) Exprimez $f(n)$ et $g(n)$ en notation grand-O.

(b) Lorsque n est grand, lequel des deux algorithmes est-il préférable d'utiliser ?

5.4 Même question que la précédente avec $f(n) = n(n\log_2(n) + 2)$ et $g(n) = (n + \sqrt{n})(\sqrt[3]{n^2} + 12)$.

5.5 Exprimez les fonctions suivantes en notation grand-O à l'aide d'une fonction *aussi simple que possible* qui reflète son comportement asymptotique.

(a) $f(n) = 5n^4 + 10n^2 + n - 100$

(b) $f(n) = (2n^2 + 1)(\log_2(n) + \sqrt{n})$

(c) $f(n) = (4n^3 + 2n^2 + 15)(25n^4 + 10n - 1)$

(d) $f(n) = 2n^2 + 2^{(n+3)}$

(e) $f(n) = (n^2 + n)\log(n^3 + 3) + 10!n(n^2 + 4)$

(f) $f(n) = 10 \cdot 2^n(2^n + \sqrt{n})$

(g) $f(n) = (25n\log_4(n) + n)(n + 45) + 2^n$

(h) $f(n) = (2^n + \log_3(n))(20 + 2\log_2(n^2))$

(i) $f(n) = (1 + n + n^2)(10 + 8n^2 + 6n^4 + 4n^6 + 2n^8)$

(j) $f(n) = \sqrt{n} + \left(\dfrac{1000}{1001}\right)^n$

(k) $f(n) = n^{100} + \left(\dfrac{1001}{1000}\right)^n$

(l) $f(n) = \log_2(n^3) + \sqrt{n}$

(m) $f(n) = \dfrac{3^n}{2^n}$

(n) $f(n) = \log_2(n)(n^2 + \log_3(n)) + (1 + n + n^2)(\sqrt{n} + 3)$

5.3 Sommations

Dans un algorithme, les boucles peuvent être vues comme des sommations. Les deux théorèmes suivants nous permettant de faire des calculs directement sur celles-ci, ramenant la détermination de la complexité d'un algorithme parfois moins évidente à un calcul mathématique plus simple.

Théorème 5.6 : Manipulation des sommations

Étant donné j, m et n des nombres naturels :

$$\text{(a)} \quad \sum_{i=m}^{n} c = \underbrace{c + c + c + \ldots + c}_{n-m+1 \text{ fois}} = (n - m + 1)c$$

$$\text{(b)} \quad \sum_{i=m}^{n} (a_i + b_i) = (a_m + b_m) + (a_{m+1} + b_{m+1}) + \ldots + (a_n + b_n) = \sum_{i=m}^{n} a_i + \sum_{i=m}^{n} b_i$$

$$\text{(c)} \quad \sum_{i=m}^{n} (ca_i) = (ca_m) + (ca_{m+1}) + \ldots + (ca_n) = c \cdot \sum_{i=m}^{n} a_i$$

$$\text{(d)} \quad \sum_{i=m}^{n} a_i = a_m + a_{m+1} + \ldots + a_n = \sum_{i=j}^{n} a_i - \sum_{i=j}^{m-1} a_i, \quad \text{si } j < m$$

Théorème 5.7 : Formes closes de sommation

Étant donné $n \in \mathbb{N}$ une constante :

$$\text{(a)} \quad \sum_{i=1}^{n} i = 1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}$$

$$\text{(b)} \quad \sum_{i=1}^{n} i^2 = 1^2 + 2^2 + 3^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\text{(c)} \quad \sum_{i=1}^{n} i^3 = 1^3 + 2^3 + 3^3 + \ldots + n^3 = \frac{n^2(n+1)^2}{4}$$

$$\text{(d)} \quad \sum_{i=0}^{n} r^i = r^0 + r^1 + r^2 + \ldots + r^n = \frac{r^{n+1} - 1}{r - 1} \quad \text{si } r \in \mathbb{R} \setminus \{1\}$$

▷ ***Démonstration*** Prouvons la lettre (a). Posons $S_n = 1 + 2 + \cdots + (n-1) + n$. Si on additionne deux fois S_n, on obtient

$$\begin{array}{rcccccccccc} S_n & = & 1 & + & 2 & + & \cdots & + & (n-1) & + & n \\ +\,S_n & = & n & + & (n-1) & + & \cdots & + & 2 & + & 1 \\ \hline 2\,S_n & = & (n+1) & + & (n+1) & + & \cdots & + & (n+1) & + & (n+1) \end{array}$$

Le membre de droite de la dernière équation contient n fois $(n+1)$, d'où

$$2\,S_n = n(n+1)$$

et ainsi

$$S_n = \frac{n(n+1)}{2}.$$

Nous verrons plus loin, à l'exemple 7.1, comment redémontrer ce résultat à l'aide d'une preuve par récurrence. ◁

Exemple 5.17

Trouvez le résultat de la sommation suivante en utilisant les théorèmes 5.6 et 5.7 :

$$\sum_{i=4}^{50}(2+4\,i).$$

Solution :

$$\begin{aligned}
\sum_{i=4}^{50}(2+4\,i) &= \sum_{i=4}^{50}2+\sum_{i=4}^{50}4\,i && \text{par le théorème 5.6 (b)}\\
&= \sum_{i=4}^{50}2+4\sum_{i=4}^{50}i && \text{par le théorème 5.6 (c)}\\
&= (50-4+1)\,2+4\left(\sum_{i=1}^{50}i-\sum_{i=1}^{3}i\right) && \text{par le théorème 5.6 (a) et (d)}\\
&= 94+4\left(\frac{50(50+1)}{2}-\frac{3(3+1)}{2}\right) && \text{par le théorème 5.7 (a)}\\
&= 94+5100-24\\
&= 5\,170
\end{aligned}$$

Exemple 5.18

Donnez la forme close de la sommation suivante en utilisant les théorèmes 5.6 et 5.7 :

$$\sum_{i=3}^{n}\sum_{j=1}^{n-2}2^i.$$

Solution :

$$\begin{aligned}
\sum_{i=3}^{n}\sum_{j=1}^{n-2}2^i &= \sum_{i=3}^{n}\big((n-2)-1+1\big)2^i && \text{par le théorème 5.6 (a)}\\
&= (n-2)\sum_{i=3}^{n}2^i && \text{par le théorème 5.6 (c)}\\
&= (n-2)\left(\sum_{i=0}^{n}2^i-\sum_{i=0}^{2}2^i\right) && \text{par le théorème 5.6 (d)}\\
&= (n-2)\left(\frac{2^{n+1}-1}{2-1}-\frac{2^{2+1}-1}{2-1}\right) && \text{par le théorème 5.7 (d)}\\
&= (n-2)\left(2^{n+1}-1-(2^3-1)\right)\\
&= (n-2)\left(2^{n+1}-8\right)
\end{aligned}$$

Exercices

5.6 Soit

$$f(n) = \sum_{i=2}^{n-2} (n+2i).$$

(a) Détaillez tous les termes de la sommation ci-dessus pour le cas où $n = 6$. Calculez la valeur numérique de la somme.

(b) •Exprimez $f(n)$ sous forme close (c'est-à-dire une formule qui ne dépend que de n, sans utiliser le symbole de sommation).
•Citez les théorèmes appropriés.
•Simplifiez votre réponse.
•Validez votre résultat avec le cas $n = 6$.

5.7 Dans chacun des cas suivants, exprimez $f(n)$ sous forme close (c'est-à-dire sans utiliser le symbole de sommation ni les « . . . »). Tous les calculs doivent être faits **à la main** en utilisant les théorèmes 5.6 et 5.7. Simplifiez votre réponse.

(a) $f(n) = \sum_{i=1}^{n} (n+i+1)$

(b) $f(n) = \sum_{i=0}^{n+1} (5n+3i+4)$

(c) $f(n) = \sum_{i=n}^{2n} (i+1) - n$

(d) $f(n) = \sum_{i=2}^{n-1} 4^i$

(e) $f(n) = \sum_{i=1}^{2n-1} \sum_{j=0}^{n-1} (j+1)$

(f) $f(n) = \sum_{i=0}^{n-1} \sum_{j=1}^{n} (n^2+i+j)$

(g) $f(n) = \sum_{k=0}^{n} \left(\sum_{l=1}^{n^2} (k-l) + k^2 \right)$

(h) $f(n) = \sum_{k=10}^{n} 2 \cdot 5^k$

5.4 Établir la fonction de complexité d'un algorithme

Dans le calcul de la complexité d'un algorithme, nous devons d'abord construire une fonction $f(n)$ représentant le nombre d'opérations effectuées lors de l'appel de l'algorithme avec une donnée de taille n. Cette fonction peut ensuite être identifiée à l'une des différentes classes de complexité suivantes :

$\Theta(1)$	Constante
$\Theta(\log(n))$	Logarithmique
$\Theta(n)$	Linéaire
$\Theta(n\log(n))$	$n\log(n)$
$\Theta(n^a)$	Polynomiale
$\Theta(a^n)$	Exponentielle
$\Theta(n!)$	Factorielle

Exemple 5.19

Comptez le nombre de comparaisons effectuées lors de l'appel de l'algorithme suivant. Quelle est sa complexité?

```
1: algorithme TriBulles(T tableau de 1 à n d'éléments)
2:     pour i = 1 à n − 1 faire
3:         pour j = 1 à n − i faire
4:             si T[j] > T[j + 1] alors
5:                 échanger T[j] et T[j + 1]
6:             fin si
7:         fin pour
8:     fin pour
9: fin algorithme
```

Solution :

$$\sum_{i=1}^{n-1}\sum_{j=1}^{n-i} 1 = \frac{n^2}{2} - \frac{n}{2} \in \Theta(n^2)$$

Exemple 5.20

Comptez le nombre de comparaisons effectuées lors de l'appel de l'algorithme suivant. Quelle est sa complexité?

```
1: algorithme TriSelection(T tableau de 1 à n d'entiers)
2:     pour i = 1 à n − 1 faire
3:         m := i
4:         pour j = i + 1 à n faire
5:             si T[j] < T[m] alors
6:                 m := j
7:             fin si
8:         fin pour
9:         échanger T[i] et T[m]
10:    fin pour
11: fin algorithme
```

Solution :

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} 1 = \frac{n^2}{2} - \frac{n}{2} \in \Theta(n^2)$$

Exercices

5.8 Donnez la complexité de la portion de l'algorithme suivant en comptant le nombre d'additions, en fonction de n.

```
1: i := 2
2: tant que i ≤ n faire
3:     a[i] := a[i] + 3 + n
4:     pour j de 1 à n − i faire
5:         a[i] := a[i] + 2j
6:     fin pour
7:     i := i + 1
8: fin tant que
```

5.9 On suppose que le type *élément* a été défini. Soit $f(n)$ le nombre de comparaisons d'éléments effectuées par un appel à la fonction `P` *au pire cas*, où n est la taille du tableau T.

```
1: fonction P(T : tableau de taille n, x : élément)
2:     trouvé := Faux
3:     pour i de 0 à 9 faire
4:         si T[i] == x alors
5:             trouvé := vrai
6:         fin si
7:     fin pour
8:     retourner trouvé
9: fin fonction
```

Donnez $f(n)$ et sa complexité. De plus, expliquez en vos mots ce que retourne la fonction `P`.

5.10 On suppose que le type *élément* a été défini. Soit $f(n)$ le nombre de comparaisons d'éléments effectuées par un appel à la fonction `P` *au pire cas*, où n est la taille du tableau T.

Considérons que les sous-fonctions `Traitement1` et `Traitement2` ont été définies et qu'elles requièrent respectivement $f_1(n) = 5n$ et $f_2(n) = \frac{n(n+1)}{2}$ comparaisons d'éléments.

```
1: fonction P(T : tableau de taille n, x : élément)
2:     trouvé := Faux
3:     pour i de 0 à 9 faire
4:         Traitement1(T)
5:     fin pour
6:     Traitement2(T)
7:     retourner trouvé
8: fin fonction
```

Donnez $f(n)$ et sa complexité. Vous *n'avez pas* à expliquer ce que fait la fonction `P`.

5.11 On suppose que le type *élément* a été défini. Soit $f(n)$ le nombre de comparaisons d'éléments effectuées par un appel à la fonction `P` *au pire cas,* où n est la taille du tableau T.

Considérons que les sous-fonctions `Traitement1` et `Traitement2` ont été définies et qu'elles requièrent respectivement $f_1(n) = 5n$ et $f_2(n) = \frac{n(n+1)}{2}$ comparaisons d'éléments.

```
1: fonction P(T : tableau de taille n, x : élément)
2:     trouvé := Faux
3:     pour i de 0 à n − 1 faire
4:         Traitement2(T)
5:     fin pour
6:     Traitement1(T)
7:     retourner trouvé
8: fin fonction
```

Donnez $f(n)$ et sa complexité.

5.12 Dans chacun des cas ci-dessous, donnez $f(n)$ sous la forme d'une sommation, puis sous forme close et finalement en notation grand-O.

Remarques: Pour cet exercice, tous les tableaux et les chaînes de caractères sont indicés de 0 à $n-1$.

(a) La fonction `estPrésent` détermine si l'élément x est présent dans le tableau d'éléments T. On suppose que le type *élément* a été défini. Soit $f(n)$ le nombre de comparaisons d'éléments effectuées par un appel à `estPrésent` *au pire cas,* où n est la taille du tableau T.

```
1: fonction estPrésent(T : tableau de taille n, x : élément)
2:     trouvé := Faux
3:     pour i de 0 à n − 1 faire
4:         si T[i] == x alors
5:             trouvé := vrai
6:         fin si
7:     fin pour
8:     retourner trouvé
9: fin fonction
```

(b) Voici une deuxième version de la fonction `estPrésent`, $f(n)$ est défini comme en (a).

```
1: fonction estPrésent(T : tableau de taille n, x : élément)
2:     trouvé := Faux
3:     i := 0
4:     tant que i < n et non trouvé faire
5:         trouvé := (T[i] == x)
6:         i := i + 1
7:     fin tant que
8:     retourner trouvé
9: fin fonction
```

(c) La fonction `nbOccurrences` compte le nombre d'occurrences de l'élément x dans le tableau d'éléments T. Soit $f(n)$ le nombre de comparaisons d'éléments effectuées par un appel à `nbOccurrences` *au pire cas*, où n est la taille du tableau T.

```
1: fonction nbOccurrences(T : tableau de taille n, x : élément)
2:     compte := 0
3:     pour i de 0 à n − 1 faire
4:         si T[i] == x alors
5:             compte := compte + 1
6:         fin si
7:     fin pour
8:     retourner compte
9: fin fonction
```

(d) La fonction `contientDoublon` détermine si le tableau T contient deux fois la même valeur. Soit $f(n)$ le nombre de comparaisons de cases du tableau effectuées par un appel à `contientDoublon` *au pire cas*, où n est la taille du tableau T.

```
1: fonction contientDoublon(T : tableau de taille n)
2:     trouvé := faux
3:     pour i de 0 à n − 2 faire
4:         pour j de i + 1 à n − 1 faire
5:             si T[i] == T[j] alors
6:                 trouvé := vrai
7:             fin si
8:         fin pour
9:     fin pour
10:    retourner trouvé
11: fin fonction
```

(e) ★ La fonction `fusion` prend en entrée deux tableaux triés $T1$, $T2$ et produit un tableau trié contenant tous les éléments de $T1$ et $T2$. Soit $f(n)$ le nombre de comparaisons de cases du tableau effectuées par un appel à `fusion` *au pire cas*, où n est la taille du plus grand des deux tableaux passés en paramètre.
(Aide : au pire cas, les deux tableaux sont de taille n)

```
1:  fonction fusion(T1 : Tableau de taille n1, T2 : Tableau de taille n2)
2:      i, j, k := 0
3:      T := tableau d'entier de taille n1 + n2
4:      tant que k < n1 + n2 faire
5:          si j == n2 ou (i < n1 et T1[i] ≤ T2[j]) alors
6:              T[k] := T1[i]
7:              i := i + 1
8:          sinon
9:              T[k] := T2[j]
10:             j := j + 1
11:         fin si
12:         k := k + 1
13:     fin tant que
14:     retourner T
15: fin fonction
```

(f) ★ ★ La fonction `recherche` détermine si la chaîne de caractères M (motif) apparaît dans la chaîne de caractères T (texte). Soit $f(n)$ le nombre de comparaisons de caractères effectuées par un appel à `recherche` *au pire cas*, où n est la taille du texte T.

Aide : pour analyser cette fonction *au pire cas*, on suppose que n est pair, que T est une lettre répétée n fois et que M est cette même lettre répétée $n/2-1$ fois suivies d'une lettre différente. Par exemple, $T = \underbrace{\texttt{aaa}\cdots\texttt{a}}_{n \text{ fois}}$ avec n pair et que $M = \underbrace{\texttt{aaa}\cdots\texttt{a}}_{\frac{n}{2}-1 \text{ fois}}\texttt{b}$.

```
1:  fonction recherche(T : Chaîne, M : Chaîne)
2:      lngT := Taille de T
3:      lngM := Taille de M
4:      trouvé := Faux
5:      i := 0
6:      tant que i ≤ (lngT − lngM) et non trouvé faire
7:          j := 0
8:          tant que (j < lngM et T[i + j] == M[j]) faire
9:              j := j + 1
10:         fin tant que
11:         si j == lngM alors
12:             trouvé := vrai
13:         fin si
14:         i := i + 1
15:     fin tant que
16:     retourner trouvé
17: fin fonction
```

5.5 Calculabilité et complexité

Il existe des problèmes qui ne peuvent être résolus par un programme.

En 1936, Alan Turing a fourni un exemple concret en fournissant un exemple de problème qui ne peut pas être résolu par un programme : le fameux problème de l'arrêt. Il serait donc totalement inutile de mettre sur pied une équipe d'ingénieurs en informatique pour tenter de résoudre un tel problème, comme il serait inutile de mettre sur pied une équipe d'ingénieurs en mécanique pour construire une machine à mouvement perpétuel.

Quels sont les problèmes résolubles ? Non résolubles ? Voilà une des questions que traite la théorie de la calculabilité.

De plus, parmi les problèmes résolubles, ***quels sont les problèmes résolubles par un algorithme efficace ?*** Voilà une autre question importante, traitée par la théorie de la complexité. Mais qu'entend-on au juste par algorithme *efficace* ? Il est clair qu'un algorithme dont la complexité est exponentielle n'est *pas* efficace. Il est plus difficile de s'entendre sur le sens du mot *efficace,* mais on convient généralement qu'il s'agit d'un algorithme dont la complexité est au plus polynomiale : $\Theta(n^d)$ peu importe le degré. Dans les faits, le degré du polynôme devra être petit pour que l'algorithme soit réellement efficace, mais ceci n'est pas important ici.

Il existe toute une gamme de problèmes importants pour lesquels aucun algorithme à complexité polynomiale n'a été trouvé. Et plusieurs de ces problèmes, quoique fort différents, sont équivalents au sens ou si l'un de ces problèmes est résoluble en temps polynomial, alors ils le sont tous. Ces problèmes sont qualifiés de **NP-complets**. En voici quelques-uns.

1. En théorie des graphes, le problème du circuit hamiltonien (HC) : existe-t-il un circuit fermé permettant de parcourir chaque sommet d'un graphe une et une seule fois ?
2. En théorie des graphes, la version décisionnelle du problème du voyageur de commerce (« travelling salesman », TS) : pour une distance d, existe-t-il un chemin plus court que d permettant au voyageur de visiter toutes villes et de revenir à son point de départ ?
3. En logique, le problème de la satisfaisabilité d'une proposition (SAT) : étant donnée une proposition sous forme normale conjonctive, par exemple

   $$(p \vee \neg q \vee s) \ \wedge \ (\neg p \vee q \vee s),$$

   existe-t-il une fonction d'interprétation qui la rend vraie ? En d'autres mots, y a-t-il un choix de valeurs de vérité pour les propositions simples p, q et s qui rend la proposition composée vraie ?

Il existe un algorithme très simple qui permet de répondre à cette dernière question : vérifier si la table de vérité de la proposition composée contient une ligne Vrai. Mais comme nous l'avons vu au chapitre 1, si la proposition contient n propositions simples, la table de vérité comporte 2^n lignes. L'algorithme décrit n'est donc malheureusement pas polynomial.

Il en va de même des autres problèmes : on peut les résoudre en énumérant toutes les possibilités puis en vérifiant, mais l'énumération est d'ordre exponentiel ou factoriel alors que l'on recherche un algorithme polynomial.

Une autre caractéristique commune de ces problèmes est que, bien qu'il soit difficile (ou impossible ?) de fournir un algorithme qui produit une solution en temps polynomial, il est facile de fournir un algorithme qui vérifie en temps polynomial si une solution donnée est valide.

Les problèmes NP-complets : très longs à résoudre, très rapides à vérifier.

Une question à un million de dollars!

Le fait qu'à ce jour aucun des problèmes NP-complets n'ait été résolu en temps polynomial nous porte à croire que cela est impossible. Mais l'impossibilité de les résoudre en temps polynomial n'a pas été démontrée. Cette question ouverte constitue d'ailleurs un des sept *problèmes du millénaire* pour chacun desquels un million de dollars est offert en récompense par le Clay Mathematics Institute of Cambridge. Au moment d'écrire ces lignes, seulement un des sept problèmes a été résolu.

Cela ne veut pas dire qu'il faille abandonner toute tentative de résoudre les problèmes NP-complets par des algorithmes efficaces. Par exemple, si le problème en est un d'optimisation, il est peut-être impossible de trouver un algorithme efficace produisant *la* solution optimale, mais possible d'en trouver un produisant une solution *très proche* de la solution optimale. Parfois, il sera possible de trouver un algorithme fonctionnant efficacement pour la grande majorité des cas traités et on s'en contentera.

Si vous êtes confrontés à un problème et que vous établissez qu'il est NP-complet, vous saurez qu'il vaudra mieux chercher une solution approximative. À moins de révolutionner l'informatique et de devenir millionnaire!

Chapitre 6

Algorithmes récursifs

En programmation, on dit qu'une fonction est **récursive** si elle s'appelle elle-même. Par exemple, étant donné un ensemble non vide, la fonction suivante génère la liste de toutes les permutations des éléments de cet ensemble.

```
1:  fonction Permutations(A : Ensemble non vide)
2:      si |A| == 1 alors
3:          retourner la liste formée de l'unique élément de A
4:      sinon
5:          L = liste vide
6:          pour chaque x ∈ A faire
7:              pour chaque permutation p dans Permutations(A \ {x}) faire
8:                  ajouter x au début de p
9:                  ajouter p à L
10:             fin pour
11:         fin pour
12:         retourner L
13:     fin si
14: fin fonction
```

On peut vérifier que :

- `Permutations`({1}) retourne [1].
- `Permutations`({1,2}) retourne [12, 21].
- `Permutations`({1,2,3}) retourne [123, 132, 213, 231, 312, 321].
- `Permutations`({1,2,3,4}) retourne [1234, 1243, 1324, 1342, 1423, 1432, 2134, 2143, 2314, 2341, 2413, 2431, 3124, 3142, 3214, 3241, 3412, 3421, 4123, 4132, 4213, 4231, 4312, 4321].

De manière plus générale, on dit qu'un algorithme est **récursif** si, pour résoudre un problème d'une certaine taille, il trouve d'abord la solution à une ou plusieurs instances plus petites du même problème. Dans le cas de la fonction `Permutations`, pour générer la liste des permutations de n éléments, la fonction effectue n appels récursifs et chacun de ces appels récursifs génère la liste des permutations de $n-1$ éléments. Finalement, on remarque un cas particulier : si A contient un seul élément alors la solution est obtenue directement, il n'y a pas d'appels récursifs. Il s'agit du **cas de base**.

6.1 Définition et exemples d'algorithmes récursifs

Définition 6.1 : Algorithme récursif

Un algorithme est **récursif** s'il permet de résoudre un problème en le réduisant au même problème avec une ou plusieurs entrées de plus petites tailles (et donc avec un ou plusieurs appels à lui-même).

Pour analyser la complexité d'un algorithme récursif, on définit une relation de récurrence (ou fonction récursive) qui compte le nombre d'opérations effectuées lors de l'appel de celui-ci. On trouve ensuite une forme close pour son terme général, ce qui nous permet de déterminer sa complexité.

Exemple 6.1

Soit $a \in \mathbb{R}$ et $n \in \mathbb{N}$.

(a) Implémentez sur *Nspire* une fonction récursive pour calculer a^n, où $a \neq 0$ et $n \in \mathbb{N}$. Cette fonction retournera `undef` si $a = 0$ et $n = 0$.

(b) Donnez sa complexité. *Comptez le nombre de multiplications effectuées.*

Solution :

(a) Voici une fonction puissance définie avec l'éditeur de programme, puis une fonction p tout à fait équivalente définie directement dans une fenêtre de calculs avec une fonction par morceaux, suivie de quelques tests.

```
"puissance" enregistr. effectué
Define puissance(a,n)=
Func
© a réel, n naturel; retourne a^n ou undef si a=0 et n=0 ou si n∉N.
If a=0 and n=0 or n<0 or mod(n,1)≠0 Then
 Return undef
Else
  If n=0 Then
    Return 1
  Else
    Return a· puissance(a,n−1)
  EndIf
EndIf
EndFunc
```

$$p(a,n):=\begin{cases}\text{undef}, & a=0 \text{ and } n=0 \text{ or } n<0 \text{ or } \mathrm{mod}(n,1)\neq 0\\ 0, & a=0 \text{ and } n\neq 0\\ 1, & a\neq 0 \text{ and } n=0\\ a\cdot p(a,n-1), & \text{Else}\end{cases}$$

Terminé

$$tests:=\{p(0,0),p(8,-2),p(5,1.3),p(8,0),p(8,1),p(2,4),p(-3,2)\}$$

$$\{\text{undef},\text{undef},\text{undef},1,8,16,9\}$$

$$p(2,8) \qquad 256$$

$$p\left(\frac{2}{5},3\right) \qquad \frac{8}{125}$$

(b) Soit $f(n)$ la fonction qui compte le nombre de multiplications effectuées lors d'un appel à **`puissance`**(a,n). En analysant le code de la fonction **`puissance`**, on conclut que f est définie par :

$$f(n) = \begin{cases} 0 & \text{si } n = 0, \\ f(n-1)+1 & \text{sinon.} \end{cases}$$

Afin d'obtenir une forme close pour f, on en calcule les premières valeurs, puis on généralise.

$$\begin{aligned}
f(0) &= 0\\
f(1) &= f(0)+1\\
f(2) &= f(1)+1 = \big(f(0)+1\big)+1 = f(0)+2\\
f(3) &= f(2)+1 = \big(f(0)+2\big)+1 = f(0)+3\\
f(4) &= f(3)+1 = \big(f(0)+3\big)+1 = f(0)+4\\
&\vdots\\
f(n) &= f(0)+n
\end{aligned}$$

La complexité de la fonction **`puissance`** est donc :

$$f(n) = n \in O(n).$$

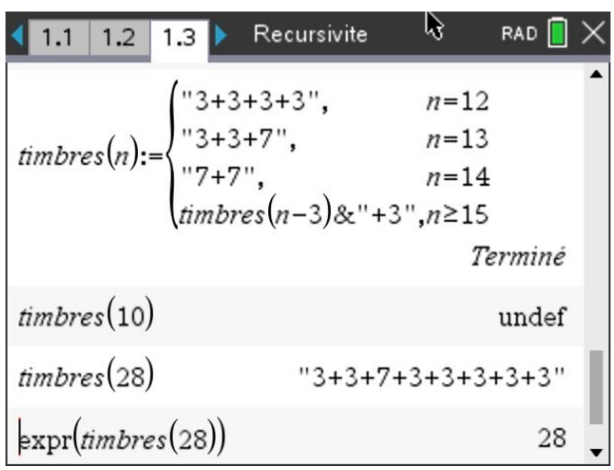
Exemple 6.2

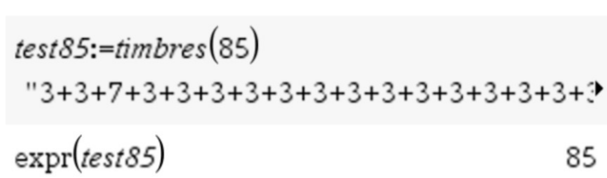
Programmez une fonction récursive qui retournera une combinaison de timbres totalisant n ¢, pour $n \geq 12$, en utilisant uniquement des timbres de 3 ¢ et de 7 ¢. Cette combinaison pourrait être donnée sous la forme d'une chaîne de caractères séparés par des virgules ou des +.

Information utile : la concaténation de chaînes de caractères en TI-Basic s'effectue avec le symbole &.

Solution :

Voici une fonction définie à l'aide d'une fonction par morceaux, suivie de quelques tests. Notez que la commande `expr( )` transforme une chaîne de caractères en expression mathématiques. Par exemple, `expr("1+1")` retournera la valeur 2.





Exemple 6.3

Donnez une fonction récursive qui déterminera si oui ou non l'élément x est présent dans la liste t. Pour ce faire, si la liste possède plus d'un élément, la fonction séparera la liste en 2 sous-listes et vérifiera si x est présent dans chacune d'elle. Programmez cette fonction sur *Nspire*.

Information utile: en TI-Basic, la commande `left(l,n)` retourne la sous-liste des n premiers éléments de l, la commande `right(l,n)` retourne la sous-liste des n derniers éléments de l et la commande `dim(t)` retourne la dimension de la liste t. Rappelons aussi que les listes sont indexées à partir de 1.

Solution :

Voici une fonction définie avec l'éditeur de programme, puis une fonction p tout à fait équivalente définie directement dans une fenêtre de calculs avec une fonction par morceaux, suivie de quelques tests.

```
est_present                                                    10/10
Define est_present(x,t)=
Func
© x élément recherché, t liste: détermine si x est dans t
Local n,gauche,droite
© n est la dimension de la liste t, gauche est la première moitié de la
  liste t et droite est l'autre moitié de la liste t.
n:=dim(t)
If n=1 Then
  Return x=t[1]
Else
  gauche:=left(t,ceiling(n/2)):  droite:=right(t,floor(n/2))
  Return est_present(x,gauche) or est_present(x,droite)
EndIf
EndFunc
```

$$p(x,t):=\begin{cases} x=t[1], & \dim(t)=1 \quad \textit{Terminé} \\ p\left(x,\text{left}\left(t,\text{ceiling}\left(\frac{\dim(t)}{2}\right)\right)\right) \text{ or } p\left(x,\text{right}\left(t,\text{floor}\left(\frac{\dim(t)}{2}\right)\right)\right), & \text{Else} \end{cases}$$

$p(123,\{9,11,17,125,0\})$	false
$p(125,\{9,11,17,125,0\})$	true
$t2:=\{4,4,45,4,75,457,1,8,90,5,3,76\}$	$\{4,4,45,4,75,457,1,8,90,5,3,76\}$
$\dim(t2)$	12
$p(90,t2)$	true
$p(91,t2)$	false

6.2 Fonctions récursives et relations de récurrence

Lorsqu'on considère des fonctions au sens mathématique, tel que vu au Chapitre 1, on définit la récursivité de la façon suivante.

Définition 6.2 : Fonction récursive

Une **fonction récursive** est une fonction dont la définition contient un ou plusieurs appels à elle-même. Si f est une fonction dont le domaine est $\mathbb{N}$, on procède ainsi pour la définir :

Cas de base :	définir $f(n)$ pour une ou plusieurs valeurs initiales (comme par exemple 0, 1 et 2 s'il y a 3 cas de base)
Étape récursive :	donner une règle pour calculer $f(n)$ à l'aide de $f(0), f(1), \ldots, f(n-2), f(n-1)$.

Note : on écrit parfois f_n au lieu de $f(n)$.

Exemple 6.4 (à compléter en classe)
Définir récursivement les fonctions suivantes, où $n \in \mathbb{N}$:

(a) $f(n) = n!$

(b) $g(n) = a^n$, où $a \in \mathbb{R}$, $a \neq 0$.

Solution :

Définition 6.3 : Relation de récurrence, terme général, solution

Une **relation de récurrence** d'une suite $\{a_n\}$ est une équation qui exprime le **terme général** a_n en fonction de $a_0, a_1, \ldots a_{n-1}$. On dit qu'une suite est **solution** de la récurrence si ses termes satisfont la relation de récurrence.

Exemple 6.5
Le suite de Fibonacci $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$ est définie par la relation de récurrence

Cas de base : $f_0 = 0, \quad f_1 = 1$
Relation de récurrence (ou étape récursive) : $f_n = f_{n-1} + f_{n-2}, \quad n \geq 2.$

Sous forme de fonction récursive, ceci est noté

$$f(n) = \begin{cases} 0 & \text{si } n = 0, \\ 1 & \text{si } n = 1, \\ f(n-1) + f(n-2) & \text{si } n \geq 2. \end{cases}$$

Une relation de récurrence est donc une définition récursive des termes de la suite $\{a_n\}$. Formellement, une suite est une fonction f dont le domaine est un sous-ensemble des entiers et on note $f(n) = a_n$ son terme général. Une relation de récurrence peut donc être vue comme la définition récursive d'une fonction.

6.2.1 Résolution de relation de récurrence par la méthode itérative

Lorsque l'on considère des relations de récurrences simples, il est parfois possible de *deviner* une forme close pour la solution en itérant directement la relation de récurrence dans sa propre définition. C'est ce qu'on appelle la **méthode itérative**.

Exemple 6.6

Résolvez la relation de récurrence $a_0 = 2; \quad a_n = a_{n-1} + 3, \quad n \geq 1$.
C'est-à-dire : trouvez une forme close, en fonction de n, pour son terme général.

Solution :
On trouve

$$\begin{aligned}
a_0 &= 2 \\
a_1 &= a_0 + 3 \\
a_2 &= a_1 + 3 = (a_0 + 3) + 3 = a_0 + 2 \cdot 3 \\
a_3 &= a_2 + 3 = (a_0 + 2 \cdot 3) + 3 = a_0 + 3 \cdot 3 \\
a_4 &= a_3 + 3 = (a_0 + 3 \cdot 3) + 3 = a_0 + 4 \cdot 3 \\
&\vdots \\
a_n &= a_0 + n \cdot 3 = a_0 + 3n
\end{aligned}$$

En utilisant $a_0 = 2$ donné par la définition, on obtient la solution : $a_n = 3n + 2$.

Remarque : à l'exemple 7.4 nous verrons comment prouver que $a_n = 3n + 2$ est bien une solution.

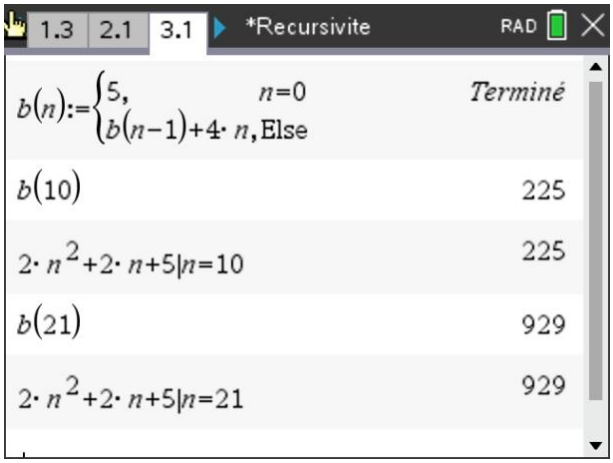
Exemple 6.7

Résolvez la relation de récurrence $b_0 = 5; \quad b_n = b_{n-1} + 4n, \quad n \geq 1$.
C'est-à-dire : trouvez une forme close, en fonction de n, pour son terme général.

Solution :
On trouve

$$\begin{aligned}
b_0 &= 5 \\
b_1 &= b_0 + 4 \cdot 1 \\
b_2 &= b_1 + 4 \cdot 2 = (b_0 + 4 \cdot 1) + 4 \cdot 2 = b_0 + 4(1 + 2) \\
b_3 &= b_2 + 4 \cdot 3 = (b_0 + 4(1 + 2)) + 4 \cdot 3 = b_0 + 4(1 + 2 + 3) \\
b_4 &= b_3 + 4 \cdot 4 = (b_0 + +4(1 + 2 + 3)) + 4 \cdot 4 = b_0 + 4(1 + 2 + 3 + 4) \\
&\vdots \\
b_n &= b_0 + 4(1 + 2 + 3 + 4 + \cdots + n) = 5 + 4\frac{n(n+1)}{2} = 2n^2 + 2n + 5
\end{aligned}$$

Remarque : à l'exercice 7.3 nous prouverons que la solution obtenue est bien la bonne. Pour l'instant, nous pouvons tester la forme close obtenue avec quelques valeurs de n.



Exercices

6.1 Déterminez les valeurs de $f(n)$ pour n allant de 1 à 7, inclusivement. De plus, décrivez dans vos mots ce que calcule la fonction f.

(a) $f(0) = 0, \quad f(n) = f(n-1) + 2$, pour $n \geq 1$.

(b) $f(0) = 1, \quad f(n) = f(n-1) + 2$, pour $n \geq 1$.

(c) $f(n) = \begin{cases} 0 & \text{si } n = 0, \\ 1 & \text{si } n = 1, \\ f(n-2) & \text{si } n \geq 2. \end{cases}$

(d) $f(0) = 0, \quad f(n) = f(n-1) + 2n - 1$, pour $n \geq 1$.

(e) $f(1) = 1, \quad f(n) = f(n-1) + 10^{n-1}$, pour $n \geq 1$.

(f) ★ $f(n) = \begin{cases} 0 & \text{si } n = 1 \\ g(n, n-1) & \text{si } n \geq 2 \end{cases}$ et $g(a,b) = \begin{cases} 1 & \text{si } b = 1, \\ 0 & \text{si } b \geq 2 \text{ et } a \bmod b = 0, \\ g(a, b-1) & \text{si } b \geq 2 \text{ et } a \bmod b \neq 0. \end{cases}$

6.2 Utilisez la méthode itérative afin de trouver une solution aux relations de récurrence suivantes, en procédant à tous les calculs à la main :

(a) $a_0 = 1, \quad a_n = 2a_{n-1}, \quad$ pour $n \geq 1$.

(b) $b_0 = 0, \quad b_n = b_{n-1} + 2n, \quad$ pour $n \geq 1$.

(c) $c_0 = 1, \quad c_n = c_{n-1} + n + 2, \quad$ pour $n \geq 1$.

(d) $f(0) = 2, \quad f(n) = 3f(n-1) + 2, \quad$ pour $n \geq 1$.

(e) $f(0) = 5, \quad f(n) = 3f(n-1) + 2, \quad$ pour $n \geq 1$.

(f) $f(0) = 5, \quad f(n) = nf(n-1), \quad$ pour $n \geq 1$.

(g) $h_0 = 3, \quad h_n = 10h_{n-1}, \quad$ pour $n \geq 1$.

(h) $f(2) = 3, \quad f(n) = 4f(n-1) + 6, \quad$ pour $n \geq 3$.

6.3 Algorithmes de type diviser pour régner

Nous avons vu à l'exemple 6.1 que lorsqu'on analyse un algorithme récursif, on obtient généralement une fonction de complexité qui est elle-même récursive. Par exemple, on considère les deux algorithmes récursifs suivants.

```
1: fonction Fibo( n : Entier )
2:     si n == 0 alors
3:         retourner 0
4:     sinon si n == 1 alors
5:         retourner 1
6:     sinon
7:         retourner Fibo(n − 1) + Fibo(n − 2)
8:     fin si
9: fin fonction
```

```
1: fonction Somme( t : Tableau d'entiers de taille n )
2:     si n == 1 alors
3:         retourner t[1]
4:     sinon
5:         S_g = Somme(t[1..n/2])        // moitié gauche
6:         S_d = Somme(t[n/2+1..n])      // moitié droite
7:         retourner S_g + S_d
8:     fin si
9: fin fonction
```

Soit $f(n)$ le nombre d'additions effectuées par l'algorithme `Fibo` pour une entrée n et $g(n)$ le nombre d'additions effectuées par l'algorithme `Somme` pour un tableau de taille n. Alors

$$f(n) = \begin{cases} 0 & \text{si } n = 0 \text{ ou } n = 1, \\ f(n-1) + f(n-2) + 1 & \text{si } n \geq 2, \end{cases}$$

$$g(n) = \begin{cases} 0 & \text{si } n = 1, \\ 2g(n/2) + 1 & \text{si } n \geq 2. \end{cases}$$

Ces deux fonctions de complexité présentent une différence fondamentale :

- pour la fonction f, le paramètre des termes récursifs est n auquel on **soustrait une constante** ;
- pour la fonction g, le paramètre des termes récursifs est n **divisé par une constante**.

Nous avons vu, à la section 6.2.1, comment s'y prendre pour résoudre certaines relations de récurrence du type de la fonction f par méthode itérative. De façon très similaire, nous verrons à la section 6.3.2 comment résoudre certaines relations de récurrence du type de la fonction g par la méthode itérative.

6.3.1 Algorithmes et relations de récurrence de type diviser pour régner

Définition 6.4 : Algorithme de type diviser pour régner (ou algorithme de fractionnement)

Un **algorithme de type diviser pour régner** (ou algorithme de fractionnement) est un algorithme qui procède de la façon suivante. Étant donné un problème à résoudre, il le fractionne en sous-problèmes dont la taille est une fraction de la taille originale. Le fractionnement est appliqué récursivement jusqu'à l'obtention d'un cas de base qu'il résout directement. Finalement, les solutions aux sous-problèmes sont combinées pour former la solution au problème initial.

Définition 6.5 : Relation de récurrence de type diviser pour régner

Considérons un algorithme de fractionnement qui procède de la façon suivante pour résoudre un problème de taille n.

- L'algorithme fractionne le problème de taille n en a sous problèmes de taille n/b.
- Il traite les a sous problèmes, puis recombine les solutions pour ainsi obtenir la solution au problème de taille n. On note $g(n)$ le nombre d'opérations requises pour cette étape de recombinaison.
- On note m le nombre d'opérations requises pour traiter un problème de taille 1.

Si l'entier n est une puissance de b, alors le nombre d'opérations pour résoudre le problème de taille n, noté $f(n)$, satisfait la relation de récurrence

$$f(n) = \begin{cases} m & \text{si } n = 1, \\ af(n/b) + g(n) & \text{si } n \geq b. \end{cases}$$

Pour cette raison, une telle relation est donc appelée **relation de type diviser pour régner**.

Pour le reste de cette section, on supposera toujours que n est une puissance de b. Cette hypothèse permet de simplifier grandement les calculs sans affecter le résultat lorsqu'on passe à la notation grand-O.

6.3.2 Résolution de relation de type diviser pour régner par la méthode itérative

En reprenant la méthode itérative présentée à la section 6.2.1, nous présentons ici une manière de résoudre les relations de récurrence par fractionnement.

Exemple 6.8

Considérons la fonction de complexité de la fonction **Somme** présentée à la page 192.

$$g(n) = \begin{cases} 0 & \text{si } n = 1, \\ 2g(n/2) + 1 & \text{si } n \geq 2. \end{cases}$$

Trouvez une forme close pour $g(n)$ lorsque n est une puissance de 2 et donnez sa complexité.

Solution :

Bien que $g(1) = 0$, nous écrirons explicitement le terme $g(1)$ jusqu'à la fin pour que l'exemple demeure le plus général possible.

n	$g(n)$		
$1 = 2^0$	$g(1)$	$=$	0
$2 = 2^1$	$g(2)$	$=$	$2g(1)+1$
$4 = 2^2$	$g(4)$	$=$	$2g(2)+1 = 2\big(2g(1)+1\big)+1$
		$=$	$2^2 g(1) + 2\cdot 1 + 1$
$8 = 2^3$	$g(8)$	$=$	$2g(4)+1 = 2\big(2^2 g(1)+2\cdot 1+1\big)+1$
		$=$	$2^3 g(1) + 2^2\cdot 1 + 2\cdot 1 + 1$
$16 = 2^4$	$g(16)$	$=$	$2g(8)+1 = 2\big(2^3 g(1)+2^2\cdot 1+2\cdot 1+1\big)+1$
		$=$	$2^4 g(1) + 2^3\cdot 1 + 2^2\cdot 1 + 2\cdot 1 + 1$
$32 = 2^5$	$g(32)$	$=$	$2g(16)+1 = 2\big(2^4 g(1)+2^3\cdot 1+2^2\cdot 1+2\cdot 1+1\big)+1$
		$=$	$2^5 g(1) + 2^4\cdot 1 + 2^3\cdot 1 + 2^2\cdot 1 + 2\cdot 1 + 1$
$n = 2^k$	$g(n)$	$=$	$2^k g(1) + 2^{k-1}\cdot 1 + \cdots + 2^3\cdot 1 + 2^2\cdot 1 + 2\cdot 1 + 1$
		$=$	$2^k g(1) + \sum_{i=0}^{k-1} 2^i$
		$=$	$2^k g(1) + \dfrac{2^{k-1+1}-1}{2-1}$ par théorème 5.7(d)
		$=$	$n\cdot g(1) + n - 1$
		$=$	$n\cdot 0 + n - 1$

Ainsi, $g(n) = n - 1 \in O(n)$.

Exemple 6.9

Lorsqu'elle est implémentée de façon récursive, la fouille dichotomique présentée à l'exemple 5.5 produit la fonction de complexité suivante :

$$f(n) = \begin{cases} 1 & \text{si } n = 1, \\ f(n/2) + 1 & \text{si } n \geq 2. \end{cases}$$

Trouvez une forme close pour $f(n)$ lorsque n est une puissance de 2 et donnez sa complexité.

Solution :

On trouve

n	$f(n)$		
$1 = 2^0$	$f(1)$	$=$	1
$2 = 2^1$	$f(2)$	$=$	$f(1)+1$
$4 = 2^2$	$f(4)$	$=$	$f(2)+1 = \big(f(1)+1\big)+1$
		$=$	$f(1)+2$
$8 = 2^3$	$f(8)$	$=$	$f(4)+1 = \big(f(1)+2\big)+1$
		$=$	$f(1)+3$
$16 = 2^4$	$f(16)$	$=$	$f(8)+1 = \big(f(1)+3\big)+1$
		$=$	$f(1)+4$
$n = 2^k$	$f(n)$	$=$	$f(1)+k$
		$=$	$f(1)+\log_2(n)$ car $n = 2^k \leftrightarrow k = \log_2(n)$
		$=$	$1+\log_2(n)$ car $f(1) = 1$

Ainsi, $f(n) = 1 + \log_2(n) \in O(\log(n))$.

Exercices

6.3 Pour cet exercice, n'utilisez la calculatrice que pour les opérations arithmétiques de base.

(a) Soit $f(n) = f(n/2) + 1$, $f(1) = 1$. Calculez $f(128)$.

(b) Soit $f(n) = 2f(n/3) + n^2$, $f(1) = 3$. Calculez $f(81)$.

(c) Soit $f(n) = 4f(n/2) + n$, $f(1) = 5$. Calculez $f(64)$.

(d) Soit $f(n) = 81f(n/9) + 3n^2$, $f(1) = 1$. Calculez $f(729)$.

6.4 Trouvez une forme close pour $f(n)$ et donnez sa complexité.

(a) $f(n) = 2f(n/2) + 3, \quad f(1) = 3,$

(b) $f(n) = 4f(n/2) + n, \quad f(1) = 1,$

(c) $f(n) = 3f(n/3) + 2n + 1, \quad f(1) = 1,$

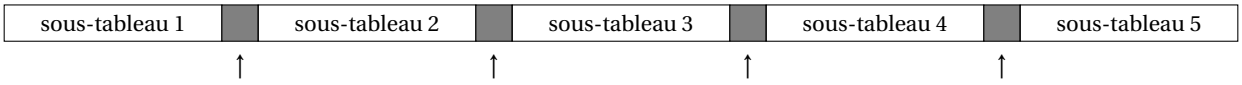
(d) $f(n) = 9f(n/3) + 3n^2, \quad f(1) = 2.$

6.5 Fouille d-chotomique

Afin de déterminer si une valeur x est présente dans un tableau trié de taille n, on peut utiliser la *fouille dichotomique*. Vulgairement, l'algorithme de la fouille dichotomique se résume à :

- Si $n = 1$ alors tester si l'unique case est x.
- Sinon, comparer x avec la case centrale du tableau. Si x est plus petit, alors on recommence avec la moitié gauche, sinon avec la moitié droite.

La fouille d-chotomique est une généralisation qui consiste à diviser le tableau en d sous tableaux (où le paramètre d est un nombre naturel). On compare alors x avec les cases situées aux extrémités de ces sous-tableaux, puis on effectue un appel récursif sur l'unique sous-tableau où x peut possiblement se trouver. Au pire cas, il faut effectuer $d - 1$ comparaisons. Par exemple, pour $d = 5$, la figure ci-dessous montre les 4 cases à tester afin de déterminer lequel des 5 sous-tableaux doit être considéré pour l'appel récursif.



La fonction $f(n)$ qui compte le nombre de comparaisons effectuées lors d'une fouille d-chotomique est

$$f(n) = f(n/d) + (d-1), \quad f(1) = 1.$$

Trouvez une forme close pour $f(n)$ lorsque n est une puissance de d et donnez sa complexité.

6.6 **Tri fusion** L'algorithme du *tri fusion* est un algorithme de fractionnement qui effectue un tri de la manière suivante. Soit n la taille du tableau, si

- $n < 2$ alors il n'y a rien à faire, le tableau est trié.
- $n \geq 2$ alors on coupe le tableau en deux moitiés, la gauche et la droite. On trie récursivement ces deux moitiés puis on les fusionne en un tableau trié.

```
1: fonction TriFusion(T : Tableau)
2:     n := taille de T
3:     si n ≥ 2 alors
4:         gauche := T[1..n/2]
5:         droite := T[n/2+1..n]
6:         TriFusion(gauche)
7:         TriFusion(droite)
8:         T := fusion(gauche, droite)
9:     fin si
10: fin fonction
```

Remarque: Cette fonction utilise la fonction `fusion` vue à l'exercice 5.12(e). Dans le cadre de cet exercice, il a été établi que pour deux tableaux de taille $n/2$, le nombre de comparaisons effectuées par `fusion` est $n-1$.

Soit $f(n)$ le nombre de comparaisons de cases du tableau effectuées par un appel à `TriFusion` *au pire cas*, où n est la taille du tableau T.

(a) Donnez les valeurs de $a, b \in \mathbb{N}$ et la fonction $g(n)$ permettant d'exprimer $f(n)$ sous la forme

$$f(n) = af(n/b) + g(n).$$

(b) Trouvez une forme close pour $f(n)$ lorsque n est une puissance de b et donnez sa complexité.

Chapitre 7

Preuve par récurrence

Étant donné une fonction propositionnelle $P(n)$, si on veut montrer que $P(n)$ est vraie pour $n \in \{0,1,2,\ldots\}$, alors on vérifie que la proposition $P(0)$ est vraie, que $P(1)$ est vraie et, finalement, que $P(2)$ est vraie. De même, si on veut montrer que la fonction $P(n)$ est vraie pour tout entier de 0 à 1 000 000, on peut encore procéder une valeur à la fois (préférablement avec un ordinateur). Par contre, si on veut montrer que $P(n)$ est vraie pour tout $n \in \mathbb{N}$, on a un problème : il y a une infinité de propositions à vérifier !

Comment vérifier une infinité de propositions en un nombre fini (et idéalement petit) d'étapes ? Le *principe de récurrence* est un raisonnement mathématique rigoureux qui permet justement de faire cela.

7.1 Preuve par récurrence simple

Le principe du raisonnement par récurrence peut être illustré simplement par l'analogie des dominos. Soit une suite infinie de dominos alignés de telle sorte que si on fait tomber le premier, tous les dominos tombent par réaction en chaîne.

Pour s'assurer que tous les dominos tombent, il suffit

1. de veiller à faire tomber le premier domino (**cas de base**) ;
2. de s'assurer que la distance entre chacun est telle que si un domino tombe, alors le suivant tombe lui aussi (**étape de récurrence**).

En utilisant une écriture plus mathématique, on peut définir la fonction propositionnelle $P(n)$: « le n^{e} domino tombe ». Pour démontrer que $P(n)$ est vraie pour tout $n \geq 1$, il suffit de montrer :

1. **Cas de base**
   Montrer que $P(1)$: « le 1^{er} domino tombe » est vraie.
2. **Étape de récurrence**
   Montrer que si le premier domino tombe, alors le second aussi, et que si le deuxième domino tombe, alors le troisième aussi, etc. Ceci se traduit par $P(1) \rightarrow P(2) \equiv$ **vrai**, et $P(2) \rightarrow P(3) \equiv$ **vrai**, ainsi de suite. De manière plus succinte, on écrit

$$\forall k \geq 1,\ P(k) \rightarrow P(k+1).$$

Ainsi, en démontrant le cas de base et l'étape de récurrence, on montre que toutes les propositions $P(1), P(2), P(3), P(4), \ldots$ sont vraies. En effet, par la règle d'inférence du *Modus ponens*, on obtient :

$$\frac{P(1) \qquad P(1) \rightarrow P(2)}{P(2)} \qquad \frac{P(2) \qquad P(2) \rightarrow P(3)}{P(3)} \qquad \frac{P(3) \qquad P(3) \rightarrow P(4)}{P(4)} \qquad \cdots$$

De manière plus générale, soit $P(n)$ une fonction propositionnelle dont le domaine est $\mathbb{N}$. La règle d'inférence suivante permet de montrer que $P(n)$ est vraie pour tout nombre naturel supérieur ou égal à n_0.

Théorème 7.1 : Principe du raisonnement par récurrence.

Soit $P(n)$ une fonction propositionnelle portant sur un nombre naturel n et soit $n_0 \in \mathbb{N}$.

$$P(n_0) \wedge [\forall k \geq n_0,\ P(k) \rightarrow P(k+1)] \quad \longrightarrow \quad \forall n \geq n_0,\ P(n)$$

Ainsi, pour démontrer $\forall n \geq n_0,\ P(n)$ en utilisant le principe du raisonnement par récurrence, on doit procéder en deux étapes, qui sont décrites dans l'encadré qui suit.

Comment démontrer $\forall n \geq n_0,\ P(n)$, par récurrence simple ?

On doit :

1. **Cas de base**
   Montrer que la fonction propositionnelle est vraie pour la première valeur (on commence habituellement à 0 ou 1, mais nous notons ici n_0 cette première valeur) :
   $$P(n_0) \equiv \textbf{vrai}.$$
2. **Étape de récurrence**
   Montrer que si la fonction propositionnelle est vraie pour un entier k, alors elle est aussi vraie pour l'entier suivant :
   $$\forall k \geq n_0,\ P(k) \rightarrow P(k+1).$$

Exemple 7.1

Démontrez la première partie du Théorème 5.7 sur les sommations : pour tout entier n strictement positif, la somme des entiers de 1 à n est égale à $\dfrac{n(n+1)}{2}$.

Solution :

On définit $P(n)$: « $\displaystyle\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ » et on veut montrer que $\forall n \in \mathbb{N}^*,\ P(n) \equiv \textbf{vrai}$.

1. **Cas de base** :
   Pour $n = 1$, on a $P(1) \equiv \textbf{vrai}$, car $\displaystyle\sum_{i=1}^{1} i = 1 = \frac{1(1+1)}{2}$.
2. **Étape de récurrence** :
   Soit $k \geq 1$, on doit montrer $P(k) \rightarrow P(k+1)$, ce qui revient à dire « si on fait l'hypothèse que la proposition $P(k)$ est vraie alors cela entraîne que $P(k+1)$ est aussi vraie ».

– **Hypothèse de récurrence**:

$$P(k) : \text{« } \sum_{i=1}^{k} i = \frac{k(k+1)}{2} \text{ » } \equiv \textbf{vrai}.$$

– **Objectif**:

$$P(k+1) : \text{« } \sum_{i=1}^{k+1} i = \frac{(k+1)(k+2)}{2} \text{ » } \equiv \textbf{vrai}.$$

– **Calculs**:

$$\begin{aligned}
\sum_{i=1}^{k+1} i &= 1+2+\cdots+k+(k+1) \\
&= \sum_{i=1}^{k} i + (k+1) \\
&= \frac{k(k+1)}{2} + (k+1) && \text{(par hyp. de réc.)} \\
&= \frac{k^2+k}{2} + \frac{2k+2}{2} \\
&= \frac{k^2+3k+2}{2} \\
&= \frac{(k+2)(k+1)}{2}
\end{aligned}$$

□

Exemple 7.2

Démontrez que pour tout entier $n \geq 1$, on a $n < 2^n$.

Solution :

On définit $P(n)$: « $n < 2^n$ » et on veut montrer que $\forall n \in \mathbb{N}$, $P(n) \equiv$ **vrai**.

1. **Cas de base**:
   Pour $n = 1$, on a $P(1)$: « $1 < 2^1$ » ≡ « $1 < 2$ » ≡ **vrai**.
2. **Étape de récurrence**:
   Soit $k \geq 1$, on doit montrer $P(k) \rightarrow P(k+1)$, ce qui revient à dire « si on fait l'hypothèse que la proposition $P(k)$ est vraie alors cela entraîne que $P(k+1)$ est aussi vraie ».
   – **Hypothèse de récurrence**:

$$P(k) : \text{« } k < 2^k \text{ » } \equiv \textbf{vrai}.$$

   – **Objectif**:

$$P(k+1) : \text{« } k+1 < 2^{k+1} \text{ » } \equiv \textbf{vrai}.$$

– **Calculs**:

$$\begin{array}{rrcll} & 1 & \leq & k & \text{(par définition de } k\text{)} \\ \rightarrow & k+1 & \leq & k+k & \\ \rightarrow & k+1 & \leq & 2\cdot k & \text{(car } k+k=2k\text{)} \\ \rightarrow & k+1 & < & 2\cdot 2^k & \text{(par hyp. de réc.)} \\ \rightarrow & k+1 & < & 2^{k+1} & \text{(comme } 2\cdot 2^k = 2^{k+1}\text{)} \end{array}$$

□

Exemple 7.3

Démontrez que pour tout entier n positif, 6 est un diviseur de (7^n-1).

Solution :

Remarque: il est difficile d'utiliser directement un énoncé de la forme « 6 est un diviseur de (7^n-1) » dans le cadre d'une preuve formelle. On utilise plutôt la définition plus mathématique suivante:

$$\exists c \in \mathbb{Z}, \quad (7^n-1) = 6c.$$

On définit ainsi $P(n)$: « $\exists c \in \mathbb{Z}, \quad (7^n-1) = 6c$ » et on veut montrer que $\forall n \in \mathbb{N}$, $P(n) \equiv$ **vrai**.

1. **Cas de base**:
   Pour $n=0$, on a $P(0)$: « $\exists c \in \mathbb{Z}, \quad (7^0-1) = 6c$ » ≡ « $0 = 6\cdot 0$ » ≡ **vrai**.
2. **Étape de récurrence**:
   Soit $k \geq 0$, on doit montrer $P(k) \rightarrow P(k+1)$, ce qui revient à dire « si on fait l'hypothèse que la proposition $P(k)$ est vraie alors cela entraîne que $P(k+1)$ est aussi vraie ».
   – **Hypothèse de récurrence**:
   $$P(k) : \text{« } \exists c \in \mathbb{Z}, \quad (7^k-1) = 6c \text{ »} \equiv \textbf{vrai}.$$
   – **Objectif**:
   $$P(k+1) : \text{« } \exists c' \in \mathbb{Z}, \quad (7^{k+1}-1) = 6c' \text{ »} \equiv \textbf{vrai}.$$
   – **Calculs**:
   $$\begin{aligned} 7^{k+1}-1 &= 7\cdot 7^k - 7 + 6 \\ &= 7\cdot(7^k-1)+6 \\ &= 7\cdot 6c + 6 && \text{(par hyp. de réc.)} \\ &= 6\cdot(7c+1) \\ &= 6c' && \text{(où } c' = 7c+1 \in \mathbb{Z}\text{)} \end{aligned}$$
   □

Exemple 7.4

À l'exemple 6.6 on a *trouvé* que la relation de récurrence

$$\begin{cases} a_0 = 2 \\ a_n = a_{n-1} + 3, \quad n \geq 1 \end{cases}$$

a comme solution $a_n = 2+3n$, sans démontrer le résultat. Prouvez que cette solution est juste.

Solution :
Posons $f(n) = 3n+2$. Définissons la fonction propositionnelle $P(n)$: « $a_n = f(n)$ » et montrons que $\forall n \in \mathbb{N}$, $P(n) \equiv$ **vrai**.

1. **Cas de base :**
   $P(0)$: « $a_0 = f(0)$ » $\equiv$ « $a_0 = 3\cdot 0+2$ » $\equiv$ « $a_0 = 2$ » $\equiv$ **vrai**.
2. **Étape de récurrence :**
   Soit $k \geq 0$, on doit montrer $P(k) \rightarrow P(k+1)$, ce qui revient à dire « si on fait l'hypothèse que la proposition $P(k)$ est vraie alors cela entraîne que $P(k+1)$ est aussi vraie ».
   - **Hypothèse de récurrence** :
     $$P(k) : \text{« } a_k = f(k) \text{ »} \equiv \textbf{vrai}.$$
   - **Objectif** :
     $$P(k+1) : \text{« } a_{k+1} = f(k+1) \text{ »} \equiv \textbf{vrai}.$$
   - **Calculs** :

$$\begin{aligned} f(k+1) &= 3(k+1)+2 && \text{(par définition de } f\text{)} \\ &= 3k+5 && \text{(par développement algébrique)} \end{aligned}$$

et

$$\begin{aligned} a_{k+1} &= a_k + 3 && \text{(par la relation de récurrence qui définit la suite } a\text{)} \\ &= 3k+2+3 && \text{(par l'hypothèse de récurrence)} \\ &= 3k+5. \end{aligned}$$

□

Exemple 7.5
Généralisation de la Loi de De Morgan. Soit $p_1, p_2, \ldots p_n$ des propositions. Démontrez que pour tout entier $n \geq 2$,

$$\neg(p_1 \wedge p_2 \wedge \cdots \wedge p_n) = \neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_n.$$

Solution :
On définit la fonction propositionnelle $P(n)$: « $\neg(p_1 \wedge p_2 \wedge \cdots \wedge p_n) = \neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_n$ » et on montre que $P(n) \equiv$ **vrai** pour tout $n \geq 2$.

1. **Cas de base :**
   $P(2) \equiv$ **vrai** car par la loi de De Morgan, $\neg(p_1 \wedge p_2) = \neg p_1 \vee \neg p_2$.
2. **Étape de récurrence :**
   Soit $k \geq 2$, on doit montrer $P(k) \rightarrow P(k+1)$, ce qui revient à dire « si on fait l'hypothèse que la proposition $P(k)$ est vraie alors cela entraîne que $P(k+1)$ est aussi vraie ».
   - **Hypothèse de récurrence** :
     $$P(k) : \text{« } \neg(p_1 \wedge p_2 \wedge \cdots \wedge p_k) = \neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_k \text{ »} \equiv \textbf{vrai}.$$
   - **Objectif** :
     $$P(k+1) : \text{« } \neg(p_1 \wedge p_2 \wedge \cdots \wedge p_{k+1}) = \neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_{k+1} \text{ »} \equiv \textbf{vrai}.$$

– **Calculs**:

$$\begin{aligned}\neg(p_1 \wedge p_2 \wedge \cdots \wedge p_{k+1}) &= \neg((p_1 \wedge p_2 \wedge \cdots \wedge p_k) \wedge p_{k+1}) \\ &= \neg(p_1 \wedge p_2 \wedge \cdots \wedge p_k) \vee \neg p_{k+1} && \text{(par } P(2)\text{, la loi de De Morgan)} \\ &= \neg p_1 \vee \neg p_2 \vee \cdots \neg p_k \vee \neg p_{k+1} && \text{(par hyp. de réc.)}\end{aligned}$$

□

Exercices

7.1 Utilisez une preuve par récurrence pour démontrer les énoncés suivants.

(a) Pour tout entier $n \geq 1$, $\sum_{i=1}^{n} i^2 = \dfrac{n(n+1)(2n+1)}{6}$.

(b) Pour tout entier $n \geq 0$, $\sum_{i=0}^{n} r^i = \dfrac{r^{n+1}-1}{r-1}$, si $r \in \mathbb{R} \setminus \{1\}$.

(c) Le nombre $n^3 - n$ est divisible par 3, pour tout entier $n \geq 0$.
Rappelons que $n^3 - n$ est divisible par 3 si et seulement si $n^3 - n = 3 \cdot a$, pour $a \in \mathbb{Z}$.

(d) Le nombre $n^2 + 3n$ est divisible par 2, pour tout entier $n \geq 0$.
Rappelons que $n^2 + 3n$ est divisible par 2 si et seulement si $n^2 + 3n = 2 \cdot a$, pour $a \in \mathbb{Z}$.

(e) $2n + 1 \leq 2^n$ pour tout entier $n \geq 3$.

(f) $n^2 \leq 2^n$ pour tout entier $n \geq 4$.

(g) $2^n < n!$ pour tout entier $n \geq 4$.

(h) $n! < n^n$ pour tout entier $n \geq 2$.

(i) Loi de De Morgan pour les ensembles. Soit $A_1, A_2, \ldots, A_n$ des ensembles. Pour tout entier $n \geq 2$,

$$\overline{A_1 \cup A_2 \cup \cdots \cup A_n} = \overline{A_1} \cap \overline{A_2} \cap \cdots \overline{A_n}.$$

(j) Pour tout entier $n \geq 0$, l'échiquier $2^n \times 2^n$ dont une case est occupée par la reine est pavable par des triominos en forme de L. Voir la figure 7.1 pour une illustration.

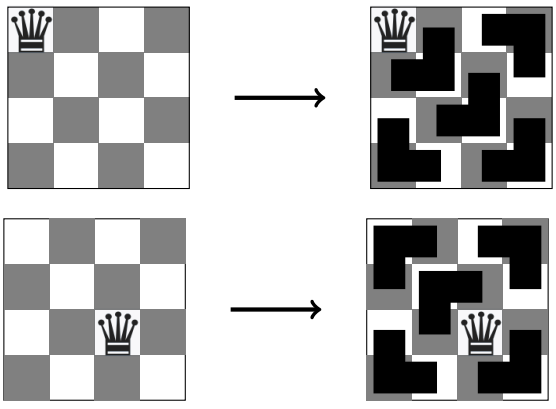


Figure 7.1 Illustration de l'exercice 7.1 (j) avec $n = 2$. Dans les deux cas, un échiquier $2^n \times 2^n$ dont une case est déjà occupée par une reine est pavé à l'aide de triominos en forme de L. Le but de l'exercice est de montrer qu'un tel pavage est possible pour tout entier $n \geq 0$ et peu importe la position de la reine.

7.2 On considère la somme des n premiers entiers impairs $\sum_{i=1}^{n}(2i-1) = 1+3+5+7+\cdots+n$.

(a) Trouvez une forme close pour cette somme.

(b) Montrez, en utilisant une preuve par récurrence, que la forme close trouvée en (a) est juste.

7.3 À l'exemple 6.7 on a *trouvé* que la relation de récurrence

$$\begin{cases} b_0 = 5 \\ b_n = b_{n-1} + 4n, \quad n \geq 1 \end{cases}$$

a comme solution $b_n = 2n^2 + 2n + 5$, sans démontrer le résultat de façon formelle. Montrez maintenant que cette solution est juste.

7.4 À l'exercice 6.2 on a *trouvé* que la relation de récurrence

$$\begin{cases} f(0) = 2 \\ f(n) = 3f(n-1) + 2, \quad n \geq 1 \end{cases}$$

a comme solution $f(n) = 3^{n+1} - 1$, sans démontrer le résultat de façon formelle. Montrez maintenant que cette solution est juste.

7.5 On considère la fonction propositionnelle : $P(n)$: « 3 divise $(n^3 + 2n + 1)$ ».

(a) Montrez que pour tout $k \geq 0$, $P(k) \rightarrow P(k+1)$.

(b) ★ Expliquez pourquoi, malgré le fait que l'*étape de récurrence* a été prouvée en (a), il est faux que $\forall n \geq 0$, $P(n)$.

7.6 Soit la fonction réelle $f(x) = x^n$, avec $n \geq 1$. Montrez, à l'aide d'une preuve par récurrence, que la dérivée de f est $f'(x) = nx^{n-1}$. *Indice : utilisez la dérivée d'un produit de fonction* $(uv)' = u'v + uv'$ *et* $x' = 1$.

7.7 Montrez, à l'aide d'une preuve par récurrence, que la dérivée $n^{\text{ième}}$ de $f(x) = \ln(x)$ est

$$f^{(n)}(x) = \frac{(-1)^{(n-1)}(n-1)!}{x^n}.$$

7.8 Considérez les nombre de Fibonacci définis par la relation de récurrence

$$\begin{cases} f_0 = 0, \quad f_1 = 1 \\ f_n = f_{n-1} + f_{n-2}, \quad n \geq 2. \end{cases}$$

Montrez par récurrence les deux énoncés suivants :

(a) $\forall n \in \mathbb{N}$, $\sum_{i=0}^{n} f_i = f_{n+2} - 1$;

(b) $\forall n \in \mathbb{N}$, $\sum_{i=0}^{n} f_i^2 = f_n \cdot f_{n+1}$.

7.2 Preuve par récurrence forte

Le principe de récurrence forte est une variante plus puissante du principe de récurrence dans la mesure où, à l'étape de récurrence, au lieu de montrer $P(k) \rightarrow P(k+1)$, on montre :

$$\big(P(0) \wedge P(1) \wedge P(2) \wedge \cdots \wedge P(k)\big) \rightarrow P(k+1).$$

Attention, il est important de souligner que même si l'énoncé peut sembler plus complexe, il est *plus facile* à démonter, car on effectue davantage d'hypothèses. En effet, pour montrer $P(k+1) \equiv$ **vrai**, au lieu de supposer seulement $P(k) \equiv$ **vrai**, on suppose : $P(0) \equiv$ **vrai** et $P(1) \equiv$ **vrai** et $P(2) \equiv$ **vrai** et $P(3) \equiv$ **vrai** et ainsi de suite jusqu'à $P(k) \equiv$ **vrai**.

Ainsi, en démontrant le cas de base $P(0)$ et l'étape de récurrence, on obtient $P(0), P(1), P(2), P(3), \ldots$, par les règles d'inférence de la *conjonction* et du *modus ponens* :

$$\frac{P(0) \qquad P(0) \rightarrow P(1)}{P(1)} \qquad \frac{P(0) \qquad P(1) \qquad P(0) \wedge P(1) \rightarrow P(2)}{P(2)} \qquad \frac{P(0) \qquad P(1) \qquad P(2) \qquad P(0) \wedge P(1) \wedge P(2) \rightarrow P(3)}{P(3)} \qquad \cdots$$

De manière plus générale, soit $P(n)$ une fonction propositionnelle dont le domaine est $\mathbb{N}$. La règle d'inférence suivante permet de montrer que $P(n)$ est vraie pour tout nombre naturel supérieur ou égal à n_0. Cette règle permet aussi de gérer la situation où il y a plusieurs cas de base

$$n_0, n_0+1, \ldots, n_0+j,$$

pour $j \in \mathbb{N}$.

Théorème 7.2 : Principe de raisonnement par récurrence forte.

Soit $P(n)$ une proposition portant sur un nombre naturel n et soit $n_0, j \in \mathbb{N}$.

$$\big[P(n_0) \wedge P(n_0+1) \wedge \ldots \wedge P(n_0+j)\big] \wedge \big[\forall k \geq (n_0+j),\, P(n_0) \wedge P(n_0+1) \wedge \cdots \wedge P(k) \rightarrow P(k+1)\big] \longrightarrow \forall n \geq n_0,\, P(n)$$

Ainsi, pour démontrer $\forall n \geq n_0,\ P(n)$ en utilisant le principe du raisonnement par récurrence forte, on doit procéder en deux étapes, qui sont décrites dans l'encadré qui suit.

Comment démontrer $\forall n \geq n_0,\ P(n)$ par récurrence forte ?

On doit :

1. **Cas de base**
   Montrer que $P(n_0), P(n_0+1), \ldots, P(n_0+j)$ sont vraies ($j \in \mathbb{N}$).
   (n_0 est le premier cas de base, n_0+j est le dernier cas de base, il y a donc $j+1$ cas de base)
2. **Étape de récurrence**
   Montrer que
   $$\forall k \geq (n_0+j),\ \big(P(n_0) \wedge P(n_0+1) \wedge \cdots \wedge P(k)\big) \rightarrow P(k+1).$$

Exemple 7.6

Montrez que tout entier $n \geq 2$ peut s'écrire comme produit de nombres premiers.

Solution :

On définit $P(n)$: « il existe des nombres premiers $p_1, p_2, \ldots,$ et p_r tels que $n = p_1 p_2 \cdots p_r$ » et on montre que $\forall n \geq 2, P(n) \equiv$ **vrai**.

1. **Cas de base** : (ici $n_0 = 2$ et $j = 0$)
   $P(2) \equiv$ **vrai** car 2 est premier. En effet, $P(2)$ est satisfait avec $r = 1$ et $p_1 = 2$.
2. **Étape de récurrence** :
   Soit $k \geq 2$, on doit montrer que $P(2) \wedge P(3) \wedge \cdots \wedge P(k) \rightarrow P(k+1)$.
   - **Hypothèse de récurrence** :
     $$P(2) \wedge P(3) \wedge \cdots \wedge P(k) \equiv \textbf{vrai}.$$
     On suppose donc que tout nombre entre 2 et k peut s'écrire comme produit de nombre premiers.
   - **Objectif** :
     $$P(k+1) \equiv \textbf{vrai}.$$
     Il faut donc montrer que l'entier $k+1$ peut s'écrire comme produit de nombres premiers.
   - **Raisonnement** :
     On considère deux cas :
     - Si $k+1$ est un nombre premier alors $P(k+1)$ est satisfait avec $r = 1$ et $p_1 = k+1$.
     - Si $k+1$ n'est pas premier alors il est composé. Cela signifie qu'il existe deux entiers a, b, $2 \leq a \leq k$, $2 \leq b \leq k$ tels que $k+1 = ab$. Par hypothèse de récurrence, $P(a)$ est vrai et donc il existe des nombres premiers $p_1, p_2, \ldots, p_r$ tels que $p_1 p_2 \cdots p_r = a$. De plus, par l'hypothèse de récurrence on a aussi que $P(b)$ est vrai et donc il existe des nombres premiers $p'_1, p'_2, \ldots, p'_{r'}$ tels que $b = p'_1 p'_2 \cdots p'_{r'}$.

       Pour conclure, on a $k+1 = ab = p_1 p_2 \cdots p_r p'_1 p'_2 \cdots p'_{r'}$ où tous les nombres du produit de droite sont premiers. □

Exemple 7.7

Soit la relation de récurrence

$$\begin{cases} a_1 = 2 \\ a_2 = 9 \\ a_n = 2a_{n-1} + 3a_{n-2}, \quad n \geq 3 \end{cases}$$

Montrez que $a_n \leq 3^n$, pour tout entier n strictement positif.

Solution :
On définit $P(n)$: « $a_n \leq 3^n$ » et on montre que $\forall n \geq 1, P(n) \equiv$ **vrai**.

1. **Cas de base**: (ici $n_0 = 1$ et $j = 1$)
   $P(1) : a_1 = 2 \leq 3 = 3^1$ est vrai.
   $P(2) : a_2 = 9 \leq 9 = 3^2$ est vrai.
2. **Étape de récurrence:**
   Soit $k \geq 2$, on doit montrer que $P(1) \wedge P(2) \wedge \cdots \wedge P(k) \rightarrow P(k+1)$.
   – **Hypothèse de récurrence**:
   $$P(1) \wedge P(2) \wedge \cdots \wedge P(k) \equiv \textbf{vrai}.$$
   En particulier, $P(k)$: « $a_k \leq 3^k$ » $\equiv$ **vrai** et $P(k-1)$: « $a_{k-1} \leq 3^{k-1}$ » $\equiv$ **vrai**.
   – **Objectif**:
   $$P(k+1) : \text{« } a_{k+1} \leq 3^{k+1} \text{ »} \equiv \textbf{vrai}.$$
   – **Raisonnement**:

$$\begin{aligned} a_{k+1} &= 2a_k + 3a_{k-1} && \text{(par la définition de la relation de récurrence)} \\ &\leq 2 \cdot 3^k + 3 \cdot 3^{k-1} && \text{(par hypothèse de récurrence)} \\ &= 2 \cdot 3^k + 3^k \\ &= 3^k(2+1) \\ &= 3^{k+1}. \end{aligned}$$

□

Exemple 7.8
Soit $n \geq 4$. Montrez que toute facture de n \$ peut être payée uniquement par une combinaison de 2 \$ et de 5 \$.

Solution :
On voit qu'on peut obtenir des montants de $4\$ = 2 \cdot 2\$$ et 5 \$ avec des 2 \$ et 5 \$. En ajoutant un 2 \$ à chacun de ces montants, on obtient des totaux de 6 \$ et 7 \$. En ajoutant à nouveau un 2 \$ à ces totaux, on arrive à payer des factures de 8 \$ et 9 \$. En continuant ainsi, on pourra payer n'importe quelle facture supérieure ou égale à 4 \$, à l'aide de 2 \$ et 5 \$. Formalisons ce raisonnement de façon algébrique.

On définit $P(n)$: « Il existe $a, b \in \mathbb{N}$ tels que $2a + 5b = n$ » et on veut montrer que $\forall n \geq 4$, $P(n) \equiv$ **vrai**.

1. **Cas de base:** (ici $n_0 = 4$ et $j = 1$)
   $P(4) \equiv$ **vrai** avec $a = 2$ et, $b = 0$ car $4 = 2 \cdot 2 + 0 \cdot 5$,
   $P(5) \equiv$ **vrai** avec $a = 0$ et, $b = 1$ car $5 = 0 \cdot 2 + 1 \cdot 5$,

2. **Étape de récurrence:**
   Soit $k \geq 5$, on doit montrer que $P(4) \wedge P(5) \wedge \cdots \wedge P(k) \rightarrow P(k+1)$.
   – **Hypothèse de récurrence**:
   $$P(4) \wedge P(5) \wedge \cdots \wedge P(k-1) \wedge P(k) \equiv \textbf{vrai}.$$
   En particulier, on a que $P(k-1) \equiv$ **vrai**. On a donc qu'il existe $a, b \in \mathbb{N}$ tels que $2a + 5b = k - 1$.

– **Objectif**:

$$P(k+1): \text{« Il existe } a_2, b_2 \in \mathbb{N} \text{ tels que } 2a_2+5b_2 = k+1 \text{ »} \equiv \textbf{vrai}.$$

– **Calculs**:

$$\begin{aligned} k+1 &= (k-1)+2 \\ &= 2a+5b+5 && \text{(par hyp. de réc.)} \\ &= 2a+5(b+1) \\ &= 2a_2+5b_2. && \text{(avec } a_2 = a \in \mathbb{N} \text{ et } b_2 = b+1 \in \mathbb{N}) \end{aligned}$$

□

Autre solution:

On voit qu'on peut obtenir des montants de 4 $ à 8 $ avec des 2 $ et 5 $. En ajoutant un 5 $ à chacun de ces montants, on obtient des totaux de 9 $ à 13 $. En ajoutant à nouveau un 5 $ à ces totaux, on arrive à payer des factures de 14 $ et 18 $. En continuant ainsi, on pourra payer n'importe quelle facture supérieure ou égale à 4 $, à l'aide de 2 $ et 5 $. Formalisons ce raisonnement de façon algébrique.

On définit $P(n)$: « il existe $a, b \in \mathbb{N}$ tels que $2a+5b = n$ » et on veut montrer que pour tout $n \geq 4$, $P(n) \equiv$ **vrai**.

1. **Cas de base:** (ici $n_0 = 4$ et $j = 3$)
   $P(4) \equiv$ **vrai** avec $a = 2$ et, $b = 0$ car $4 = 2\cdot 2 + 0\cdot 5$,
   $P(5) \equiv$ **vrai** avec $a = 0$ et, $b = 2$ car $5 = 0\cdot 2 + 2\cdot 5$,
   $P(6) \equiv$ **vrai** avec $a = 3$ et, $b = 0$ car $6 = 3\cdot 2 + 0\cdot 5$,
   $P(7) \equiv$ **vrai** avec $a = 1$ et, $b = 1$ car $7 = 1\cdot 2 + 1\cdot 5$,
   $P(8) \equiv$ **vrai** avec $a = 4$ et, $b = 0$ car $8 = 4\cdot 2 + 0\cdot 5$,

2. **Étape de récurrence:** Soit $k \geq 8$, on doit montrer que $P(4) \wedge P(5) \wedge \cdots \wedge P(k) \rightarrow P(k+1)$.
   – **Hypothèse de récurrence**:

   $$P(4) \wedge P(6) \wedge \cdots \wedge P(k-1) \wedge P(k) \equiv \textbf{vrai}.$$

   En particulier, on a que $P(k-4) \equiv$ **vrai**. On a donc qu'il existe $a, b \in \mathbb{N}$ tels que $2a+5b = k-4$.
   – **Objectif**:

   $$P(k+1): \text{« Il existe } a_2, b_2 \in \mathbb{N} \text{ tels que } 2a_2+5b_2 = k+1 \text{ »} \equiv \textbf{vrai}.$$

   – Calculs:

   $$\begin{aligned} k+1 &= (k-4)+5 \\ &= 2a+5b+5 && \text{(par hyp. de réc.)} \\ &= 2a+5(b+1) \\ &= 2a_2+5b_2. && \text{(avec } a_2 = a \in \mathbb{N} \text{ et } b_2 = b+1 \in \mathbb{N}) \end{aligned}$$

   □

Exercices

7.9 On considère la relation de récurrence $a_0 = 0$, $a_1 = 2$ et $a_n = 4a_{n-1} - 4a_{n-2}$ pour $n \geq 2$. Utilisez le principe de récurrence forte pour montrer que $a_n = n2^n$ pour tout $n \geq 0$.

7.10 On considère la relation de récurrence $a_0 = 0$, $a_1 = 0$, $a_2 = 2$ et $a_n = 6a_{n-1} - 11a_{n-2} + 6a_{n-3}$ pour $n \geq 3$. Utilisez le principe de récurrence forte pour montrer que $a_n = 3^n - 2^{n+1} + 1$ pour tout $n \geq 0$.

7.11 Soit $n \geq 12$. Montrez que tout timbre de n ¢ peut être remplacé uniquement par une combinaison de timbres de 3 ¢ et de 7 ¢.

7.12 Soit $n \geq 24$. Montrez que tout timbre de n ¢ peut être remplacé uniquement par une combinaison de 5 ¢ et de 7 ¢.

7.3 Preuve de validité d'un algorithme récursif

Les preuves par récurrence sont aussi utilisées pour montrer qu'un algorithme est **correct**, c'est-à-dire qu'il produit toujours la bonne solution.

Exemple 7.9

Montrez que l'algorithme **`puissance`**(a, n) de l'Exemple 6.1 est correct, pour tout $a \in \mathbb{R}^*$ et $n \in \mathbb{N}$.

Solution :

$$p(a,n) := \begin{cases} \text{undef}, & a=0 \text{ and } n=0 \text{ or } n<0 \text{ or } \operatorname{mod}(n,1) \neq 1 \\ 0, & a=0 \text{ and } n \neq 0 \\ 1, & a \neq 0 \text{ and } n=0 \\ a \cdot p(a,n-1), & \text{Else} \end{cases}$$

Soit $P(n)$: « $p(a,n) = a^n$ ». Montrons que $\forall n \geq 0, P(n) \equiv$ **vrai**, par récurrence simple.

1. **Cas de base**:

$$\begin{aligned} p(a,0) &= 1 && \text{(par l'algorithme)} \\ &= a^0. && \text{(propriété des exposants)} \end{aligned}$$

   Ainsi, $P(0)$: « $p(a,0) = a^0$ » $\equiv$ **vrai**.

2. **Étape de récurrence**:
   Soit $k \geq 0$, on doit montrer que $P(k) \rightarrow P(k+1)$.
   - **Hypothèse de récurrence**:
   $$P(k) : \text{« } p(a,k) = a^k \text{ »} \equiv \textbf{vrai}.$$
   - **Objectif**:
   $$P(k+1) : \text{« } p(a,k+1) = a^{k+1} \text{ »} \equiv \textbf{vrai}.$$

– **Calculs**:

$$\begin{aligned} p(a,k+1) &= a \cdot p(a,k) && \text{(par l'algorithme)} \\ &= a \cdot a^k && \text{(par l'hypothèse de récurrence)} \\ &= a^{k+1}. && \text{(propriété des exposants)} \end{aligned}$$

□

Exemple 7.10

Problème des tours de Hanoï.

But: Déplacer une tour de n disques ayant des diamètres différents de la première tige à la troisième tige.

Règles:

- déplacer un seul disque à la fois;
- déplacer un disque seulement s'il est au sommet d'une pile;
- déplacer un disque seulement sur un disque de diamètre plus grand, ou sur une tige vide.

(a) Donnez un algorithme récursif pour résoudre ce problème.

(b) Montrez que l'algorithme donné en (a) est correct.

(c) Donnez sa complexité. *Comptez le nombre de déplacements de disques.*

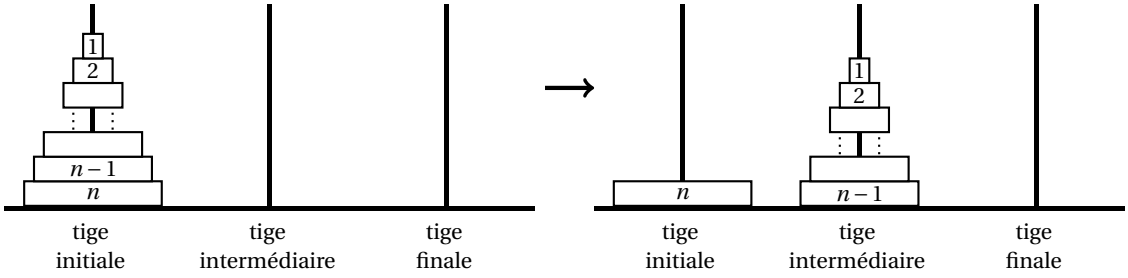
Solution :

(a) On veut déplacer tous les disques de la première tige vers la troisième. Pour simplifier, on appellera la première tige la *tige initiale*, la deuxième tige la *tige intermédiaire* et la troisième tige la *tige finale*. Un algorithme récursif peut être déduit des considérations suivantes:

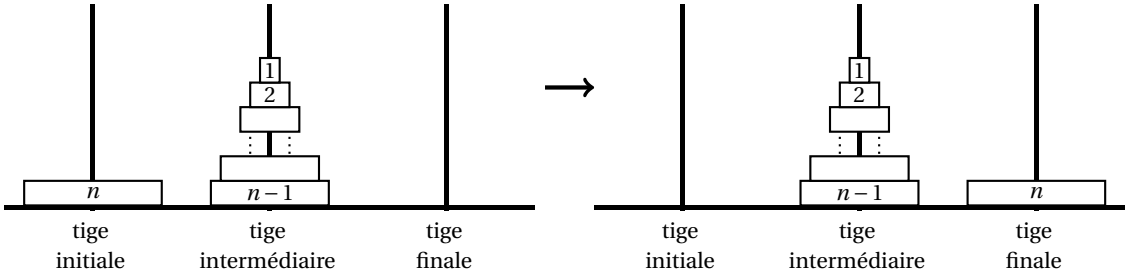
- À un moment, il faudra déplacer le plus grand disque de la tige initiale vers la tige finale.
- Pour retirer le plus grand disque de la tige initiale, il faut qu'il n'y ait aucun autre disque sur cette tige.
- Pour déposer le plus grand disque sur la tige finale, il faut qu'il n'y ait aucun disque sur cette tige.
- Ainsi, pour déplacer le plus grand disque de la tige initiale à la tige finale, il faut forcément que tous les autres disques soient empilés sur la tige intermédiaire.

L'algorithme général pour n disques est donc:

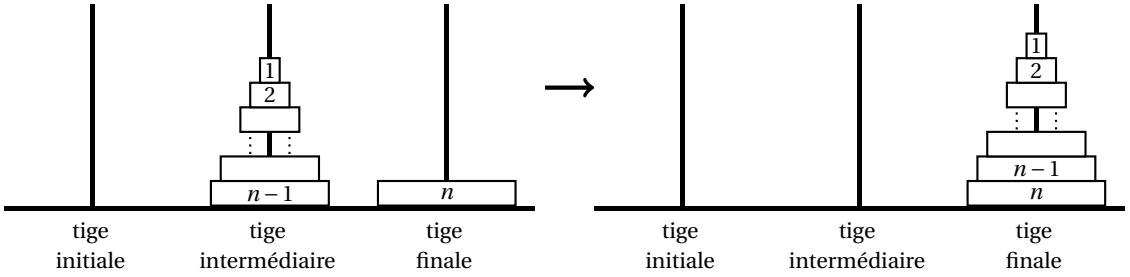
1. Déplacer $n-1$ disques de la tige initiale vers la tige intermédiaire (possiblement en plusieurs étapes).



2. Déplacer le n-ième disque de la tige initiale à la tige finale (en une étape).



3. Déplacer $n-1$ disques de la tige intermédiaire à la tige finale (possiblement en plusieurs étapes).




```
1: fonction Hanoï(n, a, b, c : Entiers)
2:                 ▷ n : nombre de disques, a tige initiale, b tige finale et c tige intermédiaire.
3:     si n > 0 alors
4:         Hanoï(n − 1, a, c, b)
5:         Déplacer le disque n de la tige a à la tige b
6:         Hanoï(n − 1, c, b, a)
7:     fin si
8: fin fonction
```

(b) On définit $P(n)$: « un appel à `Hanoï`(n, a, b, c) déplace n disques de la tige a à la tige b » et on veut montrer que pour tout $n \geq 0$, $P(n) \equiv$ **vrai**.

1.**Cas de base** :
On montre que l'algorithme est correct pour $n = 0$. Avec $n = 0$, aucun déplacement de disque n'est effectué.

2.**Étape de récurrence** :
Soit $k \geq 0$.
−**Hypothèse de récurrence** :

$P(k)$: « un appel à `Hanoï`(k, a, b, c) déplace k disques de la tige a à la tige b » ≡ **vrai**.

−**Objectif** :

$$P(k+1) \equiv \textbf{vrai}.$$

On veut montrer que l'algorithme est correct pour déplacer $k+1$ disques.

– **Argumentation** :
Tout d'abord, on remarque que $k+1>0$ et donc la condition du **si** est vraie. Ensuite, par hypothèse de récurrence, l'appel récursif de la ligne 4 déplace les k premiers disques de la tige initiale vers la tige intermédiaire. Ensuite, la ligne 5 déplace le disque $k+1$ de la tige initiale vers la tige finale. Finalement, encore par l'hypothèse de récurrence, on a que l'appel de la ligne 6 déplace les $k-1$ premiers disques de la tige intermédiaire vers à la tige finale. Tous les disques ont donc été correctement déplacés en respectant les règles des tours de Hanoï. □

(c) Soit $f(n)$ le nombre de déplacements de disques effectués par un appel à `Hanoï`(n, a, b, c). On observe que :

$$f(n) = \begin{cases} 0 & \text{si } n = 0, \\ 2f(n-1)+1 & \text{si } n \geq 1. \end{cases}$$

En utilisant la méthode itérative :

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 2f(0)+1 = 1 \\ f(2) &= 2f(1)+1 = 2+1 \\ f(3) &= 2f(2)+1 = 2(2+1)+1 = 2^2+2+1 \\ f(4) &= 2f(3)+1 = 2(2^2+2+1)+1 = 2^3+2^2+2+1 \\ &\vdots \\ f(n) &= \sum_{i=0}^{n-1} 2^i = 2^n 1 \end{aligned}$$

Donc, $f(n) = 2^n - 1 \in O(2^n)$.

Exercices

7.13 Factorielle.

(a) Écrivez un algorithme récursif pour calculer $n!$, où $n \in \mathbb{N}$.

(b) Montrez que cet algorithme est correct.

(c) Donnez sa complexité. *Compter le nombre de multiplications.*

7.14

Considérez l'algorithme récursif suivant, où $n \in \mathbb{N}$.

```
1: fonction mystère(n)
2:     si n = 0 alors
3:         retourner 0
4:     sinon
5:         retourner mystère(n − 1) + 3 · n · n − 3 · n + 1
6:     fin si
7: fin fonction
```

(a) Devinez ce que la fonction `mystère`(n) retourne.

(b) Validez le résultat obtenu en (a) en utilisant une preuve par récurrence.

(c) Soit $f(n)$ le nombre de multiplications requises lors d'un appel à `mystère`(n). Trouvez une relation de récurrence pour $f(n)$.

(d) Quelle est la complexité de cet algorithme? *Déduire une forme close pour* $f(n)$.

7.15 Fibonacci.

(a) Écrire un algorithme récursif pour calculer f_n, le n-ième nombre de Fibonacci.

(b) Montrez que cet algorithme est correct.

(c) Soit $g(n)$ le nombre d'additions effectuées lors d'un appel à l'algorithme écrit en (a). Montrer que $g(n) = f_{(n+1)} - 1$.

(d) Écrivez un algorithme itératif pour calculer f_n, le n-ième nombre de Fibonacci. Vous ne devriez utiliser que trois variables entières et un indice de boucle.

(e) Déterminez $h(n)$, le nombre d'additions effectuées lors d'un appel à l'algorithme écrit en (d).

(f) Sachant que $g(n) \in \Theta(\varphi^n)$, où $\varphi = \frac{1+\sqrt{5}}{2}$ est le nombre d'or, déterminez parmi les deux algorithmes écrits en (a) et (d), lequel est le plus efficace?

(g) ★ Écrivez un algorithme qui calcule le n-ième nombres de Fibonacci avec un nombre d'opérations arithmétiques dans $O(\log(n))$.

Chapitre 8

Dénombrement

8.1 Notions de base

8.1.1 Principe du produit

Théorème 8.1 : Principe du produit

Si l'événement A_1 peut se produire de n_1 façons différentes et l'événement A_2 peut se produire de n_2 façons différentes, alors l'événement A_1 suivi de A_2 peut de produire de $n_1 n_2$ façons différentes.

Exemple 8.1

Quel est le nombre d'additions effectuées lors de l'exécution du segment d'algorithme suivant?

1: **pour** $i = 1$ **à** n_1 **faire**
2: **pour** $j = 1$ **à** n_2 **faire**
3: $k := k + 1$
4: **fin pour**
5: **fin pour**

Solution :

$$\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} 1 = n_1 n_2$$

Exemple 8.2

Si A_1 et A_2 sont des ensembles finis, alors

$$|A_1 \times A_2| = |A_1||A_2|.$$

8.1.2 Principe de la somme

Théorème 8.2 : Principe de la somme

Si l'événement A_1 peut se produire de n_1 façons différentes et l'événement A_2 peut se produire de n_2 façons différentes et que A_1 et A_2 ne peuvent se produire simultanément, alors l'événement A_1 ou A_2 peut se produire de $n_1 + n_2$ façons différentes.

Exemple 8.3

Quel est le nombre d'additions effectuées lors de l'exécution du segment d'algorithme suivant?

1: **pour** $i = 1$ **à** n_1 **faire**
2: $k := k + 1$
3: **fin pour**
4: **pour** $j = 1$ **à** n_2 **faire**
5: $k := k + 1$
6: **fin pour**

Solution :

$$\sum_{i=1}^{n_1} 1 + \sum_{j=1}^{n_2} 1 = n_1 + n_2$$

Exemple 8.4

Si A_1 et A_2 sont des ensembles finis et **disjoints**, alors

$$|A_1 \cup A_2| = |A_1| + |A_2|.$$

8.1.3 Principe d'inclusion-exclusion

Théorème 8.3 : Principe d'inclusion-exclusion

Si A_1 et A_2 sont des ensembles finis, alors

$$|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2|$$

8.1.4 Principe des tiroirs

« Si n chaussettes occupent m tiroirs et $n > m$, alors au moins un tiroir doit contenir au moins deux chaussettes. »

Théorème 8.4 : Principe des tiroirs

Si $k+1$ objets ou plus sont rangés dans k boîtes alors il y a au moins une boîte qui contient 2 objets ou plus.

Exemple 8.5

Dans un groupe de 8 personnes, il y a au moins deux personnes qui sont nées le même jour de la semaine.

Théorème 8.5 : Principe des tiroirs généralisé

Si n objets sont placés dans k boîtes, alors il y a au moins une boîte qui contient au moins $\lceil \frac{n}{k} \rceil$ objets.

Exemple 8.6

Dans un groupe de 20 personnes, il y a au moins $\lceil \frac{20}{7} \rceil = 3$ personnes qui sont nées le même jour de la semaine.

8.2 Permutations et arrangements

Définition 8.1 : Permutation

Une **permutation** d'un ensemble d'objets distincts est un arrangement ordonné de ces objets.

Exemple 8.7

Si $S = \{1,2,3\}$, alors

$$123,\ 132,\ 213,\ 231,\ 312,\ 321$$

sont les permutations des éléments de S.

Théorème 8.6 : Principe fondamental du dénombrement

Il y a $n!$ permutations de n objets distincts.

Définition 8.2 : Arrangement

Un **arrangement** de k éléments (ou une k-permutation) est une **suite ordonnée** de k éléments d'un ensemble.

Exemple 8.8

Donnez tous les arrangements de 2 éléments de l'ensemble $S = \{1,2,3\}$.

Solution :

$$12,\ 13,\ 21,\ 23,\ 31,\ 32$$

Remarque. Nous avons utilisé ici une notation allégée, mais on pourrait séparer les éléments par des virgules et présenter l'ensemble des arrangements ainsi :

$$\{(1,2),(1,3),(2,1),(2,3),(3,1),(3,2)\}.$$

Théorème 8.7

Le nombre d'arrangements de k éléments d'un ensemble de n éléments distincts est

$$P(n,k) = n(n-1)\cdots(n-k+1) = \frac{n!}{(n-k)!}, \text{où } 1 \leq k \leq n.$$

Ainsi, lorsque l'**ordre de sélection a de l'importance**, il y a $P(n,k)$ façons de choisir k objets parmi n objets distincts.

Exemple 8.9

Si $S = \{a,b,c\}$, déterminez le nombre d'arrangements de 2 éléments de S et énumérez ces arrangements.

Solution :

Le nombre d'arrangements de 2 éléments de S est

$$P(3,2) = \frac{3!}{(3-2)!} = 6.$$

Les 6 arrangements sont les suivants (notez que l'ordre a de l'importance dans un arrangement, et qu'il ne peut pas y avoir de répétition du même élément) : ab, ac, ba, bc, ca, cb.

Exemple 8.10

Si $S = \{a, b, c, d\}$, déterminez le nombre d'arrangements de 2 éléments de S et énumérez ces arrangements.

Solution :

Le nombre d'arrangements de 2 éléments de S est

$$P(4,2) = \frac{4!}{(4-2)!} = 12.$$

Les 12 arrangements sont les suivants (notez que l'ordre a de l'importance dans un arrangement, et qu'il ne peut pas y avoir de répétition du même élément) : $ab, ac, ad, ba, bc, bd, ca, cb, cd, da, db, dc$.

8.3 Combinaisons

Définition 8.3 : Combinaison

Une **combinaison** de k éléments (ou une k-combinaison) est un sous-ensemble de cardinalité k d'un ensemble.

Exemple 8.11

Si $S = \{1,2,3\}$, énumérez les combinaisons de 2 éléments de S.

Solution :

$$\{1,2\},\ \{1,3\},\ \{2,3\}$$

Notez que l'ordre n'a pas d'importance dans une combinaison, et qu'un élément ne peut être répété.

Théorème 8.8

Le nombre de combinaisons de k éléments dans un ensemble de n éléments distincts est

$$C(n,k) = \frac{n!}{k!(n-k)!} = \binom{n}{k}$$

où $0 \leq k \leq n$. Lorsque l'**ordre de sélection n'a pas d'importance**, il y a $C(n,k)$ façons de choisir k objets parmi n objets distincts.

Note:

1. $C(n,k) = \dfrac{P(n,k)}{k!}$.
2. $C(n,k)$ est le nombre de sous-ensembles de cardinalité k d'un ensemble de n éléments.
3. $C(n,k) = C(n,n-k)$.

Exemple 8.12

Utilisez le théorème précédent pour calculer le nombre de combinaisons de 2 éléments de l'ensemble $S = \{1,2,3\}$.

Solution :

$$C(3,2) = \binom{3}{2} = \frac{3!}{2!(3-2)!} = 3.$$

Exemple 8.13
Si $S = \{a, b, c, d\}$, déterminez le nombre de combinaisons de 2 éléments de S et énumérez ces combinaisons.

Solution :

$$C(4,2) = \binom{4}{2} = \frac{4!}{2!(4-2)!} = 6.$$

Les 6 combinaisons de 2 éléments de S sont :

$$\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}.$$

Notez que l'ordre n'a pas d'importance dans une combinaison, et qu'un élément ne peut être répété.

Théorème 8.9 : Permutations avec objets indistinguables

Le nombre de permutations différentes de n objets, où il y a n_1 objets de type 1, n_2 objets de type 2, …, n_k objets de type k, est

$$\frac{n!}{n_1!n_2!\cdots n_k!}.$$

Exemple 8.14
Combien peut-on former de mots distincts de 11 lettres en utilisant chacune des lettres du mot MISSISSIPPI ?

Solution :

$$\frac{11!}{4!\,1!\,2!\,4!} = 34650$$

Exercices

8.1

(a) Combien y a-t-il de trains de bits de longueur 8 ?

(b) Combien y a-t-il de train de bits de longueur 8 qui commencent par 01 et terminent par 101 ?

(c) Combien de plaques d'immatriculation peut-on former si on utilise 3 chiffres suivis de 3 lettres ?

8.2 Combien y a-t-il de mots de passe comportant 6 à 8 caractères, avec au moins un chiffre et où chaque caractère est une lettre majuscule ou un chiffre ?

8.3

(a) Sur un total de 300 étudiants en génie logiciel, 143 sont inscrits au cours MAT210, 122 au cours MAT265, et 21 sont inscrits aux deux. Combien y a-t-il d'étudiants en génie logiciel qui ne sont ni inscrits au cours MAT210 ni inscrits au cours MAT265 ?

(b) Combien y a-t-il de train de bits de longueur 8 qui commencent par 01 ou se terminent par 101 ?

8.4

(a) Cinq femmes et deux hommes attendent à la pharmacie. Combien de façons y a-t-il de les placer dans la file d'attente ?

(b) De combien de façons 5 femmes et 2 hommes peuvent-ils être disposés dans une file d'attente si les 2 hommes doivent se suivre ?

(c) De combien de façons 5 femmes et 2 hommes ils être disposés dans une file d'attente si les 2 hommes doivent être séparés ?

(d) Combien de permutations des lettres A, B, C, D, E, F et G contiennent la chaîne "ABC" ?

8.5 Utilisez le principe des tiroirs ou une autre astuce afin de résoudre les problèmes suivants.

(a) Combien d'étudiants doit-il y avoir dans une classe pour qu'au moins 6 aient la même note (A, B, C, D ou E) ?

(b) Combien de mots (sur les lettres A à Z) de 6 lettres doit-on engendrer pour qu'au moins quatre mots aient les mêmes 3 premières lettres ?

(c) Déterminez quel est le plus petit nombre d'indicatifs régionaux nécessaires pour couvrir 17 millions d'abonnés téléphoniques si les numéros de téléphone ont 10 chiffres, incluant le code régional de 3 chiffres : AXX-AXX-XXXX, où $2 \leq A \leq 9$ et $0 \leq X \leq 9$.

8.6

(a) Un programme est conçu pour générer tous les mots de 2 lettres différentes formés à partir des lettres A, B, C, D et E. Combien de mots différents devrait-il générer ?

(b) De combien de façons 5 femmes et 3 hommes peuvent-ils être disposés dans une file d'attente si les 3 hommes doivent être séparés ?

8.7

(a) Combien de trains de bits de longueur 8 contiennent exactement trois 1 ?

(b) Combien de trains de bits de longueur 8 contiennent au plus trois 1 ?

(c) De combien de façons peut-on choisir les 6 numéros du tirage de la 6/49 ?

(d) On veut former un comité de 6 étudiants dont 4 sont en génie logiciel et 2 en génie électrique. Sachant qu'il y a 657 étudiants inscrits en génie logiciel et 753 en génie électrique, de combien de façons peut-on former le comité ?

8.8 On compte 40 professeurs et 300 étudiants dans un département.

(a) Combien y a-t-il de façons différentes de former un comité de 2 professeurs et 5 étudiants de ce département ?

(b) Combien y a-t-il de façons différentes de former un comité de 5 personnes de ce département : soit 2 professeurs et 3 étudiants ou 3 professeurs et 2 étudiants ?

(c) Combien y a-t-il de façons différentes de former un comité de 5 personnes de ce département, sans contrainte sur le fait de choisir des professeurs ou des étudiants ?

(d) Combien y a-t-il de façons différentes de former un comité de 4 à 8 personnes de ce département, sans contrainte sur le fait de choisir des professeurs ou des étudiants ?

8.9 Vous programmez un générateur de mots. À partir des lettres fournies (avec possibilité de doublons), le générateur formera toutes les suites possibles et vérifiera si chacune des suites de lettres constitue un nom commun dans le dictionnaire.

(a) Combien de suites de 7 lettres formera le générateur si les lettres fournies sont BONJOUR ?

(b) Combien de suites de 8 lettres formera le générateur si les lettres fournies sont RAPLAPLA ?

(c) ★ Combien de suites de 7 ou 8 lettres formera le générateur si les lettres fournies sont RAPLAPLA ?

8.10 Une chaîne ternaire est une suite de symboles choisis parmi 0, 1 ou 2. Procédez aux dénombrements suivants.

(a) Combien de chaînes ternaires de longueur 7 ne contiennent aucun 0 ?

(b) Combien de chaînes ternaires de longueur 7 commencent par 012 et se terminent par 0 ?

(c) Combien de chaînes ternaires de longueur 7 commencent par 012 ou se terminent par 0 ? (Le *ou* est inclusif, comme toujours.)

(d) Combien de chaînes ternaires de longueur 7 commencent et finissent par le même symbole : 0*****0 ou 1*****1 ou 2*****2 ?

(e) Combien de chaînes ternaires de longueur 7 possèdent exactement trois 0 ?

(f) Combien de chaînes ternaires de longueur 7 possèdent au moins trois 0 ?

8.11 Un cadenas possède une serrure réinitialisable à code de 6 chiffres (parmi les chiffres 0 à 9). Par exemple, 935011 est un code possible.

(a) Combien peut-on former de codes possibles au total ?

(b) Combien de codes commencent par 45 ou se terminent par un chiffre pair ?

(c) Combien de codes possèdent au moins un 9 ?

(d) Combien de codes possèdent exactement un 0 et un 1 ?

(e) Combien de codes contiennent deux 0, deux 1, un 2 et un 3, si chaque 1 doit être suivi d'un 0 ?

(f) Combien de codes contiennent exactement trois 9 parmi les 5 premières positions ?

(g) Si les chiffres doivent tous être distincts, combien peut-on former de codes au total ?

(h) Si les chiffres doivent tous être distincts, combien de codes possèdent un 9 ?

(i) Si les chiffres doivent tous être distincts, combien de codes commencent et se terminent par un chiffre impair ?

8.4 Relations de récurrence et dénombrement

Rappellons qu'une **relation de récurrence** de la suite $\{a_n\}$ est une équation qui exprime a_n en termes de $a_0, a_1, \ldots, a_{n-1}$. Il est possible de résoudre des problèmes liés au dénombrement en utilisant les relations de récurrence.

Exemple 8.15

Soit a_n le nombre de trains de bits de longueur n qui ne contiennent pas la chaîne "11".

(a) Trouver une relation de récurrence pour a_n.

(b) Combien y a-t-il de trains de bits de longueur 8 qui ne contiennent pas deux 1 consécutifs?

Solution :

(a) On suppose que n est *suffisamment grand* et on considère tous les trains de bits comptés par a_n. Par le principe de la somme, le nombre de trains de bits de longueur n qui ne contiennent pas la chaîne "11" est égal au nombre de trains de bits de longueur n sans une occurrence de "11" qui commencent par 0, plus le nombre de trains de bits de longueur n sans une occurrence de "11" qui commencent par 1. Ces trains de bits sont donc de l'une des formes suivantes :

Forme des trains de bits			Nombre de trains de bits de cette forme
0	train de bits de lng $n-1$ sans une occurrence de 11		a_{n-1}
1	0	train de bits de lng $n-2$ sans une occurrence de 11	a_{n-2}

On en déduit la relation de récurrence

$$\begin{cases} a_1 = 2 & (0,1) \\ a_2 = 3 & (00,01,10) \\ a_n = a_{n-1} + a_{n-2}, & n \geq 3 \end{cases}$$

(b) $a_8 = 55$

Exemple 8.16

Soit b_n le nombre de trains de bits de longueur n qui contiennent au moins une occurrence de la chaîne "11".

(a) Trouver une relation de récurrence pour b_n.

(b) Combien y a-t-il de trains de bits de longueur 8 qui contiennent deux 1 consécutifs?

Solution :

(a) On suppose que n est *suffisamment grand* et on considère tous les trains de bits comptés par b_n. Par le principe de la somme, le nombre de trains de bits de longueur n qui contiennent la chaîne "11" est égal au nombre de trains de bits de longueur n avec au moins une occurrence de "11" qui commencent par 0, plus le nombre de trains de bits de longueur n avec au moins une occurrence de "11" qui commencent par 1. Ces trains de bits sont donc de l'une des formes suivantes :

Forme des trains de bits	Nombre de trains de bits de cette forme
0 | train de bits de lng $n-1$ avec au moins une occurrence de 11	b_{n-1}
1 0 | train de bits de lng $n-2$ avec au moins une occurrence de 11	b_{n-2}
1 1 | train de bits quelconque de lng $n-2$	2^{n-2}

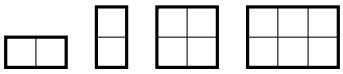
On en déduit la relation de récurrence

$$\begin{cases} b_1 = 0 & \\ b_2 = 1 & \text{le seul train de bits 11} \\ b_n = b_{n-1} + b_{n-2} + 2^{n-2}, & n \geq 3 \end{cases}$$

(b) $b_8 = 201$. On vérifie que $b_8 = 2^8 - a_8 = 2^8 - 55 = 201$.

Exemple 8.17

Soit a_n le nombre de façons de paver un trottoir de dimensions $2 \times n$ en utilisant des tuiles rectangulaires de dimensions 1×2, 2×1, 2×2 et 2×3 :

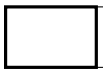


(a) Donnez une relation de récurrence pour a_n, avec les conditions initiales appropriées.

(b) Combien de façons y a-t-il de paver un plancher de dimension 2×18 en utilisant les tuiles décrites ci-haut ?

Solution :

(a) On suppose que n est *suffisamment grand* et on considère tous les pavages comptés par a_n. Ces pavages sont forcément de l'une des formes suivantes :

Forme des pavages	Nombre de pavages de cette forme
 pavage de dimensions $2 \times (n-1)$	a_{n-1}
 pavage de dimensions $2 \times (n-2)$	a_{n-2}
 pavage de dimensions $2 \times (n-2)$	a_{n-2}
 pavage de dimensions $2 \times (n-3)$	a_{n-3}

On en déduit la relation de récurrence

$$\begin{cases} a_1 = 1 \\ a_2 = 3 \\ a_n = a_{n-1} + 2a_{n-2} + a_{n-3}, \quad n \geq 3 \end{cases}$$

(b) $a_{18} = 578\,949$.

Exercices

8.12 Soit b_n le nombre de chaînes ternaires (avec des 0, 1, ou 2) de longueur n qui contiennent au moins deux 1 consécutifs.

(a) Déterminez b_0 et b_1.

(b) Trouvez une relation de récurrence pour b_n.

(c) Combien y a-t-il de chaînes ternaires de longueur 8 qui contiennent au moins deux 1 consécutifs ?

8.13 Soit a_n le nombre de chaînes ternaires de longueur n ne comportant pas trois 0 consécutifs.

(a) Établissez une relation de récurrence avec conditions initiales pour a_n. Expliquez pourquoi la relation de récurrence donnée correspond bien à la situation.

(b) Donnez la valeur de a_{15}.

(c) Quelle proportion des chaînes ternaires de longueur 15 n'ont pas trois 0 consécutifs ?

(d) Combien de chaînes ternaires de longueur 7 possèdent au moins trois 0 consécutifs ?

8.14 Soit a_n le nombre de chaînes ternaires de longueur n qui ne contiennent aucune occurrence de la chaîne "000" ni de la chaîne "01".

(a) Donnez a_1, a_2 et a_3 (conditions initiales).

(b) Trouvez une relation de récurrence satisfaite par a_n, pour $n \geq 4$.

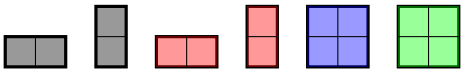
(c) Combien de chaînes ternaires de longueur 15 ne contiennent aucune occurrence de la chaîne "000" ni de la chaîne "01" ?

8.15 Soit b_n le nombre de trains de bits de longueur n qui ne contiennent pas la chaîne "10".

(a) Établissez une relation de récurrence avec conditions initiales pour b_n. Expliquez pourquoi la relation de récurrence donnée correspond bien à la situation.

(b) Donnez la valeur de b_{30}.

8.16 Soit a_n le nombre de façons de paver un trottoir de dimensions $2 \times n$ en utilisant des tuiles rectangulaires de dimensions 1×2 (et 2×1) de couleur noire et rouge, de dimensions 2×2 de couleur bleue et verte :



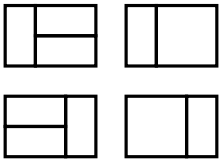
(a) Donnez une relation de récurrence pour a_n, avec les conditions initiales appropriées.

(b) Combien de façons y a-t-il de paver un plancher de dimension 2×15 en utilisant les tuiles décrites ci-haut ?

8.17 Considérons les tuiles rectangulaires de dimensions 1×2, 2×1 et 2×2 :

Soit a_n le nombre de pavages d'un trottoir $2 \times n$ avec ces tuiles, sans le motif  .

Par exemple, $a_3 = 4$ puisque les quatre pavages valides d'un trottoir 2×3 sont les suivants :



(a) Trouvez a_1, a_2 et a_4.

(b) Donnez une relation de récurrence satisfaite par a_n, ainsi que les cas de bases appropriés.

(c) Calculez a_{15}.

Réponses

Chapitre 1

Rép. 1.1 (a) Bloc (1) : exécuté lorsque $p \wedge q \vee r$ est **vrai**.
- `p` = **vrai**, `q` = **vrai** et `r` = **vrai**.
- `p` = **vrai**, `q` = **vrai** et `r` = **faux**.
- `p` = **faux**, `q` = **vrai** et `r` = **vrai**.
- `p` = **vrai**, `q` = **faux** et `r` = **vrai**.
- `p` = **faux**, `q` = **faux** et `r` = **vrai**.

Bloc (2) : exécuté lorsque $\neg(p \wedge q \vee r)$ est **vrai**.
- `p` = **vrai**, `q` = **faux** et `r` = **faux**.
- `p` = **faux**, `q` = **vrai** et `r` = **faux**.
- `p` = **faux**, `q` = **faux** et `r` = **faux**.

(b) Bloc (1) : exécuté lorsque $p \wedge q \wedge r$ est **vrai**.
- `p` = **vrai**, `q` = **vrai** et `r` = **vrai**.

Bloc (2) : exécuté lorsque $p \wedge \neg q$ est **vrai**.
- `p` = **vrai**, `q` = **faux** et `r` = **vrai**.
- `p` = **vrai**, `q` = **faux** et `r` = **faux**.

(c) Bloc (1) : exécuté lorsque $p \wedge q$ est **vrai**.
- `p` = **vrai**, `q` = **vrai** et `r` = **vrai**.
- `p` = **vrai**, `q` = **vrai** et `r` = **faux**.

Bloc (2) : exécuté lorsque $\neg(p \wedge q) \wedge r$ est **vrai**.
- `p` = **vrai**, `q` = **faux** et `r` = **vrai**.
- `p` = **faux**, `q` = **vrai** et `r` = **vrai**.
- `p` = **faux**, `q` = **faux** et `r` = **vrai**.

Bloc (3) : exécuté lorsque $\neg(p \wedge q) \wedge \neg r$ est **vrai**.
- `p` = **vrai**, `q` = **faux** et `r` = **faux**.
- `p` = **faux**, `q` = **vrai** et `r` = **faux**.
- `p` = **faux**, `q` = **faux** et `r` = **faux**.

Rép. 1.2 (a) $p \wedge q$ (c) $p \wedge \neg q$ (e) $p \rightarrow q$ (g) $p \leftrightarrow q$
(b) $q \rightarrow p$ (d) $p \vee q$ (f) $\neg p \wedge \neg q$ (h) $\neg p \rightarrow \neg q$

Rép. 1.3 La phrase (b) est la réciproque de (e). La phrase (h) est l'inverse de (e). Aucune des phrases n'est la contraposée de (e). La contraposée de (e) serait, par exemple : « Si l'eau ne gèle pas, alors la température n'est pas au-dessous de 0°C. »

Rép. 1.4 (a) La condition $x > 5$ est suffisante pour que le nombre x soit positif (mais elle n'est pas nécessaire).
(b) La condition $x > -5$ est nécessaire pour que x soit positif (mais elle n'est pas suffisante).
(c) La condition $x > -1$ est nécessaire et suffisante pour que l'entier x soit positif.
(d) Pour que le nombre x soit premier, il est nécessaire qu'il soit supérieur ou égal à 2.
(e) Pour qu'un triangle soit rectangle, il est suffisant que ses côtés mesurent respectivement 3, 4 et 5 centimètres.

Rép. 1.5 (a) « Il suffit qu'un nombre soit supérieur à 5 pour qu'il soit positif. »
(b) « Il est nécessaire qu'un nombre soit supérieur à -5 pour qu'il soit positif. »
(c) « Il faut et il suffit qu'un nombre entier soit supérieur à -1 pour qu'il soit positif. »
(d) La condition « $x \geq 2$ » est nécessaire pour que le nombre x soit premier.
(e) La condition « les côtés du triangle mesurent respectivement 3, 4 et 5 centimètres » est suffisante pour que le triangle soit rectangle.

Rép. 1.6 (a) Vrai. *La condition est nécessaire : si les quatre angles du quadrilatère Q ne mesurent pas 90°, alors Q n'est pas un carré.*
(b) Faux. *Il est nécessaire que Q ait deux paires de côtés parallèles pour que Q soit un rectangle, mais cette condition n'est pas suffisante. Autrement dit, les parallélogrammes ne sont pas tous des rectangles.*

Rép. 1.7 (a) $q \to p$ (c) $p \to q$ (e) $q \wedge \neg p$ (g) $\neg q \to \neg p$ (i) $p \vee \neg q$
(b) $p \wedge \neg q$ (d) $\neg p \to \neg q$ (f) $q \to p$ (h) $\neg p \wedge \neg q$ (j) $q \leftrightarrow p$

Rép. 1.8 La phrase (g) est la contraposée de (c). Les phrases (a) et (f) sont des réciproques de (c). La phrase (d) est l'inverse de (c).

Rép. 1.9 (a) p est nécessaire et suffisant pour q
(b) p est nécessaire pour q
(c) p est nécessaire et suffisant pour q
(d) p est suffisant pour q
(e) p est nécessaire pour q
(f) p est suffisant pour q
(g) p est suffisant pour q

Rép. 1.10 (a)

p	q	$p \to q$	$\neg q$	$(p \to q) \wedge \neg q$
V	**V**	**V**	**F**	**F**
V	**F**	**F**	**V**	**F**
F	**V**	**V**	**F**	**F**
F	**F**	**V**	**V**	**V**

(b)

p	q	$p \to q$	$\neg p \vee q$	$(p \to q) \leftrightarrow (\neg p \vee q)$
V	**V**	**V**	**V**	**V**
V	**F**	**F**	**F**	**V**
F	**V**	**V**	**V**	**V**
F	**F**	**V**	**V**	**V**

(c)

p	q	r	$p \to q$	$(p \to q) \to r$
V	**V**	**V**	**V**	**V**
V	**V**	**F**	**V**	**F**
V	**F**	**V**	**F**	**V**
V	**F**	**F**	**F**	**V**
F	**V**	**V**	**V**	**V**
F	**V**	**F**	**V**	**F**
F	**F**	**V**	**V**	**V**
F	**F**	**F**	**V**	**F**

(d)

p	q	r	$q \to r$	$p \to (q \to r)$
V	**V**	**V**	**V**	**V**
V	**V**	**F**	**F**	**F**
V	**F**	**V**	**V**	**V**
V	**F**	**F**	**V**	**V**
F	**V**	**V**	**V**	**V**
F	**V**	**F**	**F**	**V**
F	**F**	**V**	**V**	**V**
F	**F**	**F**	**V**	**V**

Rép. 1.11 (a) La proposition est une contradiction, car elle est toujours fausse, comme le montre sa table de vérité.

p	$p \vee p$	$\neg(p \wedge p)$	$(p \vee p) \leftrightarrow \neg(p \wedge p)$
V	**V**	**F**	**F**
F	**F**	**V**	**F**

(b) La proposition est une tautologie, car elle est toujours vraie, comme le montre sa table de vérité.

p	q	$p \oplus q$	$\neg(p \wedge q)$	$(p \oplus q) \rightarrow \neg(p \wedge q)$
V	**V**	**F**	**F**	**V**
V	**F**	**V**	**V**	**V**
F	**V**	**V**	**V**	**V**
F	**F**	**F**	**V**	**V**

(c) La proposition est une contingence, car elle est parfois vraie et parfois fausse, comme le montrent les deux lignes suivantes de sa table de vérité.

p	q	r	$q \wedge r$	$p \vee q$	$p \vee (q \wedge r)$	$(p \vee q) \wedge r$
V	**V**	**V**	**V**	**V**	**V**	**V**
V	**V**	**F**	**F**	**V**	**V**	**F**

(d) La proposition est une tautologie, car elle est toujours vraie, comme le montre sa table de vérité.

p	q	$p \rightarrow q$	$\neg(p \rightarrow q)$	$\neg(p \rightarrow q) \rightarrow p$
V	**V**	**V**	**F**	**V**
V	**F**	**F**	**V**	**V**
F	**V**	**V**	**F**	**V**
F	**F**	**V**	**F**	**V**

Rép. 1.12 (a)

p	q	$p \rightarrow q$	$\neg q \rightarrow \neg p$	$(p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p)$
V	**V**	**V**	**V**	**V**
V	**F**	**F**	**F**	**V**
F	**V**	**V**	**V**	**V**
F	**F**	**V**	**V**	**V**

(b)

p	q	$\neg(p \vee q)$	$\neg p \wedge \neg q$	$(\neg(p \vee q)) \leftrightarrow (\neg p \wedge \neg q)$
V	**V**	**F**	**F**	**V**
V	**F**	**F**	**F**	**V**
F	**V**	**F**	**F**	**V**
F	**F**	**V**	**V**	**V**

Rép. 1.13 (a) « Ce programme n'est pas rapide ou il n'est pas efficace. » Ou encore, « ce programme n'est pas rapide ou il est inefficace. »

(b) « Le programme ne doit pas être C++ et le programme ne doit pas être en Java. » Ce qui serait mieux de reformuler ainsi : « le programme ne doit être ni en C++, ni en Java. »

Rép. 1.14 Il suffit d'appliquer la loi de De Morgan $\neg(p \wedge q) \equiv \neg p \vee \neg q$.

Rép. 1.15 (a)

p	q	$\neg(p \vee (\neg p \wedge q))$	$\neg p \wedge \neg q$
V	**V**	**F**	**F**
V	**F**	**F**	**F**
F	**V**	**F**	**F**
F	**F**	**V**	**V**

(b) On ne sait pas quel sera le plus court chemin vers la vérité ! Voici deux preuves.

$$
\begin{array}{lcll}
\neg(p \vee (\neg p \wedge q)) & \equiv & \neg p \wedge \neg(\neg p \wedge q) & \text{De Morgan} \\
& \equiv & \neg p \wedge (\neg(\neg p) \vee \neg q) & \text{De Morgan} \\
& \equiv & \neg p \wedge (p \vee \neg q) & \text{Double négation} \\
& \equiv & (\neg p \wedge p) \vee (\neg p \wedge \neg q) & \text{Distributivité} \\
& \equiv & \mathbf{F} \vee (\neg p \wedge \neg q) & \text{Négation} \\
& \equiv & \neg p \wedge \neg q & \text{Identité}
\end{array}
$$

Ou encore, en commençant par distribuer la disjonction (ou) :

$$\begin{array}{lcll} \neg(p \vee (\neg p \wedge q)) & \equiv & \neg((p \vee \neg p) \wedge (p \vee q)) & \text{Distributivité} \\ & \equiv & \neg(\mathbf{V} \wedge (p \vee q)) & \text{Négation} \\ & \equiv & \neg(p \vee q) & \text{Identité} \\ & \equiv & \neg p \wedge \neg q & \text{De Morgan} \end{array}$$

(c) $\text{not}\,(p \text{ or not } p \text{ and } q)$ $\qquad$ $\text{not } p \text{ and not } q$

Rép. 1.16 (a) Les propositions ne sont pas équivalentes puisque quand p, q, et r sont toutes les trois fausses, $(p \to q) \to r$ est fausse, mais $p \to (q \to r)$ est vraie.

(b) Les propositions sont équivalentes. Justification : pour que $(p \to r) \vee (q \to r)$ soit fausse, les deux implications doivent être fausses, ce qui arrive exactement quand r est fausse et p et q sont vraies. Il en va de même pour $(p \wedge q) \to r$.

Rép. 1.17 (a) Voici une preuve possible :

$$\begin{array}{lcll} (\neg q \to p) \to (r \vee p) & \equiv & (\neg\neg q \vee p) \to (r \vee p) & \text{Table 2 équivalence 1} \\ & \equiv & (q \vee p) \to (r \vee p) & \text{Double négation} \\ & \equiv & \neg(q \vee p) \vee (r \vee p) & \text{Table 2 équivalence 1} \\ & \equiv & (\neg q \wedge \neg p) \vee (r \vee p) & \text{De Morgan} \\ & \equiv & (p \vee (\neg q \wedge \neg p)) \vee r & \text{Associativité et Commutativité} \\ & \equiv & ((p \vee \neg q) \wedge (p \vee \neg p)) \vee r & \text{Distributivité} \\ & \equiv & ((p \vee \neg q) \wedge \mathbf{V}) \vee r & \text{Négation} \\ & \equiv & (p \vee \neg q) \vee r & \text{Identité} \\ & \equiv & \neg(\neg p \wedge q) \vee r & \text{De Morgan} \\ & \equiv & (\neg p \wedge q) \to r & \text{Table 2 équivalence 1} \end{array}$$

(b) Voici une preuve possible :

$$\begin{array}{lcll} ((p \vee q) \wedge (\neg p \vee r)) \to (q \vee r) & \equiv & \neg((p \vee q) \wedge (\neg p \vee r)) \vee (q \vee r) & \text{Table 2 équiv. 1} \\ & \equiv & \neg(p \vee q) \vee \neg(\neg p \vee r) \vee (q \vee r) & \text{De Morgan} \\ & \equiv & (\neg p \wedge \neg q) \vee (p \wedge \neg r) \vee (q \vee r) & \text{De Morgan} \\ & \equiv & (q \vee (\neg p \wedge \neg q)) \vee (r \vee (p \wedge \neg r)) & \text{Associativité et Comm.} \\ & \equiv & ((q \vee \neg p) \wedge (q \vee \neg q)) \vee ((r \vee p) \wedge (r \vee \neg r)) & \text{Distributivité} \\ & \equiv & ((q \vee \neg p) \wedge \mathbf{V}) \vee ((r \vee p) \wedge \mathbf{V}) & \text{Négation} \\ & \equiv & (q \vee \neg p) \vee (r \vee p) & \text{Identité} \\ & \equiv & (q \vee r) \vee (\neg p \vee p) & \text{Associativité et Comm.} \\ & \equiv & (q \vee r) \vee \mathbf{V} & \text{Négation} \\ & \equiv & \mathbf{V} & \text{Domination} \end{array}$$

Rép. 1.18 (a) Il y a plusieurs réponses possibles (en raison des équivalences logiques) ; nous avons choisi celles qui se rapprochent le plus des phrases décrivant les contraintes.

1. Au moins un des postulants numéro 2, 4 ou 5 doit être embauché, car eux seuls ont des connaissances en électricité.
$$p_2 \vee p_4 \vee p_5$$
2. Si le postulant 2 est embauché ou le postulant 4 est embauché, alors le postulant 5 ne doit pas l'être.
$$(p_2 \vee p_4) \to \neg p_5$$
3. Si les postulants 2 et 4 sont embauchés tous les deux, alors le postulant 5 ne doit pas l'être.
$$(p_2 \wedge p_4) \to \neg p_5$$

4. Les postulants 3 et 5 forment un couple : ils accepteront l'affectation seulement si les deux sont embauchés.
$$p_3 \leftrightarrow p_5$$
5. Il est impossible d'embaucher à la fois les postulants 2 et 5, et il est impossible d'embaucher à la fois les postulants 4 et 5.
$$\neg(p_2 \wedge p_5) \wedge \neg(p_4 \wedge p_5)$$
6. Il est impossible d'embaucher le trio de postulants 2, 4 et 5 : on peut en choisir un, deux ou aucun, mais pas les trois.
$$\neg(p_2 \wedge p_4 \wedge p5)$$
7. Il faut absolument embaucher au moins 2 personnes parmi les postulants 1 à 4.
$$p_1 \wedge p_2 \vee p_1 \wedge p_3 \vee p_1 \wedge p_4 \vee p_2 \wedge p_3 \vee p_2 \wedge p_4 \vee p_3 \wedge p_4$$
qui est plus facile à lire si on ajoute des parenthèses :
$$(p_1 \wedge p_2) \vee (p_1 \wedge p_3) \vee (p_1 \wedge p_4) \vee (p_2 \wedge p_3) \vee (p_2 \wedge p_4) \vee (p_3 \wedge p_4)$$
8. Il faut embaucher les postulants 1 et 3, ou embaucher le postulant 2 et au moins un des postulants parmi 1, 3 et 4, ou encore embaucher le postulant 4 et au moins un des postulants parmi 1, 2 et 3.
$$(p_1 \wedge p_3) \vee (p_2 \wedge (p_1 \vee p_3 \vee p_4)) \vee (p_4 \wedge (p_1 \vee p_2 \vee p_3))$$

(b) Les contraintes C2 et C5 sont équivalentes, en voici la preuve.

$$\begin{array}{lll} & (p_2 \vee p_4) \rightarrow \neg p_5 & \\ \equiv & \neg(p_2 \vee p_4) \vee \neg p_5 & \text{Règle 2.1} \\ \equiv & (\neg p_2 \wedge \neg p_4) \vee \neg p_5 & \text{De Morgan} \\ \equiv & (\neg p_2 \vee \neg p_5) \wedge (\neg p_4 \vee \neg p_5) & \text{Distributivité} \\ \equiv & \neg(p_2 \wedge p_5) \wedge \neg(p_4 \wedge p_5) & \text{De Morgan} \end{array}$$

Les contraintes C3 et C6 sont équivalentes, en voici la preuve.

$$\begin{array}{lll} & (p_2 \wedge p_4) \rightarrow \neg p_5 & \\ \equiv & \neg(p_2 \wedge p_4) \vee \neg p_5 & \text{Règle 2.1} \\ \equiv & (\neg p_2 \vee \neg p_4) \vee \neg p_5 & \text{De Morgan} \\ \equiv & \neg p_2 \vee \neg p_4 \vee \neg p_5 & \text{Associativité} \\ \equiv & \neg(p_2 \wedge p_4 \wedge p_5) & \text{De Morgan} \end{array}$$

Les contraintes C7 et C8 sont équivalentes, en voici la preuve.

$$\begin{array}{lll} & (p_1 \wedge p_3) \vee (p_2 \wedge (p_1 \vee p_3 \vee p_4)) \vee (p_4 \wedge (p_1 \vee p_2 \vee p_3)) & \\ \equiv & (p_1 \wedge p_3) \vee (p_2 \wedge p_1) \vee (p_2 \wedge p_3) \vee (p_2 \wedge p_4) \vee (p_4 \wedge p_1) \vee (p_4 \wedge p_2) \vee (p_4 \wedge p_3) & \text{Distributivité} \\ \equiv & (p_1 \wedge p_3) \vee (p_1 \wedge p_2) \vee (p_2 \wedge p_3) \vee (p_2 \wedge p_4) \vee (p_1 \wedge p_4) \vee (p_2 \wedge p_4) \vee (p_3 \wedge p_4) & \text{Commutativité} \\ \equiv & (p_1 \wedge p_3) \vee (p_1 \wedge p_2) \vee (p_2 \wedge p_3) \vee (p_2 \wedge p_4) \vee (p_1 \wedge p_4) \vee (p_3 \wedge p_4) & \text{Idempotence} \\ \equiv & (p_1 \wedge p_2) \vee (p_1 \wedge p_3) \vee (p_1 \wedge p_4) \vee (p_2 \wedge p_3) \vee (p_2 \wedge p_4) \vee (p_3 \wedge p_4) & \text{Commutativité} \end{array}$$

Rép. 1.19 (a) Vrai. Un exemple suffit pour démontrer une proposition existentielle : $P(1)$ est vrai.

(b) Faux. Un contre-exemple suffit pour infirmer une proposition universelle. Autrement dit, pour conclure qu'une proposition universelle est fausse, un seul exemple qui la contredit suffit. Ici, $P(2)$ est faux.

(c) Vrai. Un exemple suffit pour démontrer une proposition existentielle : $P(2)$ est faux donc $\neg P(2)$ est vrai.

(d) Faux. Un contre-exemple suffit, or $\neg P(1)$ est faux.

Rép. 1.20 (a) $P(0) \vee P(1) \vee P(2)$
(b) $P(0) \wedge P(1) \wedge P(2)$
(c) $\neg\big(P(0) \vee P(1) \vee P(2)\big) \equiv \neg P(0) \wedge \neg P(1) \wedge \neg P(2)$
(d) $\neg\big(P(0) \wedge P(1) \wedge P(2)\big) \equiv \neg P(0) \vee \neg P(1) \vee \neg P(2)$
(e) $\neg P(0) \vee \neg P(1) \vee \neg P(2))$
(f) $\neg P(0) \wedge \neg P(1) \wedge \neg P(2)$

Rép. 1.21 (a) 1
(b) 8
(c) 4
(d) 7
(e) 9
(f) 8
(g) 3
(h) 9
(i) 3

Rép. 1.22 (a) Vrai: par exemple, Baba est ingénieur et sait programmer.
(b) Faux: par exemple, Denis n'est pas ingénieur et ne sait pas programmer.
(c) Faux. $\forall x\, \neg\big(I(x) \vee P(x)\big) \equiv \forall x\, \big(\neg I(x) \wedge \neg P(x)\big)$ Or cet énoncé est faux, car par exemple, Alain sait programmer.
(d) Vrai. Par exemple, pour $x = France$.
(e) Vrai: $\big(P(Alain) \wedge \neg I(Alain)\big)$
(f) Faux car $P(Alain)$.
(g) Vrai, les 3 ingénieurs savent tous programmer.
(h) Faux: Alain sait programmer, mais il n'est pas ingénieur.
(i) Faux: Alain sait programmer, mais il n'est pas ingénieur.

Rép. 1.23 Rappel: il peut y avoir plusieurs formulations équivalentes. Si la vôtre est différente de la réponse ci-dessous, n'hésitez pas à consulter votre prof!

(a) $\exists x\, \big(M(x) \wedge J(x)\big)$
(b) $\exists x\, \big(M(x) \wedge \neg J(x)\big)$
(c) $\forall x\, J(x)$
(d) $\neg\exists x\, \big(M(x) \vee J(x)\big) \equiv \forall x\, \neg\big(M(x) \vee J(x)\big) \equiv \forall x\, \big(\neg M(x) \wedge \neg J(x)\big)$
(e) $\forall x\, \big(M(x) \rightarrow J(x)\big)$

Rép. 1.24 (a) Vrai. Un exemple suffit pour démontrer une proposition existentielle: prendre $n = 0$.
(b) Faux. Un contre-exemple suffit pour infirmer proposition universelle: prendre $n = -1$.
(c) Faux. Un exemple ne suffit pas, un argument général serait requis pour démontrer que cette proposition existentielle est fausse.
(d) Faux. Un contre-exemple suffit pour infirmer proposition universelle: prendre $n = 0$.
(e) Vrai. Un exemple ne suffit pas, un argument général serait requis.

Rép. 1.25 (a) $\exists x\, A(x) \wedge \neg V(x)$
(b) $\forall x\, \big(V(x) \rightarrow A(x)\big)$
(c) $\forall x\, \neg V(x)$
(d) $\neg\forall x\, V(x)$
(e) $\forall x\, \big(A(x) \rightarrow V(x)\big)$
(f) $\exists x\, \big(A(x) \wedge V(x)\big)$
(g) $\forall x\, \big(A(x) \wedge V(x)\big)$ ou encore $\big(\forall x\, A(x)\big) \wedge \big(\forall x\, V(x)\big)$
(h) $\big(\neg\forall x\, A(x)\big) \vee \big(\exists x\, \neg V(x)\big)$

Rép. 1.26 (a) L'énoncé est vrai, car $2^5 \bmod 10 = 32 \bmod 10 = 2$ et $4^3 \bmod 10 = 64 \bmod 10 = 4$.
(b) L'énoncé universel est faux. Prenons $x = 2$ comme contre-exemple:
$x^3 \bmod 10 = 8 \bmod 10 = 8 \neq x$. Ainsi, $T(2)$ est faux.
(c) L'énoncé existentiel est vrai, car $T(0)$ est vrai.
(d) L'énoncé existentiel est vrai, car $\neg T(2)$ est vrai.
(e) L'énoncé universel est vrai: on peut tester toutes les valeurs possibles pour x et conclure que $P(0) \wedge P(1) \wedge P(2) \wedge P(3) \wedge P(4) \wedge P(5) \wedge P(6) \wedge P(7) \wedge P(8) \wedge P(9)$ est vrai. (De façon plus générale, l'énoncé $x^5 \bmod 10 = x$ est vrai quelque soit la valeur du nombre entier x, mais ceci ne peut être démontré en testant toutes les possibilités: il faudrait fournir une preuve générale.)
(f) Faux, car $\exists x \neg P(x) \equiv \neg\forall x\, P(x)$ et $\forall x\, P(x)$ est vrai (voir (e)).

Rép. 1.27 2 et 3.

Rép. 1.28 Il y a plusieurs formulations possibles. Tentez d'être le plus clair possible, dans un bon français : reformulez vos traductions initiales pour leur donner une tournure plus naturelle, moins robotisée.

(a) $\forall B \in R,\ \exists x \in X, M(x,B)$ Chaque répertoire peut être modifé par au moins un employé (par nécessairement le même employé). **Vrai.**

(b) $\exists x \in X,\ \forall B \in R, M(x,B)$ Il existe au moins un employé qui peut modifier tous les répertoires (le même employé). **Faux.**

(c) $\exists B \in R,\ \forall x \in X, M(x,B)$ Il existe au moins un répertoire que tous les employés peuvent modifier. **Faux.**

(d) $\exists B \in R,\ \forall x \in X, L(x,B)$ Il existe au moins un répertoire que tous les employés peuvent lire. **Vrai** (P et Z).

(e) $\exists B \in R,\ \forall x \in X, \neg L(x,B)$ Il existe au moins un répertoire qu'aucun employé ne peut lire. **Faux.**

(f) $\forall x \in X,\ \exists B \in R, \neg L(x,B)$ Pour chaque employé, il y a au moins un répertoire qu'il ne peut pas lire. *Dans un langage plus naturel :* aucun employé ne peut lire tous les répertoires. **Vrai** car chaque colonne contient au moins une case sans la lettre L.

(g) $\forall B \in R, \big(L(\text{Manon}, B) \rightarrow M(\text{Guy}, B)\big)$ Pour chaque répertoire B, si ce répertoire B peut être lu par Manon, alors ce répertoire B peut être modifié par Guy. *Dans un langage plus naturel :* chaque répertoire qui peut être lu par Manon peut être modifié par Guy. **Vrai.**

(h) $\forall B \in R,\ \forall x \in X,\ \big(M(x,B) \rightarrow L(x,B)\big)$ Si un employé peut modifier un répertoire, alors il peut le lire. **Vrai.**

(i) $\exists B \in R,\ \exists! x \in X, M(x,B)$ Il existe au moins un répertoire pour lequel un et un seul employé a le droit de modification. **Vrai** (répertoires K, J, Z).

(j) $\exists! B \in R,\ \exists x \in X, M(x,B)$ Il existe un et un seul répertoire qui peut être modifié par au moins un employé. **Faux.**

(k) $\exists! B \in R,\ \exists! x \in X, L(x,B)$ Il existe un et un seul répertoire qui peut être lu par un et un seul employé. **Vrai**, le répertoire J.

Rép. 1.29

(a) $\forall y\, C(\text{Ève}, y)$

(b) $\exists y\, \neg C(\text{Julie}, y)$

(c) $\exists y\, \forall x\, \neg C(x,y)$

(d) $\forall x\, C(x,x)$

(e) $\exists x\, \forall y\, (C(x,y) \leftrightarrow x = y)$

(f) $\forall x\, C(x, \text{Karim})$

(g) $\forall x\, \exists y\, C(x,y)$

(h) $\exists y\, \forall x\, C(x,y)$

(i) $\forall x\, \exists y\, \neg C(x,y)$

Rép. 1.30 Il y a plusieurs formulations possibles. Tentez d'être le plus clair possible, dans un bon français : reformulez vos traductions initiales pour leur donner une tournure plus naturelle, moins robotisée.

(a) Il y a au moins une usine qui fabrique le produit c. **Vrai**, Amos par exemple.

(b) L'usine de Montréal et celle de Chicoutimi n'ont aucun produit en commun dans leur fabrication. Aucun produit n'est fabriqué à la fois par l'usine de Montréal et par celle de Chicoutimi. **Faux.** Par exemple, on voit que le produit a est fabriqué dans les deux usines.

(c) Si l'usine de Montréal fabrique un produit, alors ce produit n'est pas fabriqué à l'usine de Gatineau. **Vrai.** L'usine de Montréal fabrique les produits a, b et f et ceux-ci ne sont pas fabriqués à Gatineau.

(d) À elles deux, les usines de Montréal et Québec ne couvrent pas l'ensemble de la production : il y a au moins un produit qui n'est fabriqué ni à Montréal, ni à Québec. **Vrai,** car le produit c n'est fabriqué ni à Montréal, ni à Québec.

(e) Il y a au moins un produit qui est fabriqué dans chaque usine (il s'agit du même produit). **Faux.**

(f) Chacune des usines fabrique au moins un produit (pas nécessairement le même produit dans chacune des usine). **Vrai.**

(g) Chacune des usine fabrique chacun des produits. **Faux,** par exemple, Amos ne fabrique pas le produit a.

(h) Si une usine fabrique le produit b, alors elle ne fabrique ni le produit d, ni le produit e. Vrai.

(i) Si une usine fabrique le produit a, alors elle doit aussi fabriquer le produit f ou le produit g (ou les deux, le « ou » étant inclusif). **Faux.** Québec fabrique a, mais ne fabrique ni f, ni g.

(j) Chaque produit est fabriqué dans au moins une usine. **Vrai.**

(k) Aucune usine ne fabrique à la fois le produit c et le produit d. **Faux.** Amos fabrique les produits c et d entre autres.

(l) Les produits a et b doivent être fabriqués ensemble dans la même usine : une usine fabrique a si et seulement si elle fabrique b. **Faux,** car Québec fabrique a sans b.

(m) Il y a au moins 2 usines qui fabriquent le produit a. **Vrai,** Chicoutimi, Montréal et Québec.

(n) Il y a au moins un produit qui est fabriqué dans une seule usine (aucune autre usine ne le fabrique). **Vrai,** le produit g est fabriqué uniquement à Gatineau.

Rép. 1.31 (a) $\forall i\, \forall j\, \exists k\, C(i,j,k)$

(b) $\forall i\, \forall j\, \forall k\, \forall l\, \big(C(i,j,k) \wedge C(i,j,l) \to k = l\big)$

(c) $\forall j\, \forall k\, \exists i\, C(i,j,k)$

(d) $\forall i\, \forall k\, \exists j\, C(i,j,k)$

(e) $\forall i\, \forall j\, \forall k\, \forall l\, \big(c(i,j,k) \wedge c(i,l,k) \to j = l\big)$

(f) $\forall i\, \forall j\, \forall k\, \forall l\, \big(c(i,j,k) \wedge c(l,j,k) \to i = l\big)$

(g) $\forall k\, \forall r \in \{0,1,2\}\, \forall s \in \{0,1,2\}\, \exists i \in \{1,2,3\}\, \exists j \in \{1,2,3\},\ C(3r+i, 3s+j, k)$

(h) $\forall k\, \forall s, r \in \{0,1,2\}\, \forall a, b, a', b' \in \{1,2,3\},\ \big(c(3r+a, 3s+b, k) \wedge c(3r+a', 3s+b', k)\big) \to (a = a' \wedge b = b')$

Rép. 1.32 (a) Vrai. Il suffit d'un exemple pour prouver un énoncé existentiel. Puisque l'inégalité $3 < 4$ est vraie, la proposition $P(x,4)$ est vraie pour $x = 3$. Ainsi, l'énoncé existentiel $\exists x\, P(x,4)$ est vrai.

(b) Faux. Justification : aucun nombre naturel n'est inférieur à 0. Par contre, l'énoncé serait vrai si l'univers du discours était l'ensemble des nombres entiers.

(c) Vrai. Soit x quelconque. Posons $y = x + 1$. Alors $P(x,y)$ désigne $x < x + 1$ qui est toujours vrai. Donc l'énoncé est vrai.

(d) Vrai. $\neg\, \forall x\, \forall y\, P(x,y) \equiv \exists x\, \exists y\, \neg P(x,y)$. Un exemple suffit. Posons $x = 1$ et $y = 0$, alors $P(x,y)$ est faux, $\neg P(x,y)$ est vrai et l'énoncé est vrai.

(e) Vrai. $\neg\, \exists x\, \forall y\, P(x,y) \equiv \forall x\, \exists y\, \neg P(x,y)$. Soit x quelconque. Posons $y = x$. Alors $\neg P(x,y)$ est vrai. Donc l'énoncé est vrai.

(f) Vrai. Prenons $x = 0$. Quelque soit le nombre naturel non nul y, on a $x < y$. Par ailleurs, l'énoncé serait faux si l'univers du discours était l'ensemble des nombres entiers ou l'ensemble des nombres réels : dans ces ensembles, il n'y a aucun nombre qui est inférieur à tous les autres ; il n'y a pas de nombre minimal.

(g) Faux. On considère un contre-exemple, prenons $x = 5$ et $y = 5$. Alors $P(x,y)$ est faux, $P(y,x)$ est également faux et donc, $P(x,y) \vee P(y,x)$ est faux.

Rép. 1.33 (a) $x = 1$ et $y = -1$

(b) $x = 1$ et $y = -1$

(c) Pour $x = 0$ et y quelconque, la proposition $xy = 1$ est fausse.

Rép. 1.34 On pose :
p : « il pleut »,
m : « les trottoirs sont mouillés ».

(a)
$$\begin{array}{l} p \to m \\ \neg p \\ \hline \neg m \end{array}$$

Raisonnement invalide, car si $p = \mathbf{F}$, $m = \mathbf{V}$ alors,

$$\begin{aligned} \big((p \to m) \wedge \neg p\big) \to \neg m &\equiv ((\mathbf{F} \to \mathbf{V}) \wedge \neg \mathbf{F}) \to \neg \mathbf{V} \\ &\equiv (\mathbf{V} \wedge \mathbf{V}) \to \mathbf{F} \\ &\equiv \mathbf{F}. \end{aligned}$$

(b)
$$\begin{array}{l} p \to m \\ p \\ \hline m \end{array}$$

Raisonnement valide par *modus ponens*.

(c) On pose :
e : « tu fais tous les exercices du livre de référence »,
r : « tu réussis le cours ».

$$\begin{array}{l} e \to r \\ r \\ \hline e \end{array}$$

Raisonnement invalide, car si $e = \mathbf{F}$, $r = \mathbf{V}$ alors,

$$\begin{array}{rcl} ((e \to r) \wedge r) \to e & \equiv & ((\mathbf{F} \to \mathbf{V}) \wedge \mathbf{V}) \to \mathbf{F} \\ & \equiv & (\mathbf{V} \wedge \mathbf{V}) \to \mathbf{F} \\ & \equiv & \mathbf{F}. \end{array}$$

(d) On pose :
s : « la science peut expliquer ce phénomène »,
m : « c'est un miracle ».

$$\begin{array}{l} s \vee m \\ \neg s \\ \hline m \end{array}$$

Raisonnement valide par *syllogisme disjonctif*.

Remarque : On aurait également pu interpréter la première affirmation comme étant un « *ou exclusif* ». Dans ce cas, on ne peut pas utiliser directement les règles d'inférence :

$$\begin{array}{ll} s \oplus m & \\ \neg s & \\ \hline (s \vee m) \wedge \neg(s \wedge m) & \text{logiquement équivalente à la l'hypothèse } s \oplus m. \\ s \vee m & \text{par } \textit{simplification} \text{ de la ligne précédente.} \\ m & \text{par } \textit{syllogisme disjonctif}. \end{array}$$

(e) On pose :
v : « ce sandwich contient de la viande »,
c : « ce sandwich contient des champignons »,
m : « ce sandwich contient de la mayonnaise ».

$$\begin{array}{l} \neg v \vee c \\ v \vee m \\ \hline c \vee m \end{array}$$

Raisonnement valide par *résolution*.

Rép. 1.35 (a) $\exists x \in U \, \forall y \in U \, \big(P(x) \wedge (P(y) \to x = y)\big)$.
(b) $\exists x \in U \, \exists y \in U \, \big(P(x) \wedge P(y) \wedge x \neq y\big)$.

Rép. 1.36 On montre que les cinq hypothèses de l'énoncé mènent à la conclusion : *Le téléphone est sur le comptoir de la cuisine*. Afin de simplifier la rédaction, on pose :

t : « le téléphone est sur la table de chevet »,
v : « j'ai vu mon téléphone en me levant »,
r : « j'ai été réveillée par la sonnerie de mon téléphone »,
p : « j'ai parlé au téléphone dans la cuisine »,
c : « le téléphone est sur le comptoir de la cuisine ».

On considère le raisonnement suivant :

1. $t \to v$
2. $r \to t$
3. $p \to c$
4. $r \vee p$
5. $\neg v$
6. $\overline{r \to v}$ par *syllogisme hypothétique* des lignes 2 et 1.
7. $\neg r$ par *modus tollens* des lignes 5 et 6.
8. p par *syllogisme disjonctif* des lignes 4 et 7.
9. c par *modus ponens* des lignes 8 et 3.

Rép. 1.37

1. $a \vee \neg b \to \neg c \wedge d$
2. $e \to c \vee \neg d$
3. $\neg f$
4. $e \vee f$
5. $\overline{e}$ par *syllogisme disjonctif* des lignes 4 et 3.
6. $\neg(\neg c \wedge d) \to \neg(a \vee \neg b)$ équivalent à la ligne 1 par Table 2, ligne 2 (contraposée).
7. $(c \vee \neg d) \to (\neg a \wedge b)$ équivalent à la ligne 6 par *De Morgan* et *double négation*.
8. $e \to (\neg a \wedge b)$ par *syllogisme hypothétique* des lignes 2 et 7.
9. $\neg a \wedge b$ par *modus ponens* des lignes 5 et 8.

Rép. 1.38 Chaque prénom est abrégé par sa première lettre, en minuscule.

1. $\exists! x\, C(x)$ voir 1.35 pour symbole $\exists!$
2. $\forall x \big(C(x) \to P(x)\big)$
3. $\exists x \big(P(x) \wedge \neg C(x)\big)$
4. $P(a) \wedge P(d) \to C(c)$
5. $P(a) \vee P(c) \to C(b)$
6. $\neg P(d)$
7. $\neg C(d) \to \neg C(a)$
8. $P(a) \leftrightarrow \neg P(c)$

Rép. 1.39 On peut déduire que Benoît est coupable. Voici le détail d'un raisonnement possible permettant d'arriver à cette conclusion.

9.	$\neg C(d)$	*inst. univ.* de 2 et *modus tollens* avec 6.
10.	$\neg C(a)$	*modus ponens* de 7 et 9.
11.	$\exists! x\, C(x) \wedge P(x)$	*modus ponens* de 1 et 2.
12.	$\exists x \exists y, (x \neq y) \wedge P(x) \wedge P(y)$	par 3 et 11, car $C(x) \wedge \neg C(x) \equiv \mathbf{F}$.
13.	$(P(a) \wedge P(b)) \vee (P(a) \wedge P(c)) \vee (P(b) \wedge P(c))$	par à 12 et 6, car $U = \{a, b, c, d\}$
14.	$\neg(P(a) \wedge P(c))$	par 8 et table 3.4.
15.	$(P(a) \wedge P(b)) \vee (P(b) \wedge P(c))$	par *syllogisme disjonctif* de 13 et 14.
16.	$P(b) \wedge (P(a) \vee P(c))$	équivalent à 15 par *distributivité*.
17.	$P(a) \vee P(c)$	par *simplification* de 16.
18.	$C(b)$	par *modus ponens* de 17 et 5.

Remarque 1. Voici comment déduire 12 à partir de 3 et 11 sans s'embourber dans les règles logiques : il y a au moins une personne présente et non coupable (3), il y a une personne présente et coupable (11) ; on peut donc déduire qu'il y a au moins deux personnes présentes.
Remarque 2. Il y a d'autres façons de résoudre ce type de problème, comme l'utilisation de tableaux.

Rép. 1.40 On pose :
$M(x)$: « x est inscrit au cours de Mathématiques discrètes »,
$E(x)$: « x a fait les exercices suggérés »,
$I(x)$: « x a réussi l'intra ».

1.	$\exists x(M(x) \wedge \neg E(x))$	
2.	$\forall x(M(x) \rightarrow I(x))$	
3.	$M(c) \wedge \neg E(c)$	pour un un certain c, par *instanciation existentielle* de 1.
4.	$M(c)$	par *simplification* de 3.
5.	$M(c) \rightarrow I(c)$	par *instanciation universelle* de 2.
6.	$I(c)$	par *modus ponens* de 4 et 5.
7.	$\neg E(c)$	par *simplification* de 3.
8.	$I(c) \wedge \neg E(c)$	par *conjonction* de 6 et 7.
9.	$\exists x(I(x) \wedge \neg E(x)$	par *généralisation existentielle* de 8.

Rép. 1.41 (a) On pose :
$H(x)$: « x est un homme »,
$M(x)$: « x est mortel ».

1.	$\forall x, H(x) \rightarrow M(x)$	
2.	$H(\text{Socrate})$	
3.	$H(\text{Socrate}) \rightarrow M(\text{Socrate})$	par *instanciation universelle* de 1.
4.	$M(\text{Socrate})$	par *modus ponens* de 2 et 3.

Le raisonnement est donc **valide**.

(b) On pose, $B(x)$: « x est bleu ».

1.	$\forall x, H(x) \rightarrow B(x)$	
2.	$H(\text{Socrate})$	
3.	$H(\text{Socrate}) \rightarrow M(\text{Socrate})$	par *instanciation universelle* de 1.
4.	$B(\text{Socrate})$	par *modus ponens* de 2 et 3.

Le raisonnement est donc **valide**.

(c) 1. $\exists x, H(x) \wedge B(x)$
2. $H(\text{Socrate})$
3. $B(\text{Socrate})$

Ce raisonnement est **invalide**. En effet, supposons que Socrate est un homme, que Socrate n'est pas bleu, mais qu'il existe un autre homme qui n'est pas Socrate et que cet autre homme est bleu. On a alors :

$$\begin{aligned}
\exists x, H(x) \wedge B(x) &\equiv \mathbf{V}, \\
H(\text{Socrate}) &\equiv \mathbf{V}, \\
B(\text{Socrate}) &\equiv \mathbf{F}, \\
((\exists x, H(x) \wedge B(x)) \wedge H(\text{Socrate})) \rightarrow B(\text{Socrate}) &\equiv (\mathbf{V} \wedge \mathbf{V}) \rightarrow \mathbf{F} \\
&\equiv \mathbf{F}.
\end{aligned}$$

Remarque, on a traduit la phrase « *certains hommes sont bleus* » par $\exists x, \mathsf{H}(x) \wedge B(x)$ qui signifie plutôt « *il existe un homme bleu* ». Pour être plus rigoureux, on aurait pu remarquer que la phrase « *certains hommes sont bleus* » est équivalente à la phrase « *il y a au moins deux hommes bleus* » ce qui peut être traduit par :

$$\exists x, \exists y, x \neq y \wedge (H(x) \wedge B(x)) \wedge (H(y) \wedge B(y)),$$

Cela dit, le raisonnement reste **invalide** et pour la même raison.

(d) 1. $\exists x, H(x) \wedge M(x)$
2. $H(\text{Socrate})$
3. $M(\text{Socrate})$

Ce raisonnement est **invalide**. En effet, supposons que Socrate est un homme, que Socrate n'est pas mortel, mais qu'il existe un autre homme qui n'est pas Socrate et que cet autre homme est mortel. On a alors :

$$\begin{aligned}
\exists x, H(x) \wedge M(x) &\equiv \mathbf{V}, \\
H(\text{Socrate}) &\equiv \mathbf{V}, \\
M(\text{Socrate}) &\equiv \mathbf{F}, \\
((\exists x, H(x) \wedge M(x)) \wedge H(\text{Socrate})) \rightarrow M(\text{Socrate}) &\equiv (\mathbf{V} \wedge \mathbf{V}) \rightarrow \mathbf{F} \\
&\equiv \mathbf{F}.
\end{aligned}$$

Rép. 1.42 (a) Le raisonnement B est invalide. On montre que l'implication :

$$\Big((a \rightarrow b) \wedge \neg a\Big) \rightarrow \neg b$$

n'est pas une tautologie. En effet, avec l'affectation $a \equiv \mathbf{F}$ et $b \equiv \mathbf{V}$:

$$\begin{aligned}
\Big((a \rightarrow b) \wedge \neg a\Big) \rightarrow \neg b &\equiv \Big((\mathbf{F} \rightarrow \mathbf{V}) \wedge \neg \mathbf{F}\Big) \rightarrow \neg \mathbf{V} \\
&\equiv (\mathbf{V} \wedge \mathbf{V}) \rightarrow \mathbf{F} \\
&\equiv \mathbf{F}
\end{aligned}$$

(b) Le raisonnement A est valide :

1.	$a \rightarrow (b \wedge c)$	
2.	$\neg c$	
3.	$(a \rightarrow b) \wedge (a \rightarrow c)$	Table 2.6 et ligne 1
4.	$a \rightarrow c$	Simplification de 3.
5.	$\neg a$	Modus tollens de 2 et 4.

Le raisonnement C est valide :

1.	$(a \vee b) \wedge c$	
2.	$d \rightarrow \neg c$	
3.	$d \vee \neg a$	
4.	c	Simplification de 1.
5.	$\neg d$	Modus tollens de 5 et 2.
6.	$\neg a$	Syllogisme disjonctif de 3 et 5.
7.	$a \vee b$	Simplification de 1.
8.	b	Syllogisme disjonctif de 7 et 6.

Rép. 1.43 L'ensemble de spécifications est cohérent.
Démonstration. Soit f : La communication est fluide, c : le logiciel Communik est installé, s : la version 10 ou plus du système d'exploitation est installée. Alors :

p1 : $c \rightarrow f$, ce qui est équivalent à $\neg c \vee f$

p2 : $c \rightarrow s$, ce qui est équivalent à $\neg c \vee s$

p3 : $\neg s$

p4 : f

L'assignation $c = \mathbf{F}$, $f = \mathbf{V}$, $s = \mathbf{F}$ permet de satisfaire chacune des spécifications.

	proposition	justification	affectation	prop. satisfaite
1.	$c \rightarrow f$			
2.	$c \rightarrow s$			
3.	$\neg s$		$s = \mathbf{F}$	$p3$
4.	f		$f = \mathbf{V}$	$p4$
5.	$\neg c$	par *modus tollens* de 2 et 3.	$c = \mathbf{F}$	$p2$ (et $p1$)

Rép. 1.44 L'ensemble de spécifications est incohérent.

	proposition	justification	affectation	prop. satisfaite
1.	$f \rightarrow c$			
2.	$c \rightarrow s$			
3.	$\neg s$		$s = \mathbf{F}$	$p3$
4.	f		$f = \mathbf{V}$	$p4$
5.	$\neg c$	par *modus tollens* de 2 et 3.	$c = \mathbf{F}$	$p2$
6.	$\neg f$	par *modus tollens* de 1 et 5.	$f = \mathbf{F}$	contradiction avec $p4$

Rép. 1.45 L'ensemble de spécifications est cohérent.
Démonstration. Soit f : La communication est fluide, c : le logiciel Communik est installé, s : la version 10 ou plus du système d'exploitation est installée. Les propositions 1 à 4 doivent être vraies.

	proposition	justification	affectation	prop. satisfaite
1.	$f \rightarrow c$			
2.	$c \rightarrow s$			
3.	$s \rightarrow f$			
4.	$\neg s$		$s = \mathbf{F}$	$p4$ (et $p3$)
5.	$\neg c$	par *modus tollens* de 2 et 4.	$c = \mathbf{F}$	$p2$
6.	$\neg f$	par *modus tollens* de 1 et 5.	$f = \mathbf{F}$	$p1$

Ainsi, l'assignation $s = \mathbf{F}$, $c = \mathbf{F}$ et $f = \mathbf{F}$ satisfait chacune des spécifications.

Rép. 1.46 L'ensemble de spécifications est cohérent.

Rép. 1.47 Il y a 2 assignations qui permettent de satisfaire chacune des spécifications :
$a = b = \mathbf{V}$ et $c = d = \mathbf{F}$ ou encore $a = b = c = \mathbf{F}$ et $d = \mathbf{V}$.

Rép. 1.48 L'ensemble de spécifications est incohérent.

Rép. 1.49 L'assignation $p = \mathbf{V}, q = \mathbf{F}, r = \mathbf{V}, s = \mathbf{F}, t = \mathbf{V}$ permet de satisfaire chacune des spécifications.

	proposition	justification	affectation	prop. satisfaite
1.	$r \wedge (\neg p \rightarrow q)$			
2.	$\neg q \vee s$			
3.	$t \rightarrow q \vee \neg s$			
4.	$\neg(\neg q \wedge s) \rightarrow t$			
5.	$s \rightarrow \neg r$			
6.	r	par *Simplification* de 1.	$r = \mathbf{V}$	
7.	$\neg s$	par *Modus Tollens* de 6 et 5.	$s = \mathbf{F}$	5.
8.	$\neg p \rightarrow q$	par *Simplification* de 1.		
9.	$\neg q$	par *Syllogisme disjonctif* de 2 et 7.	$q = \mathbf{F}$	2. et 3.
10.	p	par *Modus Tollens* de 8 et 9, et *Double négation*.	$p = \mathbf{V}$	1.
11.	$\neg s \vee q$	par *Addition* sur 7.		
12.	$(q \vee \neg s) \rightarrow t$	par *De Morgan* sur 4.		
13.	t	par *Modus Ponens* de 11 et 12.	$t = \mathbf{V}$	4.

Rép. 1.50 Geneviève est coupable. En effet, les affirmations de Xavier et de Stéphane sont incompatibles : seule l'une d'elles peut-être vraie. Si c'est Xavier qui dit vrai, alors Stéphane est coupable et dans ce cas Geneviève dit vrai aussi (mais c'est impossible, car on sait qu'un seul ami dit vrai). On peut en déduire que Xavier ment. Ainsi, Stéphane n'est pas coupable. Par conséquent, Stéphane dit vrai et on peut en conclure que les autres amis mentent. Puisque Geneviève se dit non coupable est qu'elle ment, elle est coupable !

Voici une autre façon de rédiger la solution. Posons x : Xavier est coupable, g : Geneviève est coupable, etc. Les quatre déclarations des amis sont donc :

$$p1 : x \qquad p2 : \neg g \qquad p3 : s \qquad p4 : \neg s.$$

Nous savons qu'une seule des propositions 1 à 4 est vraie. Analysons les quatre possibilités (il s'agit d'une preuve par cas). Si $p1$ est vraie, alors on doit avoir

$$x \wedge g \wedge s \wedge \neg s,$$

ce qui est impossible, car $\neg s \wedge s \equiv \mathbf{F}$. Si $p2$ est vraie, on doit avoir

$$\neg x \wedge \neg g \wedge \neg s \wedge s,$$

ce qui, encore une fois, est impossible. Si $p3$ est vraie, on doit avoir

$$\neg x \wedge g \wedge s \wedge s,$$

ce qui est impossible, car il n'y a qu'un seul coupable donc $g \wedge s = \mathbf{F}$. Si $p4$ est vraie, on doit avoir

$$\neg x \wedge g \wedge \neg s \wedge \neg s,$$

ce qui est possible : aucune incohérence et un seul coupable. Geneviève est donc coupable.

Rép. 1.51 Xavier est coupable.

Rép. 1.52 La discussion a lieu un mercredi. Anouk est sincère quand elle dit « hier (mardi) j'ai menti » et Geneviève ment quand elle dit « moi aussi » : elle ne peut avoir menti la veille, car elle est sincère le mardi. (Un tableau est suggéré pour y voir plus clair.)

Rép. 1.53
(a) A est sincère et B est menteur.
(b) A est menteur et B est sincère.
(c) A et B sont sincères.
(d) On ne sait pas.
(e) A est menteur et B est sincère.

Rép. 1.54 La question « Si je demande à l'autre gardien quelle porte mène au cours de Mathématiques discrètes, laquelle me pointera-t-il ? » conduira chaque gardien à pointer la porte de l'enfer et il suffira d'ouvrir l'autre porte. En effet, soit A est sincère et il pointera alors la porte que son collègue menteur indiquerait (donc la mauvaise), soit A est menteur et alors B est sincère : B indiquerait la bonne porte, mais A va mentir et indiquera la mauvaise porte.

Rép. 1.55
(a) $\forall x \in \mathbb{Z}, P(x^2) \to P(x) \equiv \forall x \in \mathbb{Z}, \neg P(x) \to \neg P(x^2)$.
Vrai, preuve par contraposée. Il faut donc fournir une preuve. Voir celle présentée à l'exemple de la page 50.

(b) $\forall x \in \mathbb{R}, \forall y \in \mathbb{R}, Q(x) \wedge \neg Q(y) \to \neg Q(x+y)$.
Vrai. Il faut donc fournir une preuve. Voir celle présentée en classe.

(c) $p \equiv \forall x \in \mathbb{R}, \forall y \in \mathbb{R}, Q(x) \wedge x \neq 0 \wedge \neg Q(y) \to \neg Q(xy)$.
Vrai. Preuve par contradiction. Supposons que l'énoncé est faux. Par De Morgan et 2.5, on obtient : $\neg p \equiv \exists x \in \mathbb{R}, \exists y \in \mathbb{R}, \big(Q(x) \wedge x \neq 0 \wedge \neg Q(y)\big) \wedge Q(xy)$.
Ainsi, il existe un nombre $x \in \mathbb{Q}$, avec $x \neq 0$, et un nombre $y \notin \mathbb{Q}$ tels que leur produit est rationnel : $xy \in \mathbb{Q}$. Ainsi, il existe des entiers $a \neq 0$, $b \neq 0$, c et $d \neq 0$ tels que $x = \frac{a}{b}$ et $xy = \frac{c}{d}$. Donc $y = (xy) \cdot \frac{1}{x} = \frac{c}{d}\frac{b}{a} = \frac{bc}{ad}$, et on conclut que y est rationnel, ce qui contredit $y \notin \mathbb{Q}$.
On a montré que $\neg p \to \mathbf{F}$, ce qui permet de conclure que $\neg p$ est faux, et donc que p est vrai.

(d) $\neg Q\left(\frac{\sqrt{2}}{3}\right)$. Vrai, c'est un cas particulier de (c) : $\frac{\sqrt{2}}{3} = \frac{1}{3} \cdot \sqrt{2}$, $\frac{1}{3} \in \mathbb{Q}$ et $\sqrt{2} \notin \mathbb{Q}$.

(e) $\neg Q\left(\frac{1}{\sqrt{2}}\right)$. Vrai. Preuve par contradiction. Supposons que l'énoncé est faux. Ainsi, $x = \frac{1}{\sqrt{2}} \in \mathbb{Q}$. Alors il existe des entiers a et b tels que $x = \frac{a}{b}$ et $b \neq 0$. De plus, $a \neq 0$ car $x \neq 0$. Ainsi, $\frac{1}{x} = 1 \div \frac{a}{b} = \frac{b}{a} \in \mathbb{Q}$. Or $\frac{1}{x} = 1 \div \frac{1}{\sqrt{2}} = \sqrt{2} \notin \mathbb{Q}$. Contradiction. On conclut donc que $x = \frac{1}{\sqrt{2}} \notin \mathbb{Q}$.

(f) $p \equiv \forall x \in \mathbb{R}, \neg Q(x) \to \neg Q\left(\frac{1}{x}\right)$.
Vrai. Il faut donc fournir une preuve. Puisque $\neg Q(x) \equiv x \notin \mathbb{Q}$ et $Q(x) \equiv x \in \mathbb{Q}$ on a, par contraposée : $p \equiv \forall x \in \mathbb{R}, \frac{1}{x} \in \mathbb{Q} \to x \in \mathbb{Q}$.
Prenons un nombre réel x quelconque et supposons que $\frac{1}{x} \in \mathbb{Q}$, c'est-à-dire que $\frac{1}{x}$ peut s'écrire comme une fraction $\frac{a}{b}$, avec $a \in \mathbb{Z}$ et $b \in \mathbb{Z}$ et $b \neq 0$. Il suffit d'inverser la fraction pour obtenir x : $x = \frac{b}{a}$. On sait que $a \neq 0$, car $\frac{a}{b} = \frac{1}{x}$. Ainsi, x est rationnel. L'implication est démontrée.

(g) $p \equiv \forall x \in \mathbb{R}, \forall y \in \mathbb{R}, \neg Q(x) \wedge \neg Q(y) \to \neg Q(x+y)$.
Faux. Montrons que $\neg p$ est vrai. Par De Morgan et l'équivalence 2.5, on a :
$\neg p \equiv \exists x \in \mathbb{R}, \exists y \in \mathbb{R}, \big(\neg Q(x) \wedge \neg Q(y)\big) \wedge Q(x+y)$.
Prenons $x = \sqrt{2} \notin \mathbb{Q}$ et $y = 1 - \sqrt{2}$. On sait que $y \notin \mathbb{Q}$, car $(-1) \cdot \sqrt{2} \notin \mathbb{Q}$ par (c) et donc $y = 1 + (-1) \cdot \sqrt{2} \notin \mathbb{Q}$ par (b). Or $x + y = 1 \in \mathbb{Q}$. Nous avons ainsi trouvé un contre-exemple à l'énoncé universel : il est donc faux.

(h) $p \equiv \forall x \in \mathbb{R}, \forall y \in \mathbb{R}, \neg Q(x) \wedge \neg Q(y) \to \neg Q(xy)$.
Prenons $x = y = \sqrt{2} \notin \mathbb{Q}$. On a $xy = 2 \in \mathbb{Q}$. Nous avons ainsi trouvé un contre-exemple à l'énoncé universel : il est donc faux.
Ou encore, prenons $x = \sqrt{2} \notin \mathbb{Q}$ et $y = \frac{1}{\sqrt{2}}$. On sait que $y \notin \mathbb{Q}$, tel que vu en (e). Or $xy = 1 \in \mathbb{Q}$. Nous avons ainsi trouvé un contre-exemple à l'énoncé universel : il est donc faux.

(i) $p \equiv \forall x \in \mathbb{R}, Q(x) \wedge x \neq 0 \to Q\left(\frac{1}{x}\right) \equiv \forall x \in \mathbb{R}, x \in \mathbb{Q} \wedge x \neq 0 \to \frac{1}{x} \in \mathbb{Q}$.
Vrai. Il faut donc fournir une preuve. Si x est rationnel et non nul, il suffit d'inverser la fraction pour obtenir $\frac{1}{x}$, donc $\frac{1}{x} \in \mathbb{Q}$.

(j) $p \equiv \forall x \in \mathbb{R}, \neg Q(x) \rightarrow \neg Q(\frac{1}{x})$.
Vrai. Il s'agit d'une formulation équivalente à (f).

Rép. 1.56 (a) Faux (b) Vrai (c) Vrai (d) Faux (e) Vrai (f) Vrai (g) Vrai (h) Faux (i) Vrai (j) Vrai

Rép. 1.57 (a) Vrai (b) Faux (c) Faux (d) Faux (e) Faux (f) Vrai (g) Faux en général, mais vrai dans le cas où $x = \emptyset$.

Rép. 1.58 (a) $\{1,2,3,4,7\}$
(b) $\{2\}$
(c) $\{2,3\}$
(d) $\{7\}$
(e) $\{\emptyset,\{1\},\{2\},\{1,2\}\}$
(f) $\emptyset$
(g) $\{1,8\}$
(h) $\emptyset$
(i) $\{5,6\}$
(j) $\{6,7,8\}$
(k) $\{2,3,5,6\}$
(l) $\{(6,1),(6,2),(7,1),(7,2),(8,1),(8,2)\}$

Rép. 1.59 (a) $A = \{r \in R, \mid M(\text{Alice}, r)\} = \{J\}$; ensemble de cardinalité 1.
(b) $B = \{x \in X, \mid M(x,P)\} = \{\text{Guy, Julien, Manon, Ugo}\}$; ensemble de cardinalité 4.
(c) $C = \{x \in X, \mid L(x,K)\} = \{\text{Bartez, Guy}\}$; ensemble de cardinalité 2.
(d) $D = \{r \in R, \mid \forall x \in X, L(x,r)\} = \{P,Z\}$; ensemble de cardinalité 2.
(e) $E = \{r \in R, \mid \exists x \in X, \neg L(x,r)\} = \{K,J,S\}$; ensemble de cardinalité 3.
(f) $D \cup E = \{K,J,P,S,Z\}$; ensemble de cardinalité 5.
(g) $D \cap E = \{\}$; ensemble de cardinalité 0.
(h) $B \cap C = \{\text{Guy}\}$; ensemble de cardinalité 1.
(i) $X - B = \{\text{Alice, Bartez}\}$; ensemble de cardinalité 2.
(j) $\overline{E} = \{P,Z\}$; ensemble de cardinalité 2.

Rép. 1.60 (a) $L_1 \cup L_2 = \{p_1,p_2,p_3,p_4,p_5,p_6\}$; de cardinalité 6.
(b) $L_2 \cup L_4 = \{p_1,p_2,p_3,p_6\}$; de cardinalité 4.
(c) $\big(L_1 \cup L_2\big) \cap L_4 = \{p_1,p_2,p_6\}$; de cardinalité 3.
(d) $L_2 - L_4 = \{p_3\}$; de cardinalité 1.
(e) $\overline{L_1 \cup L_2} = \{p_7\}$; de cardinalité 1.
(f) $\bigcup_{i=2}^{4} L_i = \{p_1,p_2,p_3,p_4,p_6,p_7\}$; de cardinalité 6.
(g) $\bigcap_{i=3}^{5} L_i = \{\}$; de cardinalité 0.

Rép. 1.61 (a) Vrai
(b) Vrai
(c) Faux
(d) Faux
(e) Vrai
(f) Vrai
(g) Faux
(h) Vrai
(i) Vrai
(j) Vrai

Rép. 1.62 (a) 1. Faux.
(b) 8. Faux.
(c) 14. Faux.
(d) 13. Faux.
(e) 10. Vrai.
(f) 9. Vrai.
(g) 12. Vrai.

Rép. 1.63 (a)
$$\begin{aligned} \overline{A \cap B \cap \overline{C}} \cup C &= \overline{A} \cup \overline{B} \cup \overline{\overline{C}} \cup C && \text{De Morgan} \\ &= \overline{A} \cup \overline{B} \cup U && \text{Négation} \\ &= U && \text{Domination} \end{aligned}$$

(b) $A \cap (H \cup D \cup A) = A$ Absorption

(c)
$$\begin{aligned} \overline{(A \cap B) \cup \overline{B}} &= \overline{(A \cap B)} \cap \overline{\overline{B}} && \text{De Morgan} \\ &= (\overline{A} \cup \overline{B}) \cap B && \text{De Morgan et double négation} \\ &= (\overline{A} \cap B) \cup (\overline{B} \cap B) && \text{Distributivité} \\ &= (\overline{A} \cap B) \cup \emptyset && \text{Négation} \\ &= \overline{A} \cap B && \text{Identité} \end{aligned}$$

(d)
$$\begin{aligned} (\overline{A \cup B}) \cup \overline{B} &= (\overline{A} \cap \overline{B}) \cup \overline{B} && \text{De Morgan} \\ &= \overline{B} && \text{Absoption} \end{aligned}$$

(e)
$$\begin{aligned} \overline{(A \cup \bar{B}) \cup (C \cup \bar{A})} &= \overline{(A \cup \overline{\overline{B}})} \cap \overline{(C \cup \overline{\overline{A}})} && \text{De Morgan} \\ &= (\overline{A} \cap \overline{\overline{B}}) \cap (\overline{C} \cap \overline{\overline{A}}) && \text{De Morgan} \\ &= (\overline{A} \cap B) \cap (\overline{C} \cap A) && \text{Double négation} \\ &= A \cap \overline{A} \cap B \cap \overline{C} && \text{Associativité et commutativité} \\ &= \emptyset \cap B \cap \overline{C} && \text{Négation} \\ &= \emptyset && \text{Domination} \end{aligned}$$

Rép. 1.64 (a) $(x \geq 2) \wedge (x \leq 4)$
(b) $(x \geq 2)$
(c) $(x \geq 2) \wedge (x \leq 4) \vee (x \geq 7) \wedge (x \leq 9)$
(d) $(x \leq 0) \vee (x \geq 10) \wedge (x \leq 15)$
(e) $(x \geq 2) \wedge (x \leq 8) \wedge \neg(x = 4)$

Chapitre 2

Rép. 2.1 (a) Ce n'est pas une solution admissible, car elle ne respecte pas la contrainte de coupe.
(b) Ce n'est pas une solution admissible, car elle ne respecte pas la contrainte d'assemblage (ni celle de finition).
(c) C'est une solution admissible, car elle respecte toutes les contraintes, mais elle n'est pas optimale, car la contribution marginale engendrée est de 34 500$, ce qui est moins que la contribution marginale maximale obtenue à l'aide du solveur (42 000$).
(d) C'est une solution admissible, car elle respecte toutes les contraintes, et optimale, car la contribution marginale engendrée est de 42 000$, ce qui est égal la contribution marginale maximale obtenue à l'aide du solveur.
Remarquez qu'il ne s'agit pas de la même solution optimale que celle fournie par le solveur : il peut y avoir plusieurs solutions qui respectent les contraintes et engendrent la contribution marginale maximale.
(e) Ce n'est pas une solution admissible, car elle ne respecte pas les contraintes naturelles : les valeurs ne sont pas toutes entières.

Rép. 2.2 **But :** maximiser la contribution marginale totale en \$, que l'on note z.

Variables : $\forall i \in \{1,2,3\}$, x_i = nombre d'unités à fabriquer du produit i.

Contraintes naturelles : $\forall i \in \{1,2,3\}$, $x_i \in \mathbb{N}$.

Fonction-objectif : $z = 120x_1 + 150x_2 + 90x_3$.

Contraintes :

$$\begin{array}{rrcrcrcr}
\text{Atelier 1 :} & 2x_1 & + & 1x_2 & + & 1x_3 & \leq & 400 \\
\text{Atelier 2 :} & 4x_1 & + & 5x_2 & + & 3x_3 & \leq & 1400 \\
\text{Atelier 3 :} & 1x_1 & + & 2x_2 & + & 1x_3 & \leq & 500 \\
\text{Contrainte supp. 1 :} & 0x_1 & + & 0x_2 & + & 1x_3 & \geq & 150 \\
\text{Contrainte supp. 2 :} & 1x_1 & + & 0x_2 & + & 1x_3 & \geq & 250 \\
\text{Contrainte supp. 2 :} & 1x_1 & + & -1x_2 & + & 0x_3 & \geq & 0
\end{array}$$

Notez que la dernière contrainte est équivalente à $x_1 \geq x_2$.

Une solution qui maximise la contribution marginale est de produire 75 unités du produit 1, 75 unités du produit 2 et 175 unités du produit 3. La contribution marginale est alors de 36 000\$.

Rép. 2.3 Modèle (les éléments qui diffèrent de l'exemple de la Section 2.2 sont indiqués par le symbole $\star$).

But : minimiser le nombre de points de services déployés, noté z.

Ensemble : $Q = \{1,2,3,4,5,6,7,8\}$ ensemble des quartiers de la ville.

Variables : s_q, où $q \in Q$. On déploie ou non un point de service dans le quartier q.

Contraintes naturelles : $\forall q \in Q$, $s_q \in \{0,1\}$.

Paramètres : Relation de proximité entre les quartiers décrite par la matrice A. La matrice demeure la même que celle de l'exemple.

Fonction-objectif : $z = \sum_{q \in Q} s_q$

$\star$Contraintes$\star$: $\forall i \in Q$, $\sum_{q \in Q} \left(x_{iq} \cdot s_q\right) \geq 3$

Une solution qui minimise le nombre de points de service est d'en déployer 5, soit dans les quartiers 2, 3, 4, 5 et 8.

Rép. 2.4 Modèle (les éléments qui diffèrent de l'exemple de la Section 2.2 sont indiqués par le symbole $\star$).

But : minimiser le nombre de points de services déployés, noté z.

Ensemble : $Q = \{1,2,3,4,5,6,7,8\}$ ensemble des quartiers de la ville.

Variables : s_q, où $q \in Q$. On déploie ou non un point de service dans le quartier q.

Contraintes naturelles : $\forall q \in Q$, $s_q \in \{0,1\}$.

$\star$Paramètres$\star$: Relation de proximité entre les quartiers décrite par la matrice A. La matrice A doit être modifiée pour indiquer que le quartier 5 est en quarantaine. On modifie donc la cinquième ligne et la cinquième colonne : des 0 partout, sauf en position $(5,5)$.

Fonction-objectif : $z = \sum_{q \in Q} s_q$

Contraintes : $\forall i \in Q$, $\sum_{q \in Q} \left(x_{iq} \cdot s_q\right) \geq 1$

Une solution qui minimise le nombre de points de service est d'en implanter 3, soit dans les quartiers 4, 5 et 7.

Rép. 2.5 Modèle (les éléments qui diffèrent de l'exemple de la Section 2.2 sont indiqués par le symbole $\star$).

But : minimiser le nombre de points de services déployés, noté z.

Ensemble : $Q = \{1,2,3,4,5,6,7,8\}$ ensemble des quartiers de la ville.

Variables : s_q, où $q \in Q$. On déploie ou non un point de service dans le quartier q.

Contraintes naturelles: $\forall q \in Q,\ s_q \in \{0,1\}$.

⋆Paramètres⋆:

- Relation de proximité entre les quartiers décrite par la matrice A. La matrice demeure la même que celle de l'exemple.
- Il faut ajouter un paramètre pour tenir compte du besoin minimal pour chaque quartier.
  $\forall q \in Q$, b_q : besoin minimal du quartier q (en nombre de points de service).
  $[b_1, b_2, \ldots, b_8] = [1,1,1,3,3,3,1,1]$.

Fonction-objectif: $z = \sum_{q \in Q} s_q$

⋆Contraintes⋆: $\forall i \in Q,\ \sum_{q \in Q} \left(x_{iq} \cdot s_q\right) \geq b_i$

Une solution qui minimise le nombre de points de service est d'en implanter 3, soit dans les quartiers 3, 6 et 8.

Rép. 2.6 Modèle (les éléments qui diffèrent de l'exemple de la Section 2.2 sont indiqués par le symbole ⋆).

⋆But⋆: : minimiser le coût de location, noté z

Ensemble: $Q = \{1,2,3,4,5,6,7,8\}$ ensemble des quartiers de la ville.

Variables: s_q, où $q \in Q$. On déploie ou non un point de service dans le quartier q.

Contraintes naturelles: $\forall q \in Q,\ s_q \in \{0,1\}$.

⋆Paramètres⋆:

- Relation de proximité entre les quartiers décrite par la matrice A. La matrice demeure la même que celle de l'exemple.
- *Il faut ajouter un paramètre pour tenir compte du coût de location relatif dans chaque quartier.*
  $\forall q \in Q$, c_q : coût de location relatif dans le quartier q (c_q n'est pas le coût réel, mais plutôt un facteur multiplicatif du coût de location minimal).
  $[c_1,\ c_2,\ \ldots,\ c_8] = [1,\ 1.6,\ 1.6,\ 1.6,\ 1.6,\ 1,\ 1,\ 1.6]$.

⋆Fonction-objectif⋆: $z = \sum_{q \in Q} \left(s_q \cdot c_q\right)$

Contraintes: $\forall i \in Q,\ \sum_{q \in Q} \left(x_{iq} \cdot s_q\right) \geq 1$

Une solution qui minimise le coût de location relatif total est de déployer 2 points de services, soit dans les quartiers 1 et 8. Le coût de location total sera alors de 2.6 fois le coût de location du quartier le moins cher.

Rép. 2.7 Modèle (les éléments qui diffèrent de l'exemple 2.2 sont indiqués par le symbole ⋆).

⋆But⋆: maximiser le nombre de franchises, noté z.

Ensemble: $Q = \{1,2,3,4,5,6,7,8\}$ ensemble des quartiers de la ville.

Variables: s_q, où $q \in Q$. On déploie ou non une franchise dans le quartier q.

⋆Contraintes naturelles⋆: $\forall q \in Q,\ s_q \in \mathbb{N}$.

Paramètres: Relation de proximité entre les quartiers décrite par la matrice A. La matrice demeure la même que celle de l'exemple.

Fonction-objectif: $z = \sum_{q \in Q} s_q$

⋆Contraintes⋆: *les citoyens de chaque quartier auront accès à* ***au plus 2*** *franchises situées soit dans leur propre quartier, soit dans un quartier adjacent.*

$$\forall i \in Q,\ \sum_{q \in Q} \left(x_{iq} \cdot s_q\right) \leq 2$$

On peut déployer un maximum de 3 franchises, par exemple en choisissant les quartiers 1, 2, et 5.

Rép. 2.8 Modèle (les éléments qui diffèrent de l'exemple de la Section 2.2 sont indiqués par le symbole ⋆).

⋆**But**⋆ **: maximiser** le nombre de franchises, noté z

Ensemble; $Q = \{1,2,3,4,5,6,7,8\}$ ensemble des quartiers de la ville.

Variables: s_q, où $q \in Q$. On déploie ou non une franchise dans le quartier q.

⋆**Contraintes naturelles**⋆**:** $\forall q \in Q,\ s_q \in \mathbb{N}$.

Paramètres: Relation de proximité entre les quartiers décrite par la matrice A. La matrice demeure la même que celle de l'exemple.

Fonction-objectif: $z = \sum_{q \in Q} s_q$

⋆**Contraintes**⋆ : $\forall i \in Q,\ \sum_{q \in Q} \left(x_{iq} \cdot s_q\right) \leq 3$

On peut déployer un maximum de 4 franchises, par exemple avec une franchise dans chacun des quartiers 2 et 3, et avec 2 franchises dans le quartier 7.

Rép. 2.9 **But** : minimiser le salaire total annuel, qui est noté z.

Ensemble : $P = \{1,2,3,4,5,6\}$ ensemble des postulants.

Variables :

$$\forall p \in P,\ x_p = \begin{cases} 1 & \text{si le postulant } p \text{ est embauché} \\ 0 & \text{sinon} \end{cases}$$

Contraintes naturelles : $\forall p \in P,\ x_p \in \{0,1\}$.

Paramètres : $\forall p \in P$, c_p : salaire annuel du postulant p. $[c_1, \ldots, c_6] = [65, 55, 58, 74, 55, 74, 70, 81]$.

Fonction-objectif : $z = \sum_{p \in P} \left(c_p \cdot x_p\right)$

Contraintes :

1. Il faut embaucher exactement quatre personnes.

   $$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 = 4$$

   Ou encore, en utilisant le symbole de sommation :

   $$\sum_{p \in P} x_p = 4$$

2. Au moins un des postulants numéro 3, 5 ou 8 doit être embauché, car eux seuls ont des connaissances en électricité.

   $$x_3 + x_5 + x_8 \geq 1$$

   Ou encore, en utilisant le symbole de sommation :

   $$\sum_{p \in \{3,5,8\}} x_p \geq 1$$

3. Si le postulant 2 est embauché ou le postulant 4 est embauché, alors le postulant 5 ne doit pas l'être. *Plus difficile à traduire. Les règles d'équivalence logique sont utiles, en particulier la 2.1 et la distributivité!*

   $$(x_2 + x_5 \leq 1) \quad \wedge \quad (x_4 + x_5 \leq 1)$$

4. Si les postulants 1 et 3 sont embauchés tous les deux, alors le postulant 6 ne doit pas l'être. *Plus difficile à traduire. Les règles d'équivalence logique sont utiles.*

   $$x_1 + x_3 + x_6 \leq 2$$

   Ou encore, en utilisant le symbole de sommation :

   $$\sum_{p \in \{1,3,6\}} x_p \leq 2$$

5. Les postulants 3 et 5 forment un couple : ils accepteront l'affectation seulement si les deux sont embauchés.

$$x_3 - x_5 = 0$$

En utilisant le solveur, on obtient une solution qui minimise le salaire total tout en respectant l'ensemble des contraintes. Il s'agit d'embaucher les postulants 1, 3, 5 et 8. Le salaire annuel total sera alors de 259 000$. La figure 2.1 présente une feuille de calcul de ce modèle.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**BUT: minimiser la fonction-objectif, soit le total des salaires annuels**												
2													
3	**Fonction-objectif** : $z = \Sigma\, c_p\, x_p$												
4	**Ensemble** des postulants P	1	2	3	4	5	6	7	8				
5	**Noms des variables :** x_p	x1	x2	x3	x4	x5	x6	x7	x8				
6	**Définition** x_p = 1 si le postulant p est embauché, et 0 sinon.												
7										z=			
8	**Coefficients** c_p (en 1000$ par an) et **valeur de** z	65	55	58	74	55	74	70	81	**259**			
9													
10	**Contraintes**									total	signe	requis	Vrai ou faux
11	Condition 1	1	1	1	1	1	1	1	1	4	=	4	VRAI
12	Condition 2	0	0	1	0	1	0	0	1	3	>=	1	VRAI
13	Condition 3a	0	1	0	0	1	0	0	0	1	<=	1	VRAI
14	Condition 3b	0	0	0	1	1	0	0	0	1	<=	1	VRAI
15	Condition 4	1	0	1	0	0	1	0	0	2	<=	2	VRAI
16	Condition 5	0	0	1	0	-1	0	0	0	0	=	0	VRAI
17													
18	**Contraintes naturelles** : x_p = 0 ou 1												
19													
20	**Valeurs des variables** x_p	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**1**				

Figure 2.1 Feuille de calcul du modèle de l'exercice 2.9.

Chapitre 3

Rép. 3.1 Rappel :

- La somme de deux nombres pairs est paire.
- La somme de deux nombres impairs est paire.
- La somme d'un nombre pair et d'un nombre impair est impaire.

Ainsi, la somme d'un nombre impair de nombres impairs est impaire.

Preuve: Par contradiction, on suppose qu'il existe un graphe $G = (V, E)$ à n sommets avec un nombre impair de sommets de degré impair. On pose :

$$S = \sum_{v \in V} \deg(v).$$

Soit V_p l'ensemble des sommets de G de degré pair et V_i l'ensemble des sommets de G de degré impair. On a que $V_p \cup V_i = V$ et $V_p \cap V_i = \emptyset$ et donc,

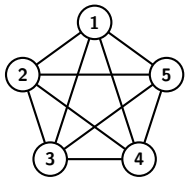
$$S = \sum_{v \in V_p} \deg(v) + \sum_{v \in V_i} \deg(v).$$

D'un côté, $\sum_{v \in V_p} \deg(v)$ est pair car c'est une somme de nombres pairs.

D'un autre côté, $\sum_{v \in V_i} \deg(v)$ est impair car c'est la somme d'un nombre impair de nombres impairs.

Ainsi, S est impair car c'est la somme d'un nombre pair et d'un nombre impair. Or, le Théorème 3.1 stipule que $S = 2|E|$, c'est donc un nombre pair. Contradiction.

Rép. 3.2 (a) Le graphe K_5 possède 10 arêtes.



(b) Numérotons les sommets du graphe K_n de 1 à n. Le sommet 1 est relié par une arête à ses $n-1$ sommets adjacents, le sommet 2 aux $n-2$ sommets adjacents restants, ainsi de suite jusqu'au sommet n. En additionnant toutes ces arêtes on obtient la sommation suivante, dont on trouve la somme avec le théorème 5.7 :

$$(n-1)+(n-2)+(n-3)+\cdots+(n-(n-1))+(n-(n-0)) = 0+1+2+3+\cdots+(n-1) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}.$$

Une autre façon d'obtenir le résultat est d'utiliser le théorème 3.1 des poignées de mains.
Soit $G=(V,E)$ un graphe non orienté. Alors

$$\sum_{v\in V} \deg(v) = 2|E|.$$

Puisque le graphe K_n est complet, chacun des n sommets de l'ensemble V est reliés aux $n-1$ autres. Ainsi, la somme des degrés des sommets est de n fois $(n-1)$. En isolant $|E|$, on trouve que le nombre d'arêtes est de $\frac{n(n-1)}{2}$.

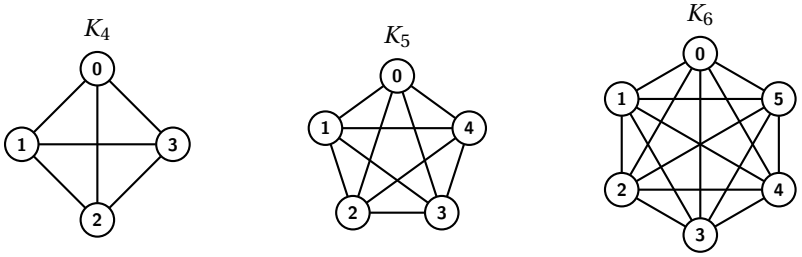
Rép. 3.3 (a) 2 147 randonnées

(b) 785 randonnées

(c) 606 randonnées

(d) 3 heures

Rép. 3.4 (a)



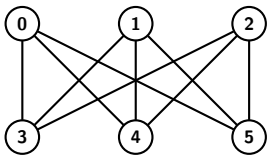
K_4
- circuit eulérien : impossible, les 4 sommets sont de degré impair.
- circuit hamiltonien : $\mathbf{0-1-2-3-0}$.

K_5
- circuit eulérien : $\mathbf{0-1-2-0-3-1-4-2-3-4-0}$;
- circuit hamiltonien : $\mathbf{0-1-2-3-4-0}$.

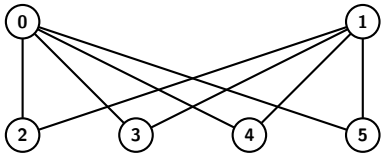
K_6
- circuit eulérien : impossible, les 6 sommets sont de degré impair.
- circuit hamiltonien : $\mathbf{0-1-2-3-4-5-0}$.

(b)

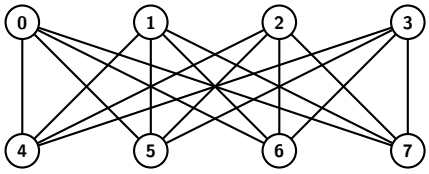
$K_{3,3}$



$K_{2,4}$



$K_{4,4}$



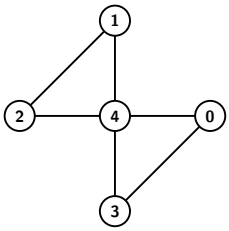
$K_{3,3}$
- circuit eulérien : impossible, les 6 sommets sont de degré impair.
- circuit hamiltonien : $\mathbf{0-3-1-4-2-5-0}$.

$K_{2,4}$
- circuit eulérien : $\mathbf{0-2-1-3-0-4-1-5-0}$;
- circuit hamiltonien : impossible. Les ensembles $B_0 = \{0,1\}$ et $B_1 = \{2,3,4,5\}$ forment une bipartition du graphe. Un chemin dans ce graphe doit forcément visiter en alternance un sommet de B_0 puis un sommet de B_1 puis un sommet de B_0 et ainsi de suite. Hors, comme $|B_0| = 2$ et $|B_1| = 4$ il est impossible qu'un circuit passe un et une seule fois par chacun des sommets.

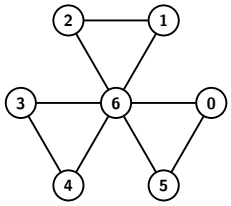
$K_{4,4}$
- circuit eulérien : $\mathbf{0-7-3-6-2-5-3-4-2-7-1-6-0-5-1-4-0}$;
- circuit hamiltonien : $\mathbf{0-4-1-5-2-6-3-7-0}$.

(c)

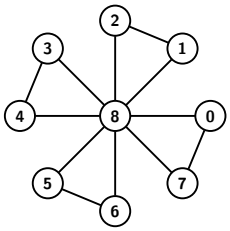
F_2



F_3



F_4

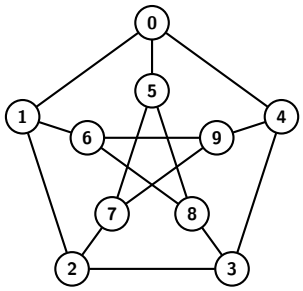


F_2
- circuit eulérien : $\mathbf{0-4-2-1-4-3-0}$;
- circuit hamiltonien : impossible. Tout d'abord, on remarque que lorsqu'un sommet est de degré 2, un circuit hamiltonien passe forcément par ses deux arêtes. Ainsi, si un sommet est adjacent à plus de deux sommets de degré 2 alors on peut affirmer qu'il n'existe pas de circuit hamiltonnien. Ici, le sommet 4 est adjacent à quatre sommets de degré 2.
  Remarque : Bien qu'il n'existe aucun circuit hamiltonien dans ce graphe, il existe des chemins hamiltoniens. Par exemple : $\mathbf{0-3-4-1-2}$.

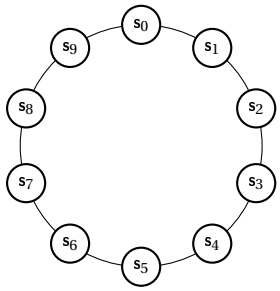
F_3
- circuit eulérien : $\mathbf{0-5-6-4-3-6-2-1-6-0}$
- circuit hamiltonien : impossible. De manière similaire au cas du graphe F_2, on remarque ici que le sommet 6 est adjacent à six sommets de degré 2.

F_4
- circuit eulérien : $\mathbf{0-7-8-6-5-8-4-3-8-2-1-8-0}$.
- circuit hamiltonien : impossible. De manière similaire au cas du graphe F_2, on remarque ici que le sommet 8 est adjacent à huit sommets de degré 2.

(d) Graphe de Peterson



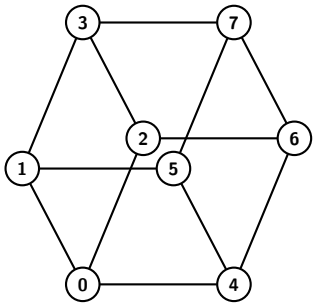
- circuit eulérien : impossible. Les 10 sommets sont de degré impair.
- circuit hamiltonien : impossible. Par contradiction, on suppose qu'il existe un circuit hamiltonien et que ce circuit est $\mathbf{s}_0 - \mathbf{s}_1 - \ldots \mathbf{s}_9 - \mathbf{s}_0$. Maintenant, on trace le graphe de Peterson, en deux étapes. Première étape : on dispose les 10 sommets le long d'un cercle, dans l'ordre donné par le circuit hamiltonien :



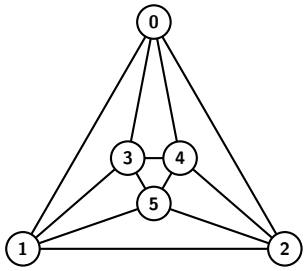
Deuxième étape : tracer les 5 arêtes restantes (le graphe de Peterson compte 15 arêtes). On remarque alors que lorsqu'on ajoute 5 arêtes à un tel graphe, on forme forcément un circuit de taille 3 ou 4 or les plus petits circuits du graphe de Peterson sont de longueur 5. Contradiction.
Remarque : Bien qu'il n'existe aucun circuit hamiltonien dans ce graphe, il existe des chemins hamiltoniens. Par exemple : $\mathbf{0}-\mathbf{1}-\mathbf{6}-\mathbf{8}-\mathbf{5}-\mathbf{7}-\mathbf{9}-\mathbf{4}-\mathbf{3}-\mathbf{2}$.

(e) Solides de Platon. Il existe cinq solides de Platon : le tétraèdre, le cube, l'octaèdre, le dodécaèdre et l'icosaèdre. Le graphe du tétraèdre est K_4, qui a déjà été traité en (a).

Cube



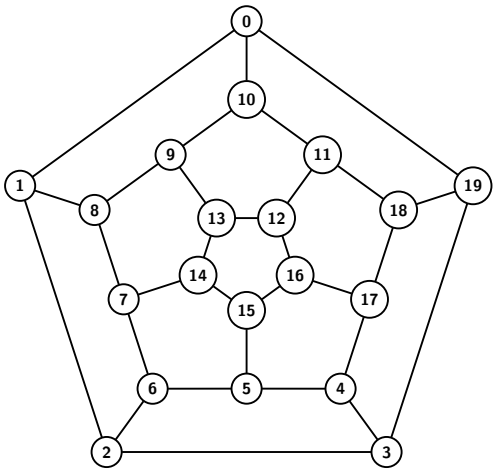
Octaèdre



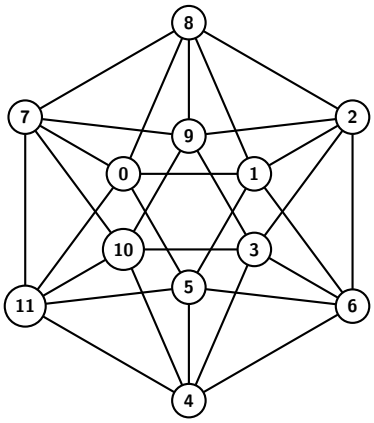
Cube
- circuit eulérien : impossible, tous les sommets sont de degré impair.
- circuit hamiltonien : $\mathbf{0}-\mathbf{1}-\mathbf{3}-\mathbf{2}-\mathbf{6}-\mathbf{7}-\mathbf{5}-\mathbf{4}-\mathbf{0}$.

Octaèdre
- circuit eulérien : $\mathbf{0}-\mathbf{4}-\mathbf{5}-\mathbf{3}-\mathbf{4}-\mathbf{2}-\mathbf{5}-\mathbf{1}-\mathbf{3}-\mathbf{0}-\mathbf{2}-\mathbf{1}-\mathbf{0}$
- circuit hamiltonien : $\mathbf{0}-\mathbf{1}-\mathbf{2}-\mathbf{4}-\mathbf{5}-\mathbf{3}-\mathbf{0}$.

Dodécaèdre



Icosaèdre



Dodécaèdre
- circuit eulérien : impossible, tous les sommets sont de degré impair.
- circuit hamiltonien : $\mathbf{0}-\mathbf{1}-\mathbf{2}-\mathbf{3}-\mathbf{19}-\mathbf{18}-\mathbf{17}-\mathbf{4}-\mathbf{6}-\mathbf{5}-\mathbf{7}-\mathbf{8}-\mathbf{9}-\mathbf{13}-\mathbf{14}-\mathbf{15}-\mathbf{16}-\mathbf{12}-\mathbf{11}-\mathbf{10}-\mathbf{0}$.

Icosaèdre
- circuit eulérien : impossible, tous les sommets sont de degrés impair.
- circuit hamiltonien : $\mathbf{0}-\mathbf{1}-\mathbf{8}-\mathbf{9}-\mathbf{2}-\mathbf{3}-\mathbf{10}-\mathbf{4}-\mathbf{6}-\mathbf{5}-\mathbf{11}-\mathbf{7}-\mathbf{0}$.

Rép. 3.5

u	A	B	C	D	E	F	G	H	S
	0	∞	∞	∞	∞	∞	∞	∞	$\emptyset$
A		$9_{\mathbf{A}}$	$8_{\mathbf{A}}$	$3_{\mathbf{A}}$	∞	∞	∞	∞	{A}
D		$9_{\mathbf{A}}$	$7_{\mathbf{D}}$		∞	∞	$10_{\mathbf{D}}$	∞	{A, D}
C		$8_{\mathbf{C}}$			$13_{\mathbf{C}}$	$10_{\mathbf{C}}$	$10_{\mathbf{D}}$	∞	{A, C, D}
B					$12_{\mathbf{B}}$	$10_{\mathbf{C}}$	$10_{\mathbf{D}}$	∞	{A, B, C, D}
F					$12_{\mathbf{B}}$		$10_{\mathbf{D}}$	$14_{\mathbf{F}}$	{A, B, C, D, F}
G					$12_{\mathbf{B}}$			$14_{\mathbf{F}}$	{A, B, C, D, F, G}
E								$13_{\mathbf{E}}$	{A, B, C, D, E, F, G}
H									{A, B, C, D, E, F, G, H}

(a) Distance minimale de **A** à **H** : 13.
(b) Chemin minimal : **A** – **D** – **C** – **B** – **E** – **H**.
(c) Distance minimale de **A** à **B** : 8.
(d) Chemin minimal : **A** – **D** – **C** – **B**.

Rép. 3.6 (a) La distance minimale est 16 via le chemin **C** – **A** – **B** – **D** – **F** – **G**.
(b) La distance minimale est 10 via le chemin **C** – **E** – **D** – **F** – **H** – **G**.

Rép. 3.7 (a) 6 itinéraires.
(b) 17 itinéraires.
(c) 10 itinéraires.
(d) 9747 itinéraires.
(e) 44 itinéraires.
(f) Oui.
(g) Non. Non.

(h) Oui. Par exemple, le circuit hamiltonien : Gaspé - Québec- Toronto - Vancouver - Ottawa - Montréal - Gaspé.

(i) Non. Le sommet Vancouver est de degré impair, le théorème 3.4 permet donc de conclure que non.

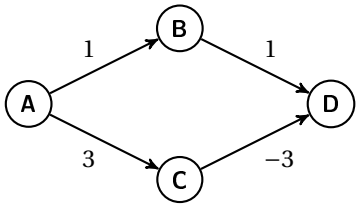
(j) Oui, car le graphe possède 2 sommets de degrés impairs (théorème 3.5). Par exemple, le chemin eulérien : Vancouver - Toronto - Québec - Gaspé - Montréal - Québec- Ottawa - Montréal - Vancouver - Ottawa - Toronto - Montréal.

(k) Graphe pondéré.

(l) Chemin : Gaspé - Q - M - T - Vancouver, pour un coût de 2150 dollars. Chemin T- M-Q - G, pour un coût de 1050 dollars. Non, pas toujours le même nombre, l'algorithme s'arrête quand le sommet de destination a été ajouté à l'ensemble S des sommets visités.

Rép. 3.8

(a) Voici un exemple de graphe pour lequel l'algorithme de Dijkstra ne produit pas un chemin de pondération minimale.

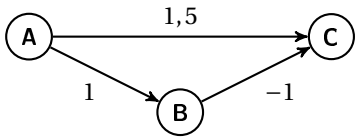


(b)

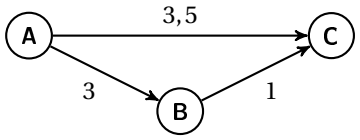
u	**A**	**B**	**C**	**D**	S
	0	∞	∞	∞	$\emptyset$
A		$1_{\mathbf{A}}$	$3_{\mathbf{A}}$		{**A**}
B			$3_{\mathbf{A}}$	$2_{\mathbf{B}}$	{**A**, **B**}
D			$3_{\mathbf{A}}$		{**A**, **B**, **D**}

Le chemin calculé est donc **A** – **B** – **D** avec un coût de 2 alors que le chemin **A** – **C** – **D** a un coût de 0.

(c) On considère le graphe suivant :



La plus petite pondération est −1. On peut donc obtenir des pondérations positives en additionnant 2 à chaque arc. On obtient alors ce graphe :

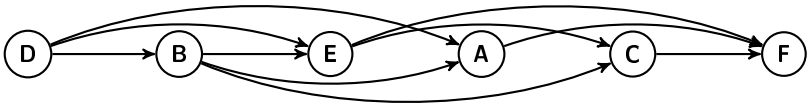


On effectue une trace de l'algorithme de Dijkstra pour le calcul d'un chemin minimum de **A** à **C**.

u	**A**	**B**	**C**	S
	0	∞	∞	$\emptyset$
A		$3_{\mathbf{A}}$	$3{,}5_{\mathbf{A}}$	{**A**}
B			$3{,}5_{\mathbf{A}}$	{**A**, **B**}
C				{**A**, **B**, **C**}

Le chemin calculé est **A**–**C**, or dans le graphe original ce chemin à un coût de 1,5 alors que le chemin **A** – **B** – **C** à un coût de 0.

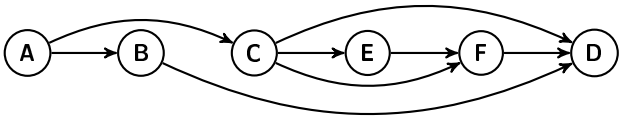
Rép. 3.9 (a) L'ordonnancement **D, B, E, A, C, F** est un tri topologique.



Note: ce graphe admet trois tris topologiques, les deux autres sont: **D, B, E, C, A, F** et **D, B, A, E, C, F**.

(b) Il n'existe pas de tri topologique dans ce graphe car il n'est pas acyclique. Par exemple, le chemin **A, B, E, C, D, A** est un cycle.

(c) L'ordonnancement **A, B, C, E, F, D** est un tri topologique.



Note: ce graphe admet quatre tris topologiques, les trois autres sont:
- **A, C, E, F, B, D**,
- **A, C, E, B, F, D**,
- **A, C, B, E, F, D**.

(d) Il n'existe pas de tri topologique dans ce graphe car il n'est pas orienté.

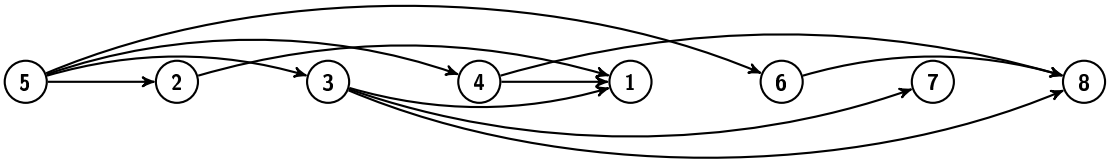
(e) L'ordonnancement **D, A, B, C** est un tri topologique.

Note: ce graphe admet de nombreux tri topologiques. Tout ordonnancement des sommets tel que **D** est avant **A** est un tri topologique.

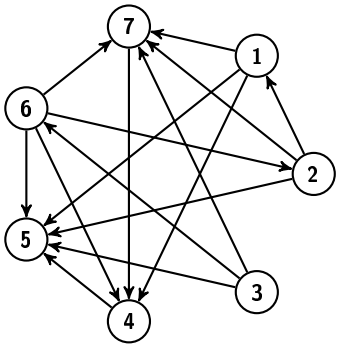
Rép. 3.10 (a)

Sommet **v**	Sommet(s) **u** avec **u**→**v** ∈ E
1	**2,3,4**
2	**5**
3	**5**
4	**5**
5	
6	**5**
7	**3**
8	**3,4,6**

Tri topologique: **5, 2, 3, 4, 1, 6, 7, 8**

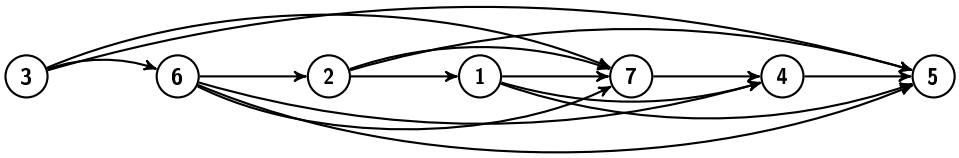


(b) Le graphe représenté par la matrice M peut être tracé comme ceci:

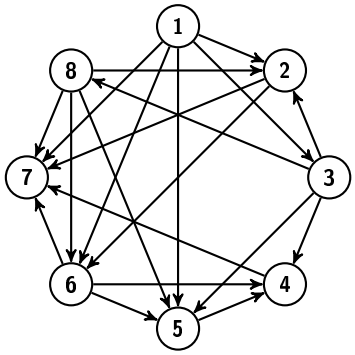


Sommet **v**	Sommet(s) **u** avec **u**→**v** ∈ E
1	**2**
2	**6**
3	
4	**1,6,7**
5	**1, 2, 3, 4, 6**
6	**3**
7	**1, 2, 3, 6**

Tri topologique : **3, 6, 2, 1, 7, 4, 5**.

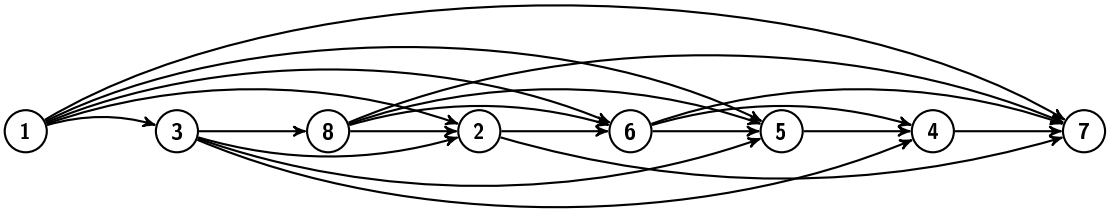


(c) Le graphe peut être représenté comme ceci :



Sommet **v**	Sommet(s) **u** avec **u**→**v** ∈ E
1	
2	**1,3,8**
3	**1**
4	**3,5,6**
5	**1,3,6,8**
6	**1,2,8**
7	**1,2,4,6,8**
8	**3**

Tri topologique : **1, 3, 8, 2, 6, 5, 4, 7**.



Rép. 3.11 (a) Le tri topologique est $T = [\mathbf{6},\mathbf{5},\mathbf{3},\mathbf{4},\mathbf{1},\mathbf{2}]$, l'état final du tableau L est

	1	**2**	**3**	**4**	**5**	**6**
$L =$	10	13	6	8	3	0

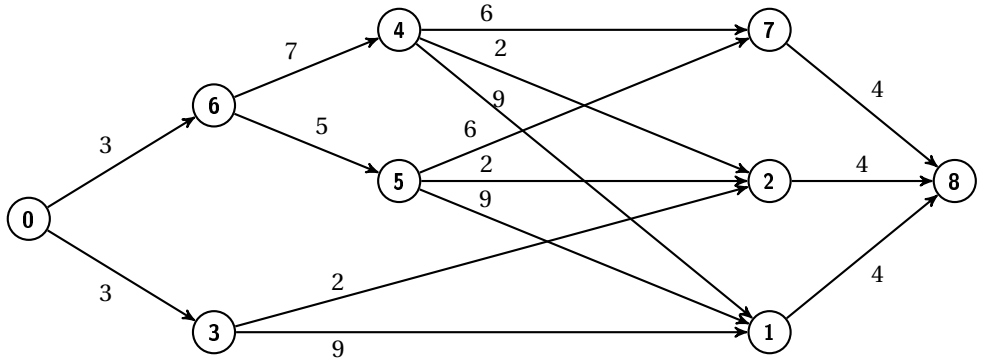
Le poids du chemin maximal est 13. Ce poids est réalisé par le chemin $\mathbf{6} - \mathbf{5} - \mathbf{4} - \mathbf{2}$.

(b) Le tri topologique est $T = [\mathbf{7},\mathbf{4},\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{5},\mathbf{6},\mathbf{8}]$, l'état final du tableau L est

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
$L =$	6	7	9	3	13	14	0	16

Le poids du chemin maximal est 16. Ce poids est réalisé par le chemin $\mathbf{7}-\mathbf{1}-\mathbf{2}-\mathbf{5}-\mathbf{8}$.

Rép. 3.12 On construit le graphe pondéré suivant :



À la fin de l'exécution de l'algorithme du chemin critique, le tableau L est :

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
$L =$	0	19	12	3	10	8	3	16	23

La durée minimale du projet est de 23 heures.

Chapitre 4

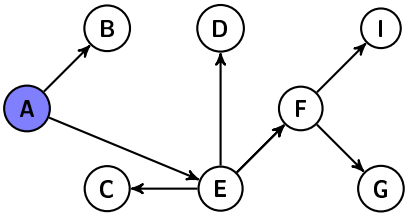
Rép. 4.1 G_1 : non, var l'ensemble des sommets du sous-graphe n'est pas égal à l'ensemble des sommets du graphe ($\mathbf{F}$ n'est pas inclus).

G_2 : Oui.

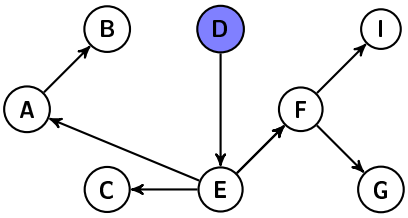
G_3 : non, le sous-graphe n'est pas connexe.

G_4 ; non, le sous-graphe n'est pas un arbre puisqu'il possède un circuit simple : $\mathbf{C}-\mathbf{E}-\mathbf{D}-\mathbf{C}$.

Rép. 4.2 Dans l'arbre enraciné en **A**, le parent de **E** est **A** et les enfants de **E** sont **C**, **D** et **F**.

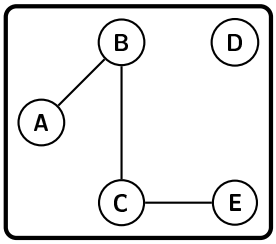


Dans l'arbre enraciné en **D**, le parent de **E** est **D** et les enfants de **E** sont **A**, **C** et **F**.

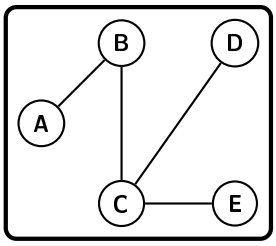


Rép. 4.3 (a) Non. Si G est un arbre ayant 5 sommets, alors, d'après le théorème 4.2 il a 4 arêtes.

(b) Oui, G est un graphe acyclique ayant 5 sommets et 3 arêtes.

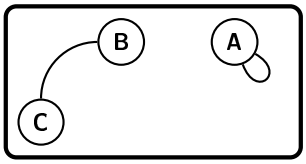


(c) Oui, G est un arbre ayant 5 sommets et 4 arêtes.

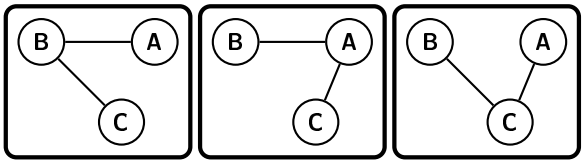


(d) Non, car si G est un arbre ayant 5 sommets, alors il a 4 arêtes (théorème 4.2).

(e) Oui, G est un graphe ayant 3 sommets, 2 arêtes et n'est pas un arbre car il n'est pas connexe. (Rappel : les boucles sont admises dans un graphe, contrairement à un graphe simple).



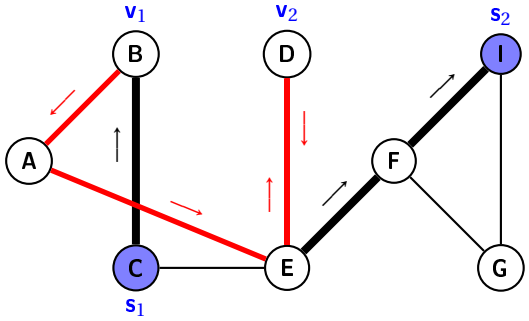
(f) Non. Il n'y a que 3 possibilités pour un graphe simple ayant 3 sommets identifiés **A**, **B** et **C** ainsi que 2 arêtes. Dans tous les cas, il s'agit d'un arbre. Les caractéristiques sont donc impossibles à réunir dans un graphe (le système est incohérent).



De façon plus générale, il est possible de démontrer que si $G = (V, E)$ est un graphe simple connexe et que $|V| - 1 = |E|$, alors G est un arbre.

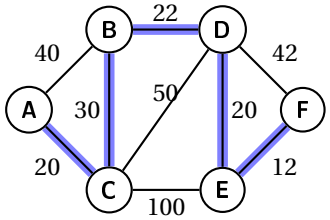
(g) Oui. Le graphe G comportant un unique sommet et aucune arête est connexe et acyclique; c'est donc un arbre n'ayant aucune arête.

Rép. 4.4 Si l'on retire l'arête $e = \mathsf{B}–\mathsf{D}$ du circuit, il est encore possible de relier **C** et **I**. En effet, on peut aller de $\mathsf{v}_1 = \mathsf{B}$ à $\mathsf{v}_2 = \mathsf{D}$ en prenant un détour dans l'autre sens du circuit: **B**–**A**–**E**–**D**. Le chemin alternatif reliant s_1 à s_2 obtenu par la méthode proposée sera donc: **C**–**B**–**A**–**E**–**D**–**E**–**F**–**I**.



Rép. 4.5 En classe ou en devoir.

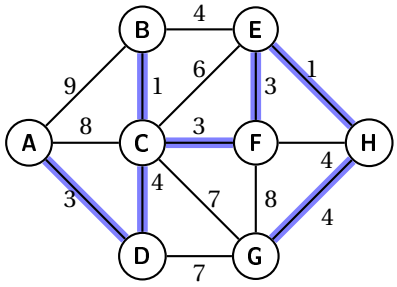
Rép. 4.6 Le poids total de cet arbre couvrant est 104. La troisième arête ajoutée est **B**–**D**.
L'arbre couvrant minimal est représenté en gras sur le graphe pondéré.



Trace de l'algorithme:

u	**a** = **A**	**B**	**C**	**D**	**E**	**F**	sommet ajouté à S	arête ajoutée à E_T
	0	∞	∞	∞	∞	∞		
A		4_{A}	2_{A}	∞	∞	∞	**A**	
C		4_{A}		800_{C}	100_{C}	∞	**C**	**A**–**C**
B				5_{B}	100_{C}	∞	**B**	**A**–**B**
D					100_{C}	600_{D}	**D**	**B**–**D**
E						300_{E}	**E**	**C**–**E**
F							**F**	**E**–**F**

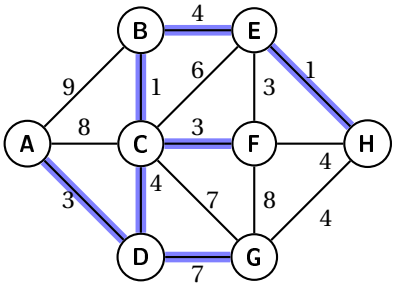
Rép. 4.7 Le poids de cet arbre est 19. La dernière arête ajoutée à l'arbre est **H** – **G**.



Rép. 4.8 Trace de l'algorithme de Dijkstra.

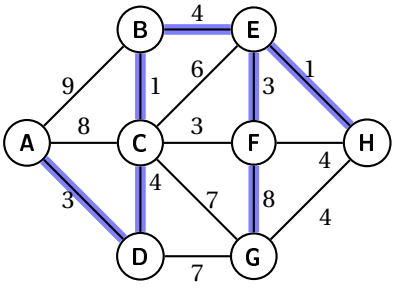
u	A	B	C	D	E	F	G	H	S
	0	∞	∞	∞	∞	∞	∞	∞	$\emptyset$
A		9_{A}	8_{A}	3_{A}	∞	∞	∞	∞	{A}
D		9_{A}	7_{D}		∞	∞	10_{D}	∞	{A, D}
C		8_{C}			13_{C}	10_{C}	10_{D}	∞	{A, C, D}
B					12_{B}	10_{C}	10_{D}	∞	{A, B, C, D}
F					12_{B}		10_{D}	14_{F}	{A, B, C, D, F}
G					12_{B}			14_{F}	{A, B, C, D, F, G}
E								13_{E}	{A, B, C, D, E, F, G}
H									{A, B, C, D, E, F, G, H}

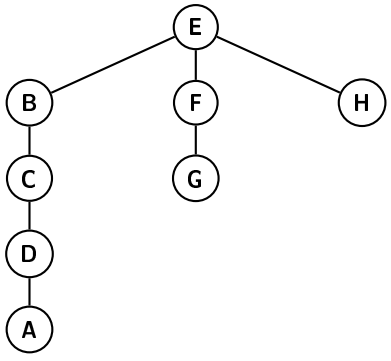
En remontant les prédécesseurs de chacun des sommets, on obtient l'arbre couvrant suivant.



Comme on l'a vu à l'exercice 4.7, un arbre couvrant minimal pour ce graphe a un poids de 19. Or l'arbre couvrant obtenu a un poids de 23. Ainsi, un arbre couvrant des chemins de longueur (ou poids) minimale n'est pas nécessairement un arbre couvrant minimal.

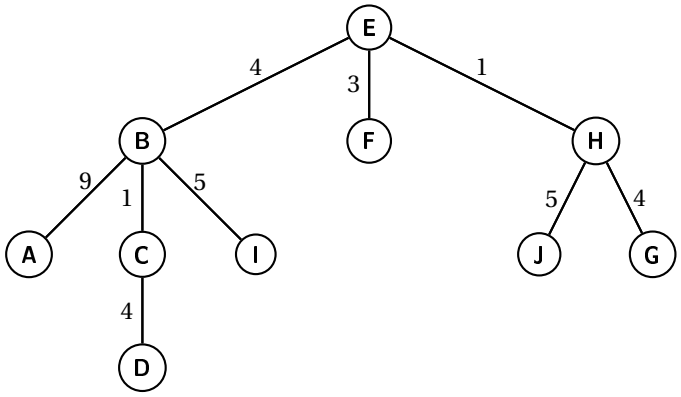
Rép. 4.9 Voici l'état des tableaux L et P à la fin de l'algorithme :
$L = [12, 4, 5, 9, 0, 3, 5, 1]$. $P = [\mathsf{D}, \mathsf{E}, \mathsf{B}, \mathsf{C}, \texttt{null}, \mathsf{E}, \mathsf{H}, \mathsf{E}]$
En remontant les prédécesseurs de chacun des sommets, on obtient l'arbre couvrant suivant.





Comme on l'a vu à l'exercice 4.7, un arbre couvrant minimal pour ce graphe a un poids de 19. Or l'arbre couvrant obtenu a un poids de 24 : il n'est donc ps minimal.

Rép. 4.10 Voici l'arbre des chemins de poids minimaux issus de **E**. Son poids est 40.



Chapitre 5

Rép. 5.1 (a) $f \in O(\sqrt{n})$
(b) $f \in O(n^2)$
(c) $f \in O(n)$
(d) $f \in O(n^3)$
(e) $f \in O(n^4 \log(n))$
(f) $f \in O(2^n)$
(g) $f \in O(5^n)$
(h) $f \in O(n!)$

Rép. 5.2 (a) $O(n^2)$
(b) $O(n^3)$
(c) $O(n^4)$
(d) $O(n^6)$
(e) $O(n^2)$
(f) $O(n^{5/3})$
(g) $O(n^{1/10})$
(h) $O(2^n)$
(i) $O(1)$
(j) $O(4^n)$
(k) $O(9^n)$

Rép. 5.3 (a) $f(n) \in O(n^{3/2})$ et $g(n) \in O(n\log(n))$
(b) Le deuxième, car, $g(n) \in O(f(n))$ mais $f(n) \notin O(g(n))$.

Rép. 5.4 (a) $f(n) \in O(n^2\log(n))$ et $g(n) \in O(n^{5/3})$
(b) Le deuxième, car, $g(n) \in O(f(n))$ mais $f(n) \notin O(g(n))$.

Rép. 5.5 (a) $O(n^4)$
(b) $O(n^{5/2})$
(c) $O(n^7)$
(d) $O(2^n)$
(e) $O(n^3)$
(f) $O(4^n)$
(g) $O(2^n)$
(h) $O(2^n\log(n))$
(i) $O(n^{10})$
(j) $O(n^{1/2})$
(k) $O(1{,}001^n)$
(l) $O(n^{1/2})$
(m) $O\left(\left(\frac{3}{2}\right)^n\right)$
(n) $O(n^{5/2})$

Rép. 5.6 (a) $(6+2\cdot 2)+(6+2\cdot 3)+(6+2\cdot 4) = 36$
(b) $f(n) = \sum_{i=2}^{n-2}(n+2i) = \sum_{i=2}^{n-2} n + \sum_{i=2}^{n-2} 2i.$
On traite les sommations séparément :
$\sum_{i=2}^{n-2} n = ((n-2)-2+1)n = n^2-3n,$
$\sum_{i=2}^{n-2} 2i = 2\sum_{i=2}^{n-2} i = 2\left(\sum_{i=1}^{n-2} i - \sum_{i=1}^{1} i\right) = 2\left(\frac{(n-2)((n-2)+1)}{2} - \frac{(1)(1+1)}{2}\right) = n^2-3n.$
En regroupant : $f(n) = (n^2-3n)+(n^2-3n) = 2n^2-6n$. Ce qui donne bien 36 lorsque $n=6$.

Rép. 5.7 (a) $f(n) = \sum_{i=1}^{n}(n+i+1) = \sum_{i=1}^{n} n + \sum_{i=1}^{n} i + \sum_{i=1}^{n} 1$
On traite les sommations séparément :
$\sum_{i=1}^{n} n = n(n-1+1) = n^2, \qquad \sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \frac{1}{2}n^2+\frac{1}{2}n, \qquad \sum_{i=1}^{n} 1 = (n-1+1) = n.$
En regroupant : $f(n) = \frac{3}{2}n^2+\frac{3}{2}n$

(b) $f(n) = \sum_{i=0}^{n+1}(5n+3i+4) = \sum_{i=0}^{n+1} 5n + \sum_{i=0}^{n+1} 3i + \sum_{i=0}^{n+1} 4.$

On traite les sommations séparément :

$\sum_{i=0}^{n+1} 5n = 5n(n+1-0+1) = 5n^2+10n,$

$\sum_{i=0}^{n+1} 3i = 3\sum_{i=0}^{n+1} i = 3\left(0+\sum_{i=1}^{n+1} i\right) = \frac{3(n+1)(n+2)}{2} = \frac{3}{2}n^2+\frac{9}{2}n+3,$

$\sum_{i=0}^{n+1} 4 = 4(n+1-0+1) = 4n+8.$

En regroupant : $f(n) = \frac{13}{2}n^2+\frac{37}{2}n+11.$

(c) Attention, le $-n$ ne fait pas partie de la sommation : $f(n) = \left(\sum_{i=n}^{2n}(i+1)\right) - n = \sum_{i=n}^{2n} i + \sum_{i=n}^{2n} 1 - n.$

On traite les sommations séparément :

$\sum_{i=n}^{2n} i = \sum_{i=1}^{2n} i - \sum_{i=1}^{n-1} i = \frac{2n(2n+1)}{2} - \frac{(n-1)((n-1)+1)}{2} = (2n^2+n) - \left(\frac{1}{2}n^2-\frac{1}{2}n\right) = \frac{3}{2}n^2+\frac{3}{2}n,$

$\sum_{i=n}^{2n} 1 = 2n-n+1 = n+1.$

En regroupant : $f(n) = \left(\frac{3}{2}n^2+\frac{3}{2}n\right)+(n+1)-n = \frac{3}{2}n^2+\frac{3}{2}n+1.$

(d) $\frac{4^n-16}{3}$

(e) $f(n) = \sum_{i=1}^{2n-1}\sum_{j=0}^{n-1}(j+1) = \sum_{i=1}^{2n-1}\left(\sum_{j=0}^{n-1} j + \sum_{j=0}^{n-1} 1\right) = \sum_{i=1}^{2n-1}\left(\sum_{j=1}^{n-1} j + n\right) = \sum_{i=1}^{2n-1}\left(\frac{n(n-1)}{2}+n\right) = \frac{1}{2}\sum_{i=1}^{2n-1} n^2 + \frac{1}{2}\sum_{i=1}^{2n-1} n.$

On traite les sommations séparément :

$\frac{1}{2}\sum_{i=1}^{2n-1} n^2 = n^2(2n-1-1+1) = n^3-\frac{1}{2}n^2.$

$\frac{1}{2}\sum_{i=1}^{2n-1} n = n(2n-1-1+1) = n^2-\frac{1}{2}n.$

En regroupant : $f(n) = n^3+\frac{1}{2}n^2-\frac{1}{2}n.$

(f) $f(n) = \sum_{i=0}^{n-1}\sum_{j=1}^{n}(n^2+i+j) = \sum_{i=0}^{n-1}\left(\sum_{j=1}^{n}(n^2+i)+\sum_{j=1}^{n} j\right)$

On traite les sommations internes séparément :

$\sum_{j=1}^{n}(n^2+i) = (n^2+i)n = n^3+in$

$\sum_{j=1}^{n} j = \frac{n(n+1)}{2} = \frac{1}{2}n^2+\frac{1}{2}n$

En regroupant :

$$f(n) = \sum_{i=0}^{n-1}\left(n^3 + in + \frac{1}{2}n^2 + \frac{1}{2}n\right) = \sum_{i=0}^{n-1}\left(n^3 + \frac{1}{2}n^2 + \frac{1}{2}n\right) + \sum_{i=0}^{n-1} in$$

$$= n^4 + \frac{1}{2}n^3 + \frac{1}{2}n^2 + n\sum_{i=1}^{n-1} i = n^4 + \frac{1}{2}n^3 + \frac{1}{2}n^2 + n\frac{n(n-1)}{2}$$

$$= n^4 + n^3.$$

(g) $$f(n) = \sum_{k=0}^{n}\left(\sum_{l=1}^{n^2}(k-l) + k^2\right) = \sum_{k=0}^{n}\left(\sum_{l=1}^{n^2} k - \sum_{l=1}^{n^2} l + k^2\right)$$

$$= \sum_{k=0}^{n}\left(kn^2 - \frac{n^2(n^2+1)}{2} + k^2\right) = n^2\sum_{k=1}^{n} k - \sum_{k=0}^{n}\frac{n^2(n^2+1)}{2} + \sum_{k=1}^{n} k^2$$

$$= n^2\frac{n(n+1)}{2} - \frac{n^2(n^2+1)}{2}(n+1) + \frac{n(n+1)(2n+1)}{6}$$

$$= \left(\frac{1}{2}n^4 + \frac{1}{2}n^3\right) - \left(\frac{1}{2}n^5 + \frac{1}{2}n^4 + \frac{1}{2}n^3 + \frac{1}{2}n^2\right) + \left(\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n\right)$$

$$= -\frac{1}{2}n^5 + \frac{1}{3}n^3 + \frac{1}{6}n.$$

(h) $$\frac{5^{n+1} - 5^{10}}{2}$$

Rép. 5.8 Soit $f(n)$ le nombre d'additions effectuées par l'algorithme en fonction de n,

$$f(n) = \sum_{i=2}^{n}\left(2 + \left(\sum_{j=1}^{n-i} 1\right) + 1\right)$$

$$= \sum_{i=2}^{n}(n - i + 3)$$

$$= \frac{1}{2}n^2 + \frac{3}{2}n - 2.$$

On conclut que $f(n) \in O(n^2)$.

Rép. 5.9 La fonction `P` détermine si l'élément x est présent dans les 10 premières positions du tableau d'éléments T. Elle requiert $f(n) = 10$ comparaisons. On a donc $f(n) \in O(1)$.

Rép. 5.10 La fonction `P` requiert $f(n) = 50n + \frac{n(n+1)}{2}$ comparaisons pour traiter un tableau de taille n. On a donc $f(n) \in O(n^2)$.

Rép. 5.11 La fonction `P` requiert $f(n) = \frac{n^2(n+1)}{2} + 5n$ comparaisons pour traiter un tableau de taille n. On a donc $f(n) \in O(n^3)$.

Rép. 5.12 (a) Compter le nombre de comparaisons d'éléments à la ligne 4 ; $f(n) = \sum_{i=0}^{n-1} 1 = n \in O(n)$.

(b) Compter le nombre de comparaisons d'éléments à la ligne 5 ; $f(n) = \sum_{i=0}^{n-1} 1 = n \in O(n)$.

(c) Compter le nombre de comparaisons d'éléments à la ligne 4 ; $f(n) = \sum_{i=0}^{n-1} 1 = n \in O(n)$.

(d) Compter le nombre de comparaisons de cases à la ligne 5 ; $f(n) = \sum_{i=0}^{n-2}\left(\sum_{j=i+1}^{n-1} 1\right) = \frac{1}{2}n^2 - \frac{1}{2}n \in O(n^2)$.

(e) Compter le nombre de comparaisons de cases à la ligne 5 ; $f(n) = \sum_{k=0}^{2n-2} 1 = 2n-1 \in O(n)$.

Note : la borne supérieure est, $2n-2$ car il ne peut pas y avoir de comparaison effectuée à la dernière itération de la boucle. En effet, à la dernière itération, on a forcément $i == n_1$ ou $j == n_2$.

(f) Compter le nombre de comparaisons de caractères à la ligne 8 ; $f(n) = \sum_{i=0}^{n/2}\left(\sum_{j=0}^{n/2-1} 1\right) = \frac{1}{4}n^2 + \frac{1}{2}n \in O(n^2)$.

Chapitre 6

Rép. 6.1 (a) $f(1)=2$, $f(2)=4$, $f(3)=6$, $f(4)=8$, $f(5)=10$, $f(6)=12$, $f(7)=14$.
Le terme $f(n)$ est le n-ième nombre pair. De manière équivalente, $f(n)=2n$.

(b) $f(1)=3$, $f(2)=5$, $f(3)=7$, $f(4)=9$, $f(5)=11$, $f(6)=13$, $f(7)=15$.
Le terme $f(n)$ est le n-ième nombre impair. De manière équivalente, $f(n)=2n+1$.

(c) $f(1)=1$, $f(2)=0$, $f(3)=1$, $f(4)=0$, $f(5)=1$, $f(6)=0$, $f(7)=1$.
Le terme $f(n)$ est 0 si n est pair, 1 si n est impair. En termes plus techniques, on dit que f est la *fonction caractéristique* des nombres impairs.

(d) $f(1)=1$, $f(2)=4$, $f(3)=9$, $f(4)=16$, $f(5)=25$, $f(6)=36$, $f(7)=49$.
Le terme $f(n)$ est le n-ième carré. De manière équivalente $f(n)=n^2$.

(e) $f(1)=1$, $f(2)=11$, $f(3)=111$, $f(4)=1111$, $f(5)=11111$, $f(6)=111111$, $f(7)=1111111$.
Le terme $f(n)$ est le nombre obtenu en concaténant n occurrences de 1, en base 10. De manière équivalente, $f(n)=\sum_{i=0}^{n-1} 10^i = \frac{10^n-1}{9}$, ou encore $f(n)=\underbrace{11111\cdots111}_{n \text{ fois}}$.

(f) $f(1)=0$, $f(2)=1$, $f(3)=1$, $f(4)=0$, $f(5)=1$, $f(6)=0$, $f(7)=1$.
La fonction f est la *fonction caractéristique* des nombres premiers.

Rép. 6.2 (a) À l'aide de la méthode itérative, on trouve

$$\begin{aligned}
a_0 &= 1\\
a_1 &= 2a_0\\
a_2 &= 2a_1 = 2(2a_0) = 2^2 a_0\\
a_3 &= 2a_2 = 2(2^2 a_0) = 2^3 a_0\\
&\vdots\\
a_n &= 2^n a_0.
\end{aligned}$$

En utilisant $a_0 = 1$ donné par la définition, on obtient la solution : $a_n = 2^n$.

(b) À l'aide de la méthode itérative, on trouve

$$\begin{aligned}
b_0 &= 0\\
b_1 &= b_0 + 2\cdot 1\\
b_2 &= b_1 + 2\cdot 2 = (b_0 + 2\cdot 1) + 2\cdot 2\\
b_3 &= b_2 + 2\cdot 3 = (b_0 + 2\cdot 1 + 2\cdot 2) + 2\cdot 3\\
&\vdots\\
b_n &= b_0 + \sum_{i=1}^{n} 2i = b_0 + 2\sum_{i=1}^{n} i = b_0 + 2\left(\frac{n(n+1)}{2}\right) = b_0 + n^2 + n.
\end{aligned}$$

En utilisant $b_0 = 0$ donné par la définition, on obtient la solution : $b_n = n^2 + n$.

(c) À l'aide de la méthode itérative, on trouve

$$\begin{aligned}
c_0 &= 1\\
c_1 &= c_0 + 1 + 2\\
c_2 &= c_1 + 2 + 2 = (c_0 + 1 + 2) + 2 + 2\\
c_3 &= c_2 + 3 + 2 = (c_0 + 1 + 2 + 2 + 2) + 3 + 2\\
&\vdots\\
c_n &= c_0 + \sum_{i=1}^{n} (i+2) = c_0 + \sum_{i=1}^{n} i + \sum_{i=1}^{n} 2 = c_0 + \frac{n(n+1)}{2} + 2n = c_0 + \frac{n^2}{2} + \frac{5n}{2}.
\end{aligned}$$

En utilisant $c_0 = 1$ donné par la définition, on obtient la solution : $c_n = 1 + \frac{n^2}{2} + \frac{5n}{2}$.

(d) À l'aide de la méthode itérative, on trouve

$$\begin{aligned}
f(0) &= 2\\
f(1) &= 3f(0) + 2\\
f(2) &= 3f(1) + 2 = 3(3f(0) + 2) + 2 = 3^2 f(0) + 3 \cdot 2 + 2\\
f(3) &= 3f(2) + 2 = 3(3^2 f(0) + 3 \cdot 2 + 2) + 2 = 3^3 f(0) + 3^2 \cdot 2 + 3 \cdot 2 + 2\\
&\vdots\\
f(n) &= 3^n f(0) + \sum_{i=0}^{n-1} 3^i \cdot 2 = 3^n f(0) + 2\left(\frac{3^n - 1}{2}\right) = 3^n f(0) + 3^n - 1 = 3^n (f(0) + 1) - 1.
\end{aligned}$$

En utilisant $f(0) = 2$ donné par la définition, on obtient la solution : $f(n) = 3^n(2+1) - 1 = 3^{n+1} - 1$.

(e) À l'aide de la méthode itérative, on trouve

$$\begin{aligned}
f(0) &= 5\\
f(1) &= 3f(0) + 2\\
f(2) &= 3f(1) + 2 = 3(3f(0) + 2) + 2 = 3^2 f(0) + 3 \cdot 2 + 2\\
f(3) &= 3f(2) + 2 = 3(3^2 f(0) + 3 \cdot 2 + 2) + 2 = 3^3 f(0) + 3^2 \cdot 2 + 3 \cdot 2 + 2\\
&\vdots\\
f(n) &= 3^n f(0) + \sum_{i=0}^{n-1} 3^i \cdot 2 = 3^n f(0) + 2\left(\frac{3^n - 1}{2}\right) = 3^n f(0) + 3^n - 1 = 3^n (f(0) + 1) - 1.
\end{aligned}$$

En utilisant $f(0) = 5$ donné par la définition, on obtient la solution : $f(n) = 3^n(5+1) - 1 = 6 \cdot 3^n - 1$.

(f) À l'aide de la méthode itérative, on trouve

$$\begin{aligned}
f(0) &= 5\\
f(1) &= 1 \cdot f(0)\\
f(2) &= 2 \cdot f(1) = 2(1 \cdot f(0)) = 2 \cdot 1 \cdot f(0)\\
f(3) &= 3 \cdot f(2) = 3(2 \cdot 1 \cdot f(0)) = 3 \cdot 2 \cdot 1 \cdot f(0)\\
&\vdots\\
f(n) &= n \cdots 3 \cdot 2 \cdot 1 \cdot f(0).
\end{aligned}$$

En utilisant $f(0) = 5$ donné par la définition, on obtient la solution : $f(n) = n! \cdot 5$.

(g) À l'aide de la méthode itérative, on trouve

$$\begin{aligned} h_0 &= 3 \\ h_1 &= 10h_0 \\ h_2 &= 10h_1 = 10(10h_0) = 10^2 h_0 \\ h_3 &= 10h_2 = 10(10^2 h_0) = 10^3 h_0 \\ &\vdots \\ h_n &= 10^n h_0. \end{aligned}$$

En utilisant $h_0 = 3$ donné par la définition, on obtient la solution : $h_n = 3 \cdot 10^n$.

(h) À l'aide de la méthode itérative, on trouve

$$\begin{aligned} f(2) &= 3 \\ f(3) &= 4f(2) + 6 \\ f(4) &= 4f(3) + 6 = 4(4f(2) + 6) + 6 = 4^2 f(2) + 4 \cdot 6 + 6 \\ f(5) &= 4f(4) + 6 = 4(4^2 f(2) + 4 \cdot 6 + 6) + 6 = 4^3 f(2) + 4^2 \cdot 6 + 4 \cdot 6 + 6 \\ &\vdots \\ f(n) &= 4^{n-2} f(2) + \sum_{i=0}^{n-3} 4^i \cdot 6 = 4^{n-2} f(2) + 6\Big(\frac{4^{n-2} - 1}{4 - 1}\Big) = 4^{n-2} f(2) + 2 \cdot 4^{n-2} - 2 = 4^{n-2}(f(2) + 2) - 2. \end{aligned}$$

En utilisant $f(2) = 3$, on obtient la solution : $f(n) = 4^{n-2}(3 + 2) - 2 = 5 \cdot 4^{n-2} - 2$.

Rép. 6.3 (a) $f(128) = 8$
(b) $f(81) = 8463$
(c) $f(64) = 24512$
(d) $f(729) = 5\,314\,410$

Rép. 6.4 (a) $f(n) = 6n - 3 \in O(n)$.

(b) $f(n) = 2n^2 - n \in O(n^2)$.

(c) $f(n) = 2n \log_3(n) + \dfrac{3n}{2} - \dfrac{1}{2} \in O(n \log(n))$.

(d) $f(n) = 2n^2 + 3n^2 \log_3(n) \in O(n^2 \log(n))$.

Rép. 6.5 $f(n) = (d-1)\log_d(n) + 1 \in O(\log(n))$

Rép. 6.6 (a) $f(n) = 2f(n/2) + n - 1, \quad f(1) = 0.$
(b) $f(n) = n \cdot \log_2(n) - n + 1 \in O(n \log(n))$.

Chapitre 7

Rép. 7.1 (a) On définit $P(n)$: « $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$ » et on veut montrer que $\forall n \geq 1$, $P(n) \equiv$ **vrai**.

1. **Cas de base**:
   Pour $n = 1$, on a $P(1) \equiv$ **vrai**, car $\sum_{i=1}^{1} i^2 = 1$.
2. **Étape de récurrence**:
   Soit $k \geq 1$.
   - Hypothèse de récurrence: $P(k)$: « $\sum_{i=1}^{k} i^2 = \frac{k(k+1)(2k+1)}{6}$ » $\equiv$ **vrai**
   - Objectif: $P(k+1)$: « $\sum_{i=1}^{k} i^2 = \frac{(k+1)(k+2)(2k+3)}{6}$ » $\equiv$ **vrai**
   - Calculs:

$$\begin{aligned}
\sum_{i=1}^{k+1} i^2 &= \sum_{i=1}^{k} i^2 + (k+1)^2 \\
&= \frac{k(k+1)(2k+1)}{6} + (k+1)^2 && \text{(par hyp. de réc.)} \\
&= \frac{k(k+1)(2k+1) + 6(k+1)^2}{6} \\
&= \frac{(k+1)\big(k(2k+1) + 6(k+1)\big)}{6} \\
&= \frac{(k+1)\big(2k^2 + 7k + 6\big)}{6} \\
&= \frac{(k+1)(k+2)(2k+3)}{6}
\end{aligned}$$

□

(b) On définit $P(n)$: « $\sum_{i=0}^{n} r^i = \frac{r^{n+1}-1}{r-1}$, si $r \in \mathbb{R} \setminus \{1\}$ » et on veut montrer que $\forall n \geq 0$, $P(n) \equiv$ **vrai**.

1. **Cas de base**:
   Pour $n = 0$, on a $P(0) \equiv$ **vrai**, car $\sum_{i=0}^{0} r^i = 1 = \frac{r-1}{r-1}$.
2. **Étape de récurrence**:
   Soit $k \geq 0$.
   - Hypothèse de récurrence: $P(k)$: « $\sum_{i=0}^{k} r^i = \frac{r^{k+1}-1}{r-1}$, si $r \in \mathbb{R} \setminus \{1\}$ » $\equiv$ **vrai**
   - Objectif: $P(k+1)$: « $\sum_{i=0}^{k+1} r^i = \frac{r^{k+2}-1}{r-1}$, si $r \in \mathbb{R} \setminus \{1\}$ » $\equiv$ **vrai**
   - Calculs:

$$\begin{aligned}
\sum_{i=0}^{k+1} r^i &= \sum_{i=0}^{k} r^i + r^{k+1} \\
&= \frac{r^{k+1}-1}{r-1} + r^{k+1} && \text{(par hyp. de réc.)} \\
&= \frac{r^{k+1} - 1 + r^{k+1}(r-1)}{r-1} \\
&= \frac{r^{k+1}(1 + (r-1)) - 1}{r-1} \\
&= \frac{r^{k+1} r - 1}{r-1} \\
&= \frac{r^{k+2}-1}{r-1}
\end{aligned}$$

□

(c) Remarque : il est difficile d'utiliser directement un énoncé de la forme « 3 divise $(n^3 - n)$ » dans le cadre d'une preuve formelle. On va plutôt utiliser la définition mathématique :

$$\exists a \in \mathbb{Z}, \quad n^3 - n = 3a.$$

On définit $P(n)$: « $\exists a \in \mathbb{Z}, \quad n^3 - n = 3a$ » et on veut montrer que $\forall n \geq 0$, $P(n) \equiv$ **vrai**.

1. **Cas de base** :
   Pour $n = 0$, on a $P(0) \equiv \mathbf{V}$, car $0^3 - 0 = 0 = 3 \cdot 0$.
2. **Étape de récurrence** :
   Soit $k \geq 0$.
   - Hypothèse de récurrence : $P(k)$: « $\exists a \in \mathbb{Z}, \quad k^3 - k = 3a$ » $\equiv$ **vrai**.
   - Objectif : $P(k+1)$: « $\exists a \in \mathbb{Z}, \quad (k+1)^3 - (k+1) = 3a$ » $\equiv$ **vrai**.
   - Calculs :

$$\begin{aligned}
(k+1)^3 - (k+1) &= (k^3 + 3k^2 + 3k + 1) - (k+1) \\
&= (k^3 - k) + 3k^2 + 3k \\
&= 3a + 3k^2 + 3k && \text{(par hyp. de réc.)} \\
&= 3(a + k^2 + k) \\
&= 3b && \text{(où } b = a + k^2 + k \in \mathbb{Z}\text{)}
\end{aligned}$$

□

(d) Remarque : il est difficile d'utiliser directement un énoncé de la forme « 2 divise $(n^2 + 3n)$ » dans le cadre d'une preuve formelle. On va plutôt utiliser la définition mathématique :

$$\exists a \in \mathbb{Z}, \quad n^2 + 3n = 2a.$$

On définit $P(n)$: « $\exists a \in \mathbb{Z}, \quad n^2 + 3n = 2a$ » et on veut montrer que $\forall n \geq 0$, $P(n) \equiv$ **vrai**.

1. **Cas de base** :
   Pour $n = 0$, on a $P(0) \equiv \mathbf{V}$, car $0^2 + 3 \cdot 0 = 0 = 2 \cdot 0$.
2. **Étape de récurrence** :
   Soit $k \geq 0$.
   - Hypothèse de récurrence : $P(k)$: « $\exists a \in \mathbb{Z}, \quad k^2 + 3k = 2a$ » $\equiv$ **vrai**.
   - Objectif : $P(k+1)$: « $\exists a \in \mathbb{Z}, \quad (k+1)^2 + 3(k+1) = 2a$ » $\equiv$ **vrai**.
   - Calculs :

$$\begin{aligned}
(k+1)^2 + 3(k+1) &= (k^2 + 2k + 1) + 3(k+1) \\
&= (k^2 + 3k) + 2k + 4 \\
&= 2a + 2k + 4 && \text{(par hyp. de réc.)} \\
&= 2(a + k + 2) \\
&= 2b && \text{(où } b = a + k + 2 \in \mathbb{Z}\text{)}
\end{aligned}$$

□

(e) On définit $P(n)$: « $2n + 1 \leq 2^n$ » et on veut montrer que $\forall n \geq 3$, $P(n) \equiv$ **vrai**.

1. **Cas de base** :
   Pour $n = 3$, on a $P(3) \equiv$ **vrai**, car $2 \cdot 3 + 1 = 7$, $2^3 = 8$, et donc « $2 \cdot 3 + 1 \leq 2^3$ » $\equiv$ **vrai**.
2. **Étape de récurrence** :
   Soit $k \geq 3$.
   - Hypothèse de récurrence : $P(k)$: « $2k + 1 \leq 2^k$ » $\equiv$ **vrai**.
   - Objectif : $P(k+1)$: « $2(k+1) + 1 \leq 2^{k+1}$ » $\equiv$ **vrai**.

– Calculs :

$$\begin{aligned} 2(k+1)+1 = (2k+1)+2 &\leq 2^k+2 && \text{(par hyp. de réc.)} \\ &\leq 2^k+2^k && \text{(comme } 2 \leq 2^k \text{ pour tout } k \geq 1) \\ &= 2\cdot 2^k \\ &= 2^{k+1} \end{aligned}$$

□

(f) On définit $P(n)$: « $n^2 \leq 2^n$ » et on veut montrer que $\forall\, n \geq 4$, $P(n) \equiv$ **vrai**.

1. **Cas de base** :
   Pour $n = 4$, on a $P(4) \equiv$ **vrai**, car $4^2 = 16$, $2^4 = 16$, et donc « $4^2 \leq 2^4$ » $\equiv$ **vrai**.
2. **Étape de récurrence** :
   Soit $k \geq 4$.
   – Hypothèse de récurrence : $P(k)$: « $k^2 \leq 2^k$ » $\equiv$ **vrai**.
   – Objectif : $P(k+1)$: « $(k+1)^2 \leq 2^{k+1}$ » $\equiv$ **vrai**.
   – Calculs :

$$\begin{aligned} (k+1)^2 = k^2+(2k+1) &\leq 2^k+2^k && \text{(par hyp. de réc. et exercice 7.1 (e))} \\ &= 2\cdot 2^k \\ &= 2^{k+1} \end{aligned}$$

□

(g) On définit $P(n)$: « $2^n < n!$ » et on veut montrer que $\forall\, n \geq 4$, $P(n) \equiv$ **vrai**.

1. **Cas de base** :
   Pour $n = 4$, on a $P(4) \equiv$ **vrai**, car $2^4 = 16$, $4! = 24$, et donc « $2^4 < 4!$ » $\equiv$ **vrai**.
2. **Étape de récurrence** :
   Soit $k \geq 4$.
   – Hypothèse de récurrence : $P(k)$: « $2^k < k!$ » $\equiv$ **vrai**.
   – Objectif : $P(k+1)$: « $2^{k+1} < (k+1)!$ » $\equiv$ **vrai**.
   – Calculs :

$$\begin{aligned} 2^{k+1} = 2\cdot 2^k &< (k+1)\cdot 2^k && \text{(comme } k \geq 4 \text{ on a que } 2 < k+1) \\ &< (k+1)\cdot k! && \text{(par hyp. de réc.)} \\ &< (k+1)! && \text{(par la déf. de la factorielle)} \end{aligned}$$

□

(h) On définit $P(n)$: « $n! < n^n$ » et on veut montrer que $\forall\, n \geq 2$, $P(n) \equiv$ **vrai**.

1. **Cas de base** :
   Pour $n = 2$, on a $P(2) \equiv$ **vrai**, car $2! = 2$, $2^2 = 4$, et donc « $2! < 2^2$ » $\equiv$ **vrai**.
2. **Étape de récurrence** :
   Soit $k \geq 2$.
   – Hypothèse de récurrence : $P(k)$: « $k! < k^k$ » $\equiv$ **vrai**.
   – Objectif : $P(k+1)$: « $(k+1)! < (k+1)^{(k+1)}$ » $\equiv$ **vrai**.
   – Calculs :

$$\begin{aligned} (k+1)! = k!(k+1) &< k^k(k+1) && \text{(par hyp. de réc.)} \\ &< (k+1)^k(k+1) && \text{(comme } k < (k+1)\text{)} \\ &< (k+1)^{(k+1)} && \text{(par la déf. de la factorielle)} \end{aligned}$$

□

(i) On définit la fonction propositionnelle $P(n) : \overline{A_1 \cup A_2 \cup \cdots \cup A_n} = \overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_n}$ et on montre que $\forall\, n \geq 2, P(n) \equiv$ **vrai**.

1. **Cas de base:**
   $P(2) \equiv$ **vrai** car par la loi de De Morgan, $\overline{A_1 \cup A_2} = \overline{A_1} \cap \overline{A_2}$.
2. **Étape de récurrence:**
   Soit $k \geq 2$.
   - Hypothèse de récurrence: $P(k)$: « $\overline{A_1 \cup A_2 \cup \cdots \cup A_k} = \overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_k}$ » $\equiv$ **vrai**.
   - Objectif: $P(k+1)$: « $\overline{A_1 \cup A_2 \cup \cdots \cup A_{k+1}} = \overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_{k+1}}$ » $\equiv$ **vrai**.
   - Calculs:

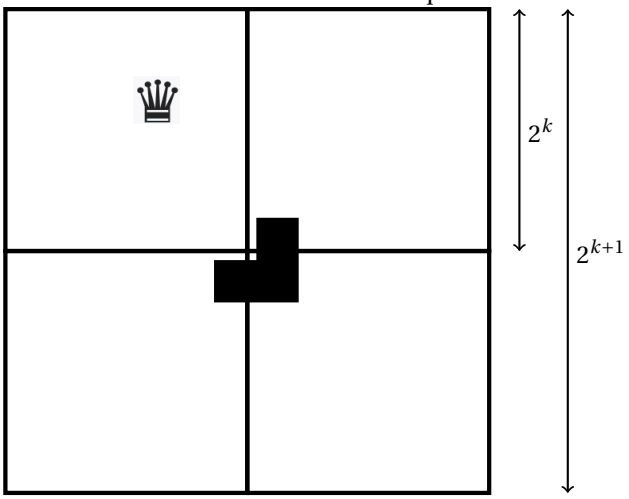
$$\begin{aligned}
\overline{A_1 \cup A_2 \cup \cdots \cup A_{k_1}} &= \overline{(A_1 \cup A_2 \cup \cdots \cup A_k) \cup A_{k+1}} \\
&= \overline{A_1 \cup A_2 \cup \cdots \cup A_k} \cap \overline{A_{k+1}} && \text{(par la loi de De Morgan)} \\
&= \overline{A_1} \cap \overline{A_2} \cap \cdots \overline{A_k} \cap \overline{A_{k+1}} && \text{(par hyp. de réc.)}
\end{aligned}$$

□

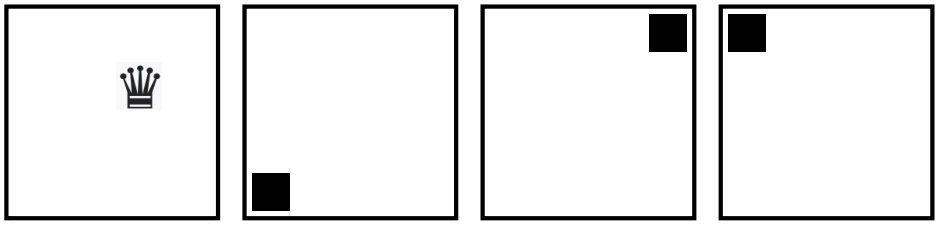
(j) On définit $P(n)$: « l'échiquier $2^n \times 2^n$ dont une case est occupée par la reine est pavable par des triominos en forme de L » et on montre que $\forall n \in \mathbb{N}, P(n) \equiv$ **vrai**.

1. **Cas de base**:
   Pour $n = 0$, on a $P(0) \equiv$ **vrai** car paver un échiquier de taille 1×1 dont l'unique case est occupée par la reine est trivial, il n'y a rien à faire:

2. **Étape de récurrence**:
   Soit $k \geq 0$.
   - Hypothèse de récurrence: $P(k)$: « l'échiquier $2^k \times 2^k$ dont une case est occupée par la reine est pavable par des triominos en forme de L »$\equiv$ **vrai**.
   - Objectif: $P(k+1)$: « l'échiquier $2^{k+1} \times 2^{k+1}$ dont une case est occupée par la reine est pavable par des triominos en forme de L » $\equiv$ **vrai**.
   - Argumentation: on considère un échiquier $2^{k+1} \times 2^{k+1}$ dont une case est occupée par la reine. Comme la hauteur et la largeur sont des nombres pairs, on peut diviser l'échiquier en quatre échiquiers de taille $2^k \times 2^k$. Sans perte de généralité, on suppose que la reine est située dans le quart supérieur gauche. On positionne un triomino en forme de L de manière à ce qu'il couvre exactement une case dans chacun des trois échiquiers restants.



   Il reste à montrer qu'il est possible de paver les 4 échiquiers $2^k \times 2^k$ suivants avec des triominos en forme de L.



Chacun de ces quatre échiquiers a exactement une case qui est occupée. On a donc que, par **hypothèse de récurrence**, ces quatre échiquiers peuvent être pavés par des triominos en forme de L, ce qui conclut la preuve. □

Rép. 7.2 (a) n^2

(b) On définit $P(n)$: « $\sum_{i=1}^{n}(2i-1) = n^2$ » et on montre que $\forall n \geq 1$, $P(n) \equiv$ **vrai**.

1. **Cas de base**:
   Pour $n = 1$, on a $P(1) \equiv$ **vrai**, car $\sum_{i=1}^{1}(2i-1) = 1 = 1^2$.
2. **Étape de récurrence**:
   Soit $k \geq 1$.
   - Hypothèse de récurrence: $P(k)$: « $\sum_{i=1}^{k}(2i-1) = k^2$ » ≡ **vrai**.
   - Objectif: $P(k+1)$: « $\sum_{i=1}^{k+1}(2i-1) = (k+1)^2$ » ≡ **vrai**.
   - Calculs:

$$\begin{aligned}\sum_{i=1}^{k+1}(2i-1) &= \sum_{i=1}^{k}(2i-1) + (2(k+1)-1)\\ &= k^2 + (2(k+1)-1) && \text{(par hyp. de réc.)}\\ &= k^2 + 2k + 1\\ &= (k+1)^2\end{aligned}$$

□

Rép. 7.3 Soit $f(n) = 2n^2 + 2n + 5$. On définit la fonction propositionnelle $P(n)$: « $b_n = f(n)$ » et on montre que $\forall n \in \mathbb{N}, P(n) \equiv$ **vrai**.

1. **Cas de base**:
   $P(0)$ est l'affirmation « $b_0 = f(0)$ ».
   Par définition de la suite b, on a que $b_0 = 5$. Par ailleurs, $f(0) = 2\cdot 0^2 + 2\cdot 0 + 5 = 5$. Donc $P(0) \equiv$ **vrai**.
2. **Étape de récurrence**:
   - Hypothèse de récurrence: $P(k)$: « $b_k = f(k)$ » ≡ **vrai**. On suppose donc que $b_k = 2k^2 + 2k + 5$.
   - Objectif: $P(k+1)$: « $b_{k+1} = f(k+1)$ » ≡ **vrai**.
   - Calculs:

$$\begin{aligned}f(k+1) &= 2(k+1)^2 + 2(k+1) + 5 && \text{(par définition de } f\text{)}\\ &= 2(k^2 + 2k + 1) + 2(k+1) + 5 && \text{(par dév. algébrique)}\\ &= 2k^2 + 6k + 9.\end{aligned}$$

et

$$\begin{aligned}b_{k+1} &= b_k + 4(k+1) && \text{(par la relation de récurrence qui définit la suite } b\text{)}\\ &= (2k^2 + 2k + 5) + 4(k+1) && \text{(par l'hypothèse de récurrence)}\\ &= 2k^2 + 6k + 9.\end{aligned}$$

□

Rép. 7.4 Soit $g(n) = 3^{n+1} - 1$. On définit la fonction propositionnelle $P(n)$: « $f(n) = g(n)$ » et on montre que $\forall n \in \mathbb{N}, P(n) \equiv$ **vrai**.

1. **Cas de base**:
   $P(0)$ est l'affirmation « $f(0) = g(0)$ ».
   Par définition de la fonction f, on a que $f(0) = 2$. Par ailleurs, $g(0) = 3 - 1 = 2$. Donc $P(0) \equiv$ **vrai**.
2. **Étape de récurrence**:
   Soit $k \geq 0$.
   - Hypothèse de récurrence: $P(k)$ « $f(k) = g(k)$ » $\equiv$ **vrai**. On suppose donc que $f(k) = 3^{k+1} - 1$.
   - Objectif: $P(k+1)$ « $f(k+1) = g(k+1)$ » $\equiv$ **vrai**. Il faut donc vérifier que $f(k+1) = 3^{k+2} - 1$.
   - Calculs:

$$\begin{aligned} f(k+1) &= 3f(k) + 2 && \text{(par la relation de récurrence qui définit la fonction } f\text{)} \\ &= 3 \cdot (3^{k+1} - 1) + 2 && \text{(par l'hypothèse de récurrence)} \\ &= 3 \cdot 3^{k+1} - 3 + 2 \\ &= 3^{k+2} - 1 \end{aligned}$$

□

Rép. 7.5 (a) Soit $k \geq 0$.
- Hypothèse: On suppose $P(k)$, il existe $a \in \mathbb{Z}$ tel que $k^3 + 2k + 1 = 3a$.
- Objectif: $P(k+1) \equiv$ **vrai**, ce qui revient à montrer qu'il existe $a' \in Z$ tel que $(k+1)^3 + 2(k+1) + 1 = 3a'$.
- Calculs:

$$\begin{aligned} (k+1)^3 + 2(k+1) + 1 &= k^3 + 2k + 1 + 3k^2 + 3k + 3 \\ &= 3a + 3k^2 + 3k + 3 && \text{(par hypothèse)} \\ &= 3(a + k^2 + k + 1) \\ &= 3a' && \text{(avec } a' = a + k^2 + k + 1 \in \mathbb{Z}\text{)} \end{aligned}$$

□

(b) En (a), on a montré l'étape de récurrence « $P(k) \rightarrow P(k+1)$ ». Cela veut dire que si, pour un certain entier n_0, $P(n_0) \equiv$ **vrai**, alors l'énoncé est vrai pour tous les entiers suivants. Hors, il n'existe aucun entier n pour lequel $P(n)$ est vrai, ainsi cette fonction propositionnelle est fausse pour tout $n \in \mathbb{Z}$.
Conclusion: sans cas de base, l'étape récursive n'a aucune valeur.

Rép. 7.6 On définit la fonction propositionnelle $P(n)$: « Si $f(x) = x^n$ alors $f'(x) = nx^{n-1}$ » et on montre que $\forall n \geq 1, P(n) \equiv$ **vrai**.

1. **Cas de base**:
   $P(1)$ est l'affirmation « Si $f(x) = x^1$ alors $f'(x) = 1x^0 = 1$ ».
   Donc $P(1) \equiv$ **vrai**.
2. **Étape de récurrence**:
   Soit $k \geq 1$.
   - Hypothèse: $P(k)$: « Si $f(x) = x^k$ alors $f'(x) = kx^{k-1}$ » $\equiv$ **vrai**
   - Objectif: $P(k+1)$ « Si $f(x) = x^{k+1}$, alors $f'(x) = (k+1)x^k$ » $\equiv$ **vrai**.
   - Calculs:

$$\begin{aligned} \left[x^{k+1}\right]' &= \left[x^k x\right]' \\ &= \left[x^k\right]' \cdot x + x^k \cdot [x]' && \text{(par la règle de la dérivée d'un produit de fonctions)} \\ &= kx^{k-1} \cdot x + x^k \cdot 1 && \text{(par hypothèse de récurrence)} \\ &= kx^k + x^k \\ &= (k+1)x^k \end{aligned}$$

□

Rép. 7.7 On définit $P(n)$: « La dérivée $n^{\text{ième}}$ de $f(x) = \ln(x)$ est $f^{(n)}(x) = \dfrac{(-1)^{(n-1)}(n-1)!}{x^n}$ » et on montre que $\forall n \geq 1, P(n) \equiv$ **vrai**.

1. **Cas de base**:
   $P(1)$ est l'affirmation « Si $f(x) = \ln(x)$ alors $f'(x) = \dfrac{(-1)^{(1-1)}(1-1)!}{x^1} = \dfrac{1}{x}$ ».
   Donc $P(1) \equiv$ **vrai**.
2. **Étape de récurrence**:
   Soit $k \geq 1$.
   - Hypothèse: $P(k)$: « La dérivée $k^{\text{ième}}$ de $f(x) = \ln(x)$ est $f^{(k)}(x) = \dfrac{(-1)^{(k-1)}(k-1)!}{x^k}$ » $\equiv$ **vrai**.
   - Objectif: $P(k+1)$ « La dérivée $(k+1)^{\text{ième}}$ de $f(x) = \ln(x)$ est $f^{(k+1)}(x) = \dfrac{(-1)^k k!}{x^{k+1}}$ » $\equiv$ **vrai**.
   - Calculs:

$$\begin{aligned}
f^{(k+1)}(x) = \left[f^{(k)}(x)\right]' &= \left[\frac{(-1)^{(k-1)}(k-1)!}{x^k}\right]' && \text{(par hypothèse de récurrence)}\\
&= (-1)^{(k-1)}(k-1)!\left[\frac{1}{x^k}\right]' && \text{(par les règles de dérivation)}\\
&= (-1)^{(k-1)}(k-1)!(-kx^{-k-1})\\
&= (-1)^k k!\, x^{-(k+1)} && \text{(puisque } (k-1)!k = k!)\\
&= \frac{(-1)^k k!}{x^{k+1}}
\end{aligned}$$

□

Rép. 7.8 (a) On définit $P(n)$: « $\sum_{i=0}^{n} f_i = f_{n+2} - 1$ » et on montre que $\forall n \in \mathbb{N}, P(n) \equiv$ **vrai**.

1. **Cas de base**:
   $P(0)$ est l'affirmation « $f_0 = f_2 - 1$ ». Comme par définition $f_0 = 0$ et $f_2 = 1$, alors $P(1) \equiv$ **vrai**.
2. **Étape de récurrence**:
   Soit $k \geq 0$.
   - Hypothèse: $P(k)$: « $\sum_{i=0}^{k} f_i = f_{k+2} - 1$ » $\equiv$ **vrai**.
   - Objectif: $P(k+1)$: « $\sum_{i=0}^{k+1} f_i = f_{k+3} - 1$ » $\equiv$ **vrai**..
   - Calculs:

$$\begin{aligned}
\sum_{i=0}^{k+1} f_i &= \sum_{i=0}^{k} f_i + f_{k+1}\\
&= f_{k+2} - 1 + f_{k+1} && \text{(par hypothèse de récurrence)}\\
&= f_{k+3} - 1 && \text{(par définition des nombres de Fibonacci}
\end{aligned}$$

□

(b) On définit $P(n)$: « $\sum_{i=0}^{n} f_i^2 = f_n \cdot f_{n+1}$ » et on montre que $\forall n \in \mathbb{N}, P(n) \equiv$ **vrai**.

1. **Cas de base**:
   $P(0)$ est l'affirmation « $f_0^2 = f_0 \cdot f_1$ ». Comme par définition $f_0 = 0$ et $f_1 = 1$, alors $P(0) \equiv$ **vrai**.
2. **Étape de récurrence**:
   Soit $k \geq 0$.
   - Hypothèse: $P(k)$: « $\sum_{i=0}^{k} f_i^2 = f_k \cdot f_{k+1}$ » $\equiv$ **vrai**.
   - Objectif: $P(k+1)$: « $\sum_{i=0}^{k+1} f_i^2 = f_{k+1} \cdot f_{k+2}$ » $\equiv$ **vrai**.

– Calculs :

$$\begin{aligned}
\sum_{i=0}^{k+1} f_i^2 &= \sum_{i=0}^{k} f_i^2 + f_{k+1}^2 \\
&= f_k \cdot f_{k+1} + f_{k+1}^2 && \text{(par hypothèse de récurrence)} \\
&= f_{k+1}(f_k + f_{k+1}) && \text{(par mise en évidence)} \\
&= f_{k+1} f_{k+2} && \text{(par définition des nombres de Fibonacci)}
\end{aligned}$$

□

Rép. 7.9 On définit $P(n)$: « $a_n = n2^n$ » et on veut montrer que $\forall n \in \mathbb{N}$, $P(n) \equiv$ **vrai**.

1. **Cas de base** :
   $P(0) \equiv$ **vrai** car $a_0 = 0$ (par définition) et $0 \cdot 2^0 = 0$.
   $P(1) \equiv$ **vrai** car $a_1 = 2$ (par définition) et $1 \cdot 2^1 = 2$.
2. **Étape de récurrence** :
   Soit $k \geq 1$.
   – Hypothèse de récurrence : $P(0) \wedge P(1) \wedge \cdots \wedge P(k) \equiv$ **vrai**. En particulier, on suppose que $P(k-1) \equiv$ **vrai** et donc que $a_{k-1} = (k-1)2^{k-1}$, $P(k) \equiv$ **vrai** et donc que $a_k = k2^k$.
   – Objectif : $P(k+1)$ « $a_{k+1} = (k+1)2^{k+1}$ » $\equiv$ **vrai**.
   – Calculs :

$$\begin{aligned}
a_{k+1} &= 4a_k - 4a_{k-1} && \text{(par définition)} \\
&= 4\,k2^k - 4\,(k-1)2^{k-1} && \text{(par hyp. de réc.)} \\
&= 2\,k2^{k+1} - (k-1)2^{k+1} \\
&= (2k - k + 1)\,2^{k+1} \\
&= (k+1)\,2^{k+1}
\end{aligned}$$

□

Rép. 7.10 On définit $P(n)$: « $a_n = 3^n - 2^{n+1} + 1$ » et on veut montrer que $\forall n \in \mathbb{N}$, $P(n) \equiv$ **vrai**.

1. **Cas de base** :
   $P(0) \equiv$ **vrai** car $a_0 = 0$ (par définition) et $3^0 - 2^1 + 1 = 0$.
   $P(1) \equiv$ **vrai** car $a_1 = 0$ (par définition) et $3^1 - 2^2 + 1 = 0$.
   $P(2) \equiv$ **vrai** car $a_2 = 2$ (par définition) et $3^2 - 2^3 + 1 = 2$.
2. **Étape de récurrence** :
   Soit $k \geq 2$.
   – Hypothèse de récurrence : $P(0) \wedge P(1) \wedge \cdots \wedge P(k) \equiv$ **vrai**. En particulier, on suppose que $P(k-2) \equiv$ **vrai** et donc que $a_{k-2} = 3^{k-2} - 2^{k-1} + 1$,
   $P(k-1) \equiv$ **vrai** et donc que $a_{k-1} = 3^{k-1} - 2^k + 1$,
   $P(k) \equiv$ **vrai** et donc que $a_k = 3^k - 2^{k+1} + 1$.
   – Objectif : $P(k+1)$: « $a_{k+1} = 3^{k+1} - 2^{k+2} + 1$ » $\equiv$ **vrai**.
   – Calculs :

$$\begin{aligned}
a_{k+1} &= 6a_k - 11a_{k-1} + 6a_{k-2} && \text{(par définition)} \\
&= 6\left(3^k - 2^{k+1} + 1\right) - 11\left(3^{k-1} - 2^k + 1\right) + 6\left(3^{k-2} - 2^{k-1} + 1\right) && \text{(par hyp. de réc.)} \\
&= \left(6 \cdot 3^k - 11 \cdot 3^{k-1} + 6 \cdot 3^{k-2}\right) - \left(6 \cdot 2^{k+1} - 11 \cdot 2^k + 6 \cdot 2^{k-1}\right) + (6 - 11 + 6) \\
&= \left(2 \cdot 3^{k+1} - 11 \cdot 3^{k-1} + 2 \cdot 3^{k-1}\right) - \left(3 \cdot 2^{k+2} - 11 \cdot 2^k + 3 \cdot 2^k\right) + 1 \\
&= \left(2 \cdot 3^{k+1} - 9 \cdot 3^{k-1}\right) - \left(3 \cdot 2^{k+2} - 8 \cdot 2^k\right) + 1 \\
&= \left(2 \cdot 3^{k+1} - 3^{k+1}\right) - \left(3 \cdot 2^{k+2} - 2 \cdot 2^{k+2}\right) + 1 \\
&= 3^{k+1} - 2^{k+2} + 1
\end{aligned}$$

□

Rép. 7.11 On définit $P(n)$: « Il existe $a, b \in \mathbb{N}$ tels que $3a+7b=n$ » et on veut montrer que $\forall n \geq 12$, $P(n) \equiv$ **vrai**.

1. **Cas de base**: (ici $n_0 = 12$ et $j = 2$)
   $P(12) \equiv$ **vrai** avec $a = 4$ et, $b = 0$ car $12 = 4 \cdot 3 + 0 \cdot 7$,
   $P(13) \equiv$ **vrai** avec $a = 2$ et, $b = 1$ car $13 = 2 \cdot 3 + 1 \cdot 7$,
   $P(14) \equiv$ **vrai** avec $a = 0$ et, $b = 2$ car $14 = 0 \cdot 3 + 2 \cdot 7$.

2. **Étape de récurrence**:
   Soit $k \geq 14$.
   - Hypothèse de récurrence: $P(12) \wedge P(13) \wedge \cdots \wedge P(k-1) \wedge P(k) \equiv$ **vrai**. En particulier, on a que $P(k-2) \equiv$ **vrai**. On a donc qu'il existe $a, b \in \mathbb{N}$ tels que $3a+7b = k-2$.
   - Objectif: $P(k+1)$: « Il existe $a_2, b_2 \in \mathbb{N}$ tels que $k+1 = 3a_2 + 7b_2$ » $\equiv$ **vrai**.
   - Calculs:

$$\begin{aligned} k+1 &= (k-2)+3 \\ &= 3a+7b+3 && \text{(par hyp. de réc.)} \\ &= 3(a+1)+7b \\ &= 3a_2+7b_2. && \text{(avec } a_2 = a+1 \in \mathbb{N} \text{ et } b_2 = b \in \mathbb{N}\text{)} \end{aligned}$$

□

Rép. 7.12 On définit $P(n)$: « Il existe $a, b \in \mathbb{N}$ tels que $5a+7b=n$ » et on veut montrer que $\forall n \geq 24$, $P(n) \equiv$ **vrai**.

1. **Cas de base:** (ici $n_0 = 24$ et $j = 4$)
   $P(24) \equiv$ **vrai** avec $a = 2$ et, $b = 2$ car $24 = 2 \cdot 5 + 2 \cdot 7$,
   $P(25) \equiv$ **vrai** avec $a = 5$ et, $b = 0$ car $25 = 5 \cdot 5 + 0 \cdot 7$,
   $P(26) \equiv$ **vrai** avec $a = 1$ et, $b = 3$ car $26 = 1 \cdot 5 + 3 \cdot 7$.
   $P(27) \equiv$ **vrai** avec $a = 4$ et, $b = 1$ car $27 = 4 \cdot 5 + 1 \cdot 7$.
   $P(28) \equiv$ **vrai** avec $a = 0$ et, $b = 4$ car $28 = 0 \cdot 5 + 4 \cdot 7$.

2. **Étape de récurrence:**
   Soit $k \geq 28$.
   - Hypothèse de récurrence: $P(24) \wedge P(25) \wedge \cdots \wedge P(k-1) \wedge P(k) \equiv$ **vrai**. En particulier, on a que $P(k-4) \equiv$ **vrai**. On a donc qu'il existe $a, b \in \mathbb{N}$ tels que $5a+7b = k-4$.
   - Objectif: $P(k+1)$: « Il existe $a_2, b_2 \in \mathbb{N}$ tels que $k+1 = 5a_2 + 7b_2$ » $\equiv$ **vrai**.
   - Calculs:

$$\begin{aligned} k+1 &= (k-4)+5 \\ &= 5a+7b+5 && \text{(par hyp. de réc.)} \\ &= 5(a+1)+7b \\ &= 5a_2+7b_2. && \text{(avec } a_2 = a+1 \in \mathbb{N} \text{ et } b_2 = b \in \mathbb{N}\text{)} \end{aligned}$$

□

Rép. 7.13 (a)

```
1: fonction factorielle(n : Entier non négatif)
2:     si n == 0 alors
3:         retourner 1
4:     sinon
5:         retourner n · factorielle(n − 1)
6:     fin si
7: fin fonction
```

(b) On définit $P(n)$: « `factorielle`(n) retourne $n!$ » et on montre que $\forall n \in \mathbb{N}, P(n) \equiv$ **vrai**.

1. **Cas de base**:
   $P(0) \equiv$ **vrai** comme en analysant le code de la fonction on vérifie que pour $n = 0$, la fonction retourne 1.

2. **Étape de récurrence**:
   Soit $k \geq 0$.
   - Hypothèse de récurrence: $P(k)$: « un appel à `factorielle`(k) retourne $k!$ »≡ **vrai**.
   - Objectif: $P(k+1)$: « un appel à `factorielle`($k+1$) retourne $(k+1)!$ » ≡ **vrai**.
   - Argumentation: comme $k \geq 0$, on a que $k+1 > 0$ et donc la condition du **si** est fausse et on passe à la ligne suivante: **sinon**. Par hypothèse de récurrence, l'appel récursif retourne $k!$ et l'appel à `factorielle`($k+1$) retourne $(k+1) \cdot k! = (k+1)!$. □

(c) Soit $f(n)$ le nombre de multiplications effectuées par un appel à `factorielle`(n), on a

$$f(n) = \begin{cases} 0 & \text{si } n = 0, \\ f(n-1)+1 & \text{si } n \geq 1. \end{cases}$$

En itérant, $f(n) = f(n-1)+1 = f(n-2)+2 = f(n-3)+3 = \cdots = f(0)+n$. On conclut que $f(n) = n$.

Rép. 7.14 (a) `mystère`(n) = n^3.

(b) On définit $P(n)$: « `mystère`(n) retourne n^3 » et on veut montrer que $\forall n \geq 0, P(n) \equiv$ **vrai**.

1. **Cas de base**:
   $P(0) \equiv$ **vrai** comme en analysant le code de la fonction on vérifie que pour $n = 0$, la fonction retourne 0.
2. **Étape de récurrence**:
   Soit $k \geq 0$,
   - Hypothèse de récurrence: $P(k)$: « un appel à `mystère`(k) retourne k^3 »≡ **vrai**.
   - Objectif: $P(k+1)$: « un appel à `mystère`($k+1$) retourne $(k+1)^3$ » ≡ **vrai**.
   - Argumentation: comme $k \geq 0$, on a que $k+1 > 0$ et donc la condition du **si** est fausse et on passe à la ligne suivante: **sinon**. Par hypothèse de récurrence, l'appel récursif retourne k^3 et l'appel à `mystère`($k+1$) retourne $k^3 + 3(k+1)^2 - 3(k+1) + 1 = (k+1)^3$. □

(c) $f(0) = 0$, $f(n) = f(n-1)+3$, $n \geq 1$.

(d) $f(n) = 3n \in O(n)$, linéaire.

Rép. 7.15 (a)

```
1: fonction Fibo(n: Entier non négatif)
2:     si n == 0 alors
3:         retourner 0
4:     sinon si n == 1 alors
5:         retourner 1
6:     sinon
7:         retourner Fibo(n − 1) + Fibo(n − 2)
8:     fin si
9: fin fonction
```

(b) On définit $P(n)$: « `Fibo`(n) retourne f_n » et on veut montrer que $\forall n \geq 0, P(n) \equiv$ **vrai**.

1. **Cas de base**:
   $P(0) \equiv$ **vrai**, en analysant le code de la fonction on vérifie que pour $n = 0$, la fonction retourne 0.
   $P(1) \equiv$ **vrai**, en analysant le code de la fonction on vérifie que pour $n = 1$, la fonction retourne 1.
2. **Étape de récurrence**:
   Soit $k \geq 1$.
   - Hypothèse de récurrence: $P(0) \wedge P(1) \wedge \cdots \wedge P(k) \equiv$ **vrai**, c'est-à-dire qu'un appel à `Fibo`(n) avec $n \in \{0, 1, \ldots, k\}$ retourne f_n.
   - Objectif: $P(k+1)$: « un appel à `Fibo`($k+1$) retourne f_{k+1} » ≡ **vrai**.
   - Argumentation: comme $k \geq 1$, on a que $k+1 \geq 2$ et donc la fonction effectue d'abord un appel à `Fibo`($k-1$). Par hypothèse de récurrence, cet appel retourne f_{k-1}. Ensuite, on effectue un appel à `Fibo`(k) qui, par hypothèse de récurrence, retourne f_k. Finalement, la fonction retourne $f_{k-1} + f_k$ hors, par la définition de la suite de Fibonacci, $f_{k-1} + f_k = f_{k+1}$. □

(c) On définit $P(n)$: « $g(n) = f_{n+1} - 1$ » et on montre que $\forall n \geq 0, P(n) \equiv$ **vrai**.

1. **Cas de base**:
   $P(0) \equiv$ **vrai**, par l'algorithme il est évident que $g(0) = 0$ ce qui est bien égal à $f_1 - 1 = 0$.
   $P(1) \equiv$ **vrai**, encore une fois, par l'algorithme il est évident que $g(1) = 0$, et on a $f_2 - 1 = 0$.
2. **Étape de récurrence**:
   Soit $k \geq 1$.
   - Hypothèse de récurrence: $P(0) \wedge P(1) \wedge \cdots \wedge P(k) \equiv$ **vrai** et donc, en particulier, on accepte les hypothèses $g(k-1) = f_k - 1$ et $g(k) = f_{(k+1)} - 1$.
   - Objectif: $P(k+1)$: « $g(k+1) = f_{(k+2)} - 1$ » $\equiv$ **vrai**.
   - Argumentation: Comme $k \geq 1$, on a que $k+1 \geq 2$ et donc lors d'un appel à `Fibo`$(k+1)$, les conditions des deux **si** sont fausses et le **retourner** de la ligne 7 est effectué. Ainsi, le nombre d'additions est donné par:

$$\begin{aligned} g(k+1) &= g(k) + g(k-1) + 1 && \text{(par l'algorithme)} \\ &= (f_{(k+1)} - 1) + (f_k - 1) + 1 && \text{(par hypothèse de récurrence)} \\ &= f_{(k+2)} - 1. && \text{(par déf. des nombres de Fibonacci)} \end{aligned}$$

□

(d)
```
1: fonction Fibo(n: Entier non négatif)
2:     a := 0
3:     b := 1
4:     pour i de 0 à n − 1 faire
5:         c := a + b
6:         a := b
7:         b := c
8:     fin pour
9:     retourner a
10: fin fonction
```

(e) $h(n) = n$.

(f) La version itérative est beaucoup plus efficace. En effet, on a que, $h(n) \in O(g(n))$ mais $g(n) \notin O(h(n))$.

(g) Malheureusement, nous ne vous donnerons pas la réponse ici. Voici quand même deux indices:
- Considérez le produit matriciel suivant:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} 0 & f_n \\ 0 & f_{n+1} \end{bmatrix} = \begin{bmatrix} 0 & af_n + bf_{n+1} \\ 0 & cf_n + df_{n+1} \end{bmatrix}$$

- Inspirez-vous de l'algorithme de l'exponentiation modulaire vu au Cours 5 afin de calculer efficacement de grandes puissances.

Chapitre 8

Rép. 8.1 (a) Il a $2^8 = 256$ trains de bits de longueur 8.
Il y a deux choix pour le premier bit **ET** deux choix pour le deuxième bit **ET** deux choix pour le troisième bit, et ainsi de suite jusqu'au huitième. On conclut en utilisant le *principe du produit*. Visuellement:

Bit #1	Bit #2	Bit #3	Bit #4	Bit #5	Bit #6	Bit #7	Bit #8
0 ou 1	0 ou 1	0 ou 1	0 ou 1	0 ou 1	0 ou 1	0 ou 1	0 ou 1
2 choix	2 choix	2 choix	2 choix	2 choix	2 choix	2 choix	2 choix

(b) Il y a $2^3 = 8$ trains de bits de longueur 8 qui commencent par 01 et terminent par 101. Par le *principe du produit*:

#1	#2	#3	#4	#5	#6	#7	#8
0	1	0 ou 1	0 ou 1	0 ou 1	1	0	1
1 choix	1 choix	2 choix	2 choix	2 choix	1 choix	1 choix	1 choix

(c) Il y a $10^3 \cdot 26^3 = 17\,576\,000$ plaques d'immatriculation de 3 chiffres suivis de 3 lettres. Par le *principe du produit*:

#1	#2	#3	#4	#5	#6
0 à 9	0 à 9	0 à 9	A à Z	A à Z	A à Z
10 choix	10 choix	10 choix	26 choix	26 choix	26 choix

Rép. 8.2 Il y a 2 684 483 063 360 mots de passe possibles.
On commence par évaluer le nombre de mots de passe valides de longueur n puis on utilise le *principe de la somme* pour les cas $n = 6$, $n = 7$ et $n = 8$.

Comme chaque caractère est soit un chiffre, soit une lettre majuscule, par le *principe de la somme* il y a 10+26 = 36 possibilités pour chaque caractère.

Ensuite, il faut tenir compte de la contrainte voulant que chaque mot de passe contienne au moins un chiffre. Le plus simple est de compter tous les mots de passe sans tenir compte de la contrainte et ensuite soustraire tout ce qui a été compté en trop.

#1	#2	#3		#n
0 à 9 ou A à Z	0 à 9 ou A à Z	0 à 9 ou A à Z	⋯	0 à 9 ou A à Z
36 choix	36 choix	36 choix		36 choix

Sans tenir compte de la contrainte, il y a 36^n mots de passe possibles. À ce nombre, il faut soustraire tous les mots de passe qui ne contiennent aucun chiffre.

#1	#2	#3		#n
A à Z	A à Z	A à Z	⋯	A à Z
26 choix	26 choix	26 choix		26 choix

Il y a donc 26^n mots de passe invalides à soustraire. Le nombre de mots de passe valides de longueur n est donc $36^n - 26^n$. En sommant pour $n = 6, 7$ et 8, on obtient: $\sum_{i=6}^{8}(36^i - 26^i)$.

Rép. 8.3 (a) Réponse: 56 étudiants.
On définit les ensembles suivants:
- U est l'ensemble des étudiants inscrits en génie logiciel,
- A_1 est l'ensemble des étudiants en génie logiciel inscrits en MAT210,
- A_2 est l'ensemble des étudiants en génie logiciel inscrits en MAT265.

On cherche à déterminer $|U| - |A_1 \cup A_2|$. Par le *principe d'inclusion-exclusion*,

$$|U| - |A_1 \cup A_2| = |U| - (|A_1| + |A_2| - |A_1 \cap A_2|) = 300 - (143 + 122 - 21) = 56.$$

(b) Réponse: 88 trains de bits.
On pose:
- A_1 l'ensemble des trains de bits de longueur 8 qui commencent par 01.
- A_2 l'ensemble des trains de bits de longueur 8 qui terminent par 101.

On a alors que

- $|A_1| = 2^6$ car: 0 1 $\underbrace{\_\_\_\_\_}_{\text{6 bits}}$,
- $|A_2| = 2^5$ car: $\underbrace{\_\_\_\_\_}_{\text{5 bits}}$ 1 0 1,
- $|A_1 \cap A_2| = 2^3$ car: 0 1 $\underbrace{\_\_\_\_\_}_{\text{3 bits}}$ 1 0 1.

Par le *principe d'inclusion-exclusion*: $|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2| = 2^6 + 2^5 - 2^3$.

Rép. 8.4 (a) Il y a 5040 dispositions.
Les individus étant considérés comme distincts, le nombre de dispositions est donné par le nombre de permutations de 7 objets: $7! = 5040$.

(b) Il y a 1440 dispositions.
On peut considérer d'un bloc les deux hommes placés consécutivement dans la file. On compte ainsi les permutations formées par ce bloc et les 5 femmes. Il s'agit du nombre de permutations de 6 objets $6! = 720$.
Attentions: pour chacune de ces permutations, les deux hommes peuvent être disposés différemment à l'intérieur du bloc. Il y a $2!$ ordres possibles à l'intérieur du bloc. Ainsi, le nombre total de dispositions est $6! \cdot 2! = 1440$.

(c) Il y a 3600 dispositions.
Il suffit de considérer le nombre total de dispositions (calculé en (a)) et y soustraire le nombre de dispositions où les deux hommes se suivent (calculé en (b)).

(d) Il y a 120 permutations.
On compte les permutations des 5 objets: "ABC", "D", "E", "F" et "G". Il y en a $5! = 120$.

Rép. 8.5 (a) Il faut au moins 26 étudiants.
Soit n le nombre d'étudiants et $k = 5$ le nombre de notes possibles. On veut *ranger* n étudiants dans k *tiroirs* et avoir au moins six étudiants dans le même tiroir.
Par le *principe des tiroirs*, on cherche le plus petit entier n tel que:

$$\left\lceil \frac{n}{5} \right\rceil = 6 \quad \leftrightarrow \quad 5 < \frac{n}{5} \leq 6 \quad \leftrightarrow \quad 25 < n \leq 30.$$

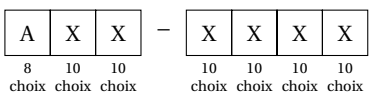
Il doit y avoir au minimum 26 étudiants garantir qu'au moins 6 ont la même note.

(b) Il faut 52729 mots.
Soit n le nombre de mots et k le nombre de suites de trois lettres. On veut *ranger* n mots dans k *tiroirs* et avoir au moins quatre mots dans le même tiroir. Par le *principe des tiroirs*, on cherche le plus petit entier n tel que:

$$\left\lceil \frac{n}{k} \right\rceil = 4 \quad \leftrightarrow \quad 3 < \frac{n}{k} \leq 4 \quad \leftrightarrow \quad 3k < n \leq 4k.$$

Par le *principe du produit*, le nombre de tiroirs est $k = 26^3$. Il faut donc $n = 3k + 1 = 3 \cdot 26^3 + 1 = 52729$ mots.

(c) Il faut au minimum 3 indicatifs régionaux.
Un numéro de téléphone est composé d'un indicatif régional (les trois premiers chiffres) et d'un suffixe (les 7 chiffres restants). Soit $n = 17 \cdot 10^6$ le nombre d'abonnés et k le nombre de suffixes possibles. On veut *ranger* n abonnés dans k tiroirs.
Le nombre d'indicatifs régionaux nécessaires est ainsi donné par le nombre d'abonnés *rangés* dans le même tiroir.
Par le `principe du produit`, le nombre de suffixes est $k = 8 \cdot 10^6$.



Par le *principe des tiroirs,* le nombre minimum d'indicatifs régionaux nécessaires est :

$$\left\lceil \frac{17 \cdot 10^6}{8 \cdot 10^6} \right\rceil = 3.$$

Rép. 8.6 (a) Il y a 20 mots.
On compte le nombre d'arrangements de 2 lettres parmi 5, $P(5,2) = 20$.

(b) Il y a 14400 dispositions.
On commence par considérer toutes les permutations possibles des 5 femmes (il y a en 5!). Pour chacune de ces permutations, on ajoute les trois hommes en s'assurant qu'ils soient séparés. On a donc la situation suivante :

$$\bullet\, F_1 \,\bullet\, F_2 \,\bullet\, F_3 \,\bullet\, F_4 \,\bullet\, F_5 \,\bullet$$

Parmi les six •, il faut choisir celui où on place le premier homme, puis celui où on place le deuxième et finalement celui où on place le troisième. On construit donc un arrangement de trois • parmi six et il y a $P(6,3)$ façons de faire.
En multipliant, on obtient $5! \cdot P(6,3) = 14400$.

Rép. 8.7 (a) Il y en a 56.
Parmi les 8 bits, on en *choisit* trois et on leur assigne la valeur 1. L'ordre dans lequel ces trois bits sont choisis n'a pas d'importance puisque les 1 ne sont pas distinguables. On compte donc le nombre de combinaisons de 3 parmi 8 ; il y en a $C(8,3)$. Visuellement, pour construire un tel train de bit, on considère 8 positions et on en choisit 3 où on affecte la valeur 1.

$$\underset{}{\_} \;\; \underset{}{\_} \;\; \underset{\uparrow}{\underline{1}} \;\; \underset{}{\_} \;\; \underset{\uparrow}{\underline{1}} \;\; \underset{}{\_} \;\; \underset{}{\_} \;\; \underset{\uparrow}{\underline{1}}$$

Ensuite, on affecte la valeur 0 à tous les bits qui n'ont pas été choisis. Dans l'exemple ci-dessus, on obtient : 00101001. Cette dernière étape ne change rien au comptage, car il n'y a qu'une seule façon de faire. Il y a donc $C(8,3) = 56$ trains de bits avec exactement 3 occurrences de 1.

(b) Il y en a 93.
Par le raisonnement expliqué en (a), on conclut que le nombre de trains de bits de longueur n possédant exactement k occurrences de 1 est donné par $C(n,k)$. Ainsi,
- il y a exactement $C(8,0)$ trains de bits de longueur 8 avec zéro 1,
- il y a exactement $C(8,1)$ trains de bits de longueur 8 avec un 1,
- il y a exactement $C(8,2)$ trains de bits de longueur 8 avec deux 1,
- il y a exactement $C(8,3)$ trains de bits de longueur 8 avec trois 1.

Par le *principe de la somme*, le nombre de trains de bits avec au plus trois 1 est :

$$\sum_{i=0}^{3} C(8,i) = 93.$$

(c) Il a 13 983 816 façons de choisir 6 numéros parmi 49.
L'ordre des numéros n'a pas d'importance, il s'agit donc du nombre de combinaisons de 6 parmi 49, soit $C(49,6)$.

(d) Il y en a $C(657,4) \cdot C(753,2) = 2\,178\,009\,835\,742\,880$.
On *choisit* 4 étudiants parmi les 657 en génie logiciel **ET** on *choisit* 2 étudiants parmi les 753 en génie électrique. Comme l'ordre dans lequel les étudiants sont choisis n'a pas d'importance,
- il y a $C(657,4)$ façon de choisir 4 étudiants parmi 657,
- il y a $C(753,2)$ façon de choisir 2 étudiants parmi 753.

Il y a donc $C(657,4) \cdot C(753,2)$ façons de former un tel comité.

Rép. 8.8 (a) $C(40,2)\cdot C(300,5) = 15\,274\,613\,296\,800$
(b) $C(40,2)\cdot C(300,3) + C(40,3)\cdot C(300,2) = 3\,918\,096\,000$
(c) $C(340,5) = 36\,760\,655\,568$
(d) $C(340,4) + C(340,5) + C(340,6) + C(340,7) + C(340,8) = 4\,176\,447\,114\,716\,383$

Rép. 8.9 (a) 2520
(b) 1680
(c) 3360

Rép. 8.10 (a) 128
(b) 27
(c) 783
(d) 729
(e) 560
(f) 939

Rép. 8.11 (a) 10^6
(b) $10^4 + 5\cdot 10^5 - 10^3\cdot 5 = 505\,000$
(c) $10^6 - 9^6 = \sum_{i=1}^{6} C(6,i)\,9^{6-i} = 468\,559$
(d) $C(6,1)\cdot C(5,1)\cdot 8^4 = 122\,880$
(e) Permutation de 4 objets (10, 10, 2 et 3), dont deux sont indistinguables : $\frac{4!}{2!1!1!} = 12$
(f) $C(5,3)\cdot 9^2\cdot 10 = 8\,100$
(g) $P(10,6) = 151\,200$
(h) 90 720
(i) 33 600

Rép. 8.12 (a) $b_0 = 0$ et $b_1 = 0$.
(b) On suppose que n est *suffisamment grand* et on considère toutes les chaînes comptées par b_n. Ces chaînes sont forcément de l'une des formes suivantes :

Forme des chaînes	Nombre de chaînes de cette forme
0 | chaîne de lng $n-1$ avec au moins une occurrence de 11	b_{n-1}
2 | chaîne de lng $n-1$ avec au moins une occurrence de 11	b_{n-1}
1 0 | chaîne de lng $n-2$ avec au moins une occurrence de 11	b_{n-2}
1 2 | chaîne de lng $n-2$ avec au moins une occurrence de 11	b_{n-2}
1 1 | chaîne ternaire quelconque de lng $n-2$	3^{n-2}

On en déduit la relation de récurrence $b_0 = 0;\quad b_1 = 0;\quad b_n = 2b_{n-1} + 2b_{n-2} + 3^{n-2}$ pour $n \geq 2$.
(c) En utilisant la commande `seqgen` de Nspire :

$$\text{seqGen}\left(2\cdot b(n-1)+2\cdot b(n-2)+3^{n-2}, n, b, \{0,8\}, \{0,0\}\right)$$
$$\{0,0,1,5,21,79,281,963,3217\}$$

On a donc que $b_8 = 3217$.

Rép. 8.13 (a) $a_0 = 1, a_1 = 3, a_2 = 9, a_n = 2a_{n-1} + 2a_{n-2} + 2a_{n-3}$. Remarque $a_0 = 1$ est obtenu en considérant la chaîne vide. Pour éviter de se casser la tête avec la chaîne vide, on peut commencer avec $a_1 = 3, a_2 = 9$ et ajouter le troisième cas de base : $a_3 = 26$.

(b) 9989792

(c) 69.6%

(d) 295

Rép. 8.14 (a) $a_1 = 3$, $a_2 = 3^2 - 1 = 8$ et $a_3 = 3^3 - 7 = 20$.

(b) On suppose que n est *suffisamment grand* et on considère toutes les chaînes comptées par a_n. Ces chaînes sont forcément de l'une des formes suivantes :

Forme des chaînes	Nombre de chaînes de cette forme
1 | chaîne de lng $n-1$ sans 000 ni 01	a_{n-1}
2 | chaîne de lng $n-1$ sans 000 ni 01	a_{n-1}
0 2 | chaîne de lng $n-2$ sans 000 ni 01	a_{n-2}
0 0 2 | chaîne de lng $n-3$ sans 000 ni 01	a_{n-3}

On en déduit la relation de récurrence $a_1 = 3$; $a_2 = 8$; $a_3 = 20$; $a_n = 2a_{n-1} + a_{n-2} + a_{n-3}$ pour $n \geq 4$.

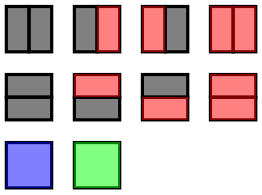
(c) $a_{15} = 1\,492\,163$.

Rép. 8.15 (a) $b_1 = 2, b_n = b_{n-1} + 1$.

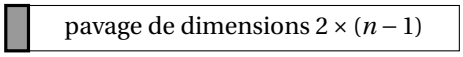
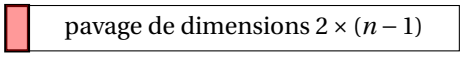
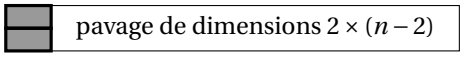
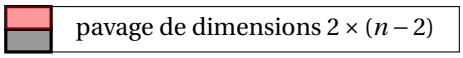
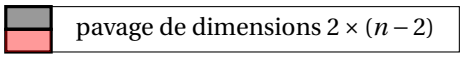
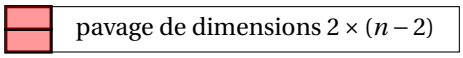
(b) 31

Rép. 8.16 (a) On trouve $a_1 = 2$, comme les pavages possibles d'un trottoir 2×1 sont :

On trouve $a_2 = 10$, comme les pavages possibles d'un trottoir 2×1 sont :



On suppose que n est *suffisamment grand* et on considère tous les pavages comptés par a_n. Ces pavages sont forcément de l'une des formes suivantes :

Forme des pavages	Nombre de pavages de cette forme
 pavage de dimensions $2 \times (n-1)$	a_{n-1}
 pavage de dimensions $2 \times (n-1)$	a_{n-1}
 pavage de dimensions $2 \times (n-2)$	a_{n-2}
 pavage de dimensions $2 \times (n-2)$	a_{n-2}
 pavage de dimensions $2 \times (n-2)$	a_{n-2}
 pavage de dimensions $2 \times (n-2)$	a_{n-2}
 pavage de dimensions $2 \times (n-2)$	a_{n-2}
 pavage de dimensions $2 \times (n-2)$	a_{n-2}

On en déduit la relation de récurrence

$$\begin{cases} a_1 = 2 \\ a_2 = 10 \\ a_n = 2a_{n-1} + 6a_{n-2}, \quad n \geq 3 \end{cases}$$

(b) $a_{15} = 184\,082\,432$.

Rép. 8.17 (a) $a_1 = 1, a_2 = 2, a_4 = 6$.

(b) $a_1 = 1,\ a_2 = 2,\ a_3 = 4,\ a_n = 2a_{n-2} + 2a_{n-3}$ pour $n \geq 4$.

(c) $a_{15} = 3456$

Rédigé par Geneviève Savard, Anouk Bergeron-Brlek et Xavier Provençal,
révisé en août 2021,
Service des enseignements généraux,
École de technologie supérieure.