MAT215 – Quelques algorithmes

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 $L(\mathbf{z})$

Algorithme 4 `Kahn`

Entrées: $G = (V, E)$ un graphe orienté et acyclique
Sortie: Un tri topologique des sommets de G

$L :=$ liste vide
tant que il reste des sommets **faire**
    u := un sommet n'ayant aucun arc entrant.
    Ajouter **u** à la fin de la liste L
    Supprimer le sommet **u** et tous les arcs sortants de **u**
fin tant que
retourner L

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

$T :=$ tri topologique des sommets de G
$L :=$ tableau associatif ▷ $L(\mathbf{v})$ est le poids d'un chemin maximal terminant en **v**
pour tout $\mathbf{v} \in V$ dans l'ordre de T **faire**
    si **v** possède au moins un arc entrant **alors**
        $L(\mathbf{v}) := \max\{L(\mathbf{u} + \omega(\mathbf{u}, \mathbf{v})) \mid \mathbf{u} \in V \text{ et } \mathbf{u} \to \mathbf{v} \in E\}$
    sinon
        $L(\mathbf{v}) := 0$
    fin si
fin pour
retourner $\max\{L(\mathbf{v}) \mid \mathbf{v} \in V\}$

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.

$L :=$ tableau indexé par les sommets $\mathbf{v} \in V$ ▷ $L(\mathbf{v})$: poids minimal d'une arête $\mathbf{u}-\mathbf{v}$, avec $\mathbf{u} \in S$
$P :=$ tableau indexé par les sommets $\mathbf{v} \in V$ ▷ $P(\mathbf{v})$: sommet $\mathbf{u} \in S$ minimisant le poids de $\mathbf{u}-\mathbf{v}$
a := un sommet de G (imposé ou choisi au hasard)
$E_T := \emptyset$ ▷ ensemble des arêtes de l'arbre T
$S := \emptyset$ ▷ ensemble des sommets de l'arbre T
pour tout sommet $\mathbf{v} \in V$ **faire**
    $L(\mathbf{v}) := \infty$ ▷ car on ne peut pas encore atteindre **v** à partir d'un sommet de S
    $P(\mathbf{v}) :=$ `null` ▷ **v** n'a pas encore de prédécesseur
fin pour
$L(\mathbf{a}) := 0$ ▷ pour que **a** soit le premier sommet ajouté à S
tant que $S \neq V$ **faire** ▷ tant que l'arbre T ne couvre pas tous les sommets du graphe G
    u := sommet $\notin S$ tel que $L(\mathbf{u})$ est minimal
    $S := S \cup \{\mathbf{u}\}$ ▷ ajouter **u** aux sommets de T
    $E_t := E_t \cup \{P(\mathbf{u})-\mathbf{u}\}$ ▷ ajouter l'arête allant de S à **u** aux arêtes de T (si $\mathbf{u} = \mathbf{a}$, ne rien ajouter car $P(\mathbf{a}) =$ `null`)
    pour tout **v** tel que $\mathbf{u}-\mathbf{v} \in E$ et $\mathbf{v} \notin S$ **faire**
        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 **v** et S
            $L(\mathbf{v}) := \omega(\mathbf{u}, \mathbf{v})$ ▷ on met à jour le poids minimal pour atteindre **v**
            $P(\mathbf{v}) := \mathbf{u}$ ▷ le prédécesseur du sommet **v** dans l'arbre T est **u**
        fin si
    fin pour
fin tant que
retourner $T = (S, E_T)$