MAT210 – Quelques algorithmes

Algorithme 2.1 : Développement en base b

Pour calculer la représentation d'un entier $x > 0$ dans une base b quelconque, on utilise l'algorithme suivant afin de calculer les chiffres a_i tels que $(a_n a_{n-1} \cdots a_1 a_0)_b = x$,

1: $i := 0$
2: **tant que** $x > 0$ **faire**
3: $\quad a_i := x \bmod b$
4: $\quad x := \lfloor x/b \rfloor$
5: $\quad i := i + 1$
6: **fin tant que**

Algorithme 2.2 : Exponentiation modulaire b^n mod m

1. Calculer le développement binaire de l'exposant : $n = (a_k \dots a_1 a_0)_2$.
2. Calculer successivement les valeurs modulo m de b, b^2, b^4, b^8, …, b^{2^k}.
3. Multiplier les termes b^{2^i} pour lesquels $a_i = 1$ afin d'obtenir b^n modulo m, en accord avec la loi des exposants

$$b^n = b^{(a_k 2^k + \cdots + a_2 2^2 + a_1 2^1 + a_0)} = b^{a_k 2^k} \cdot \ldots \cdot b^{a_3 2^3} \cdot b^{a_2 2^2} \cdot b^{a_1 2^1} \cdot b^{a_0}.$$

Algorithme 2.3 : Algorithme d'Euclide (calcul de PGCD(a, b))

L'algorithme d'Euclide calcule PGCD$(\boldsymbol{a}, \boldsymbol{b})$ où $a \geq b$

$$\begin{aligned}
a &= bq_1 + r_2 \\
b &= r_2 q_2 + r_3 \\
r_2 &= r_3 q_3 + r_4 \\
&\vdots \\
r_{n-2} &= r_{n-1} q_{n-1} + r_n \qquad \text{le dernier reste non nul, } r_n, \text{ est le pgcd cherché} \\
r_{n-1} &= r_n q_n + \mathbf{0} \qquad \text{Le reste est 0 : fin.}
\end{aligned}$$

Algorithme 1 `chemin`

Entrées: un graphe non orienté G et $\mathbf{v}$ un sommet de G
Sortie: une liste de sommets décrivant un chemin dans G

$C :=$ liste vide
Ajouter $\mathbf{v}$ à C
tant que $\deg(\mathbf{v}) > 0$ **faire**
    Choisir une arête e incidente à $\mathbf{v}$ et appeler $\mathbf{w}$ le sommet à son autre extrémité
    Ajouter $\mathbf{w}$ à la fin de C
    Retirer l'arête e du graphe G
    $\mathbf{v} := \mathbf{w}$
fin tant que
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

$circuit :=$ liste vide
Ajouter un sommet de G à $circuit$
$i := 0$
tant que G possède au moins une arête **faire**
    $v := circuit[i]$
    $C :=$ `chemin`(G, v)
    Insérer la liste C dans $circuit$, à la place de $circuit[i]$
    $i := i + 1$
fin tant que
retourner $circuit$

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, $\mathbf{a} \in V$ le sommet de départ, $\mathbf{z} \in V$ le sommet d'arrivée.
Sortie: coût minimal d'un chemin allant du sommet $\mathbf{a}$ au sommet $\mathbf{z}$

$S := \emptyset$ ▷ sommets dont le chemin optimal est connu
$L :=$ tableau indexé par les sommets ▷ coût du plus court chemin connu
$P :=$ tableau indexé par les sommets ▷ prédécesseur dans le plus court chemin connu
pour tout sommet $\mathbf{v} \in V$ **faire**
    $L(\mathbf{v}) := \infty$ ▷ coût du plus court chemin connu de $\mathbf{a}$ et $\mathbf{v}$
    $P(\mathbf{v}) :=$ `null` ▷ $\mathbf{v}$ n'a pas encore de prédécesseur
fin pour
$L(\mathbf{a}) := 0$ ▷ $\mathbf{a}$ est le point de départ, il est à distance 0 de lui-même
tant que $\mathbf{z} \notin S$ **faire**
    $\mathbf{u} :=$ sommet $\notin S$ avec $L(\mathbf{u})$ minimal
    $S := S \cup \{\mathbf{u}\}$ ▷ le plus court chemin connu de $\mathbf{a}$ à $\mathbf{u}$ est optimal
    pour tout sommets $\mathbf{v}$ tel que $(\mathbf{u}, \mathbf{v}) \in E$ et $\mathbf{v} \notin S$ **faire** ▷ les voisins de $\mathbf{u}$ qui ne sont pas dans S
        si $L(\mathbf{u}) + \omega(\mathbf{u}, \mathbf{v}) < L(\mathbf{v})$ **alors** ▷ s'il est plus court d'aller à $\mathbf{u}$ puis prendre l'arc (ou l'arête) $(\mathbf{u}, \mathbf{v})$
            $P(\mathbf{v}) := \mathbf{u}$ ▷ nouveau meilleur chemin : on accède à $\mathbf{v}$ par le sommet $\mathbf{u}$
            $L(\mathbf{v}) := L(\mathbf{u}) + \omega(\mathbf{u}, \mathbf{v})$ ▷ coût de ce nouveau meilleur chemin
        fin si
    fin pour
fin tant que
retourner $L(\mathbf{z})$