CentraleSupélec

Algorithmique et complexité

TD 2/7 – Des chemins et des arbres

Ce TD est l'occasion de s'exercer à la résolution de problèmes à base de graphes et d'approfondir la compréhension des problèmes traités en cours. Notamment le problème du plus court chemin et celui de l'arbre couvrant de poids minimal.

Exercice 1 : Quelques scènes de la vie campagnarde

Un hameau de huit maisons A, B, C, D, E, F, G et H est relié par un réseau routier dont la structure simplifiée est définie par la matrice suivante :

	A	B	C	D	E	F	G	H
A	0	4	∞	∞	∞	8	∞	∞
B	4	0	10	12	∞	9	7	3
C	∞	10	0	3	∞	∞	∞	∞
D	∞	12	3	0	6	∞	∞	10
E	∞	∞	∞	6	0	4	6	3
F	8	9	∞	∞	4	0	4	∞
G	∞	7	∞	∞	6	4	0	3
H	∞	3	∞	10	3	∞	3	0

Dans cette matrice $M(i, j)$ représente le temps de parcours direct de i à j en minutes. Bien sûr, $M(i, i) = 0$ et $M(i, j) = \infty$ signifie qu'il n'y pas de route reliant directement i à j.

Question 1

Faire une représentation graphique dans le plan du graphe non orienté correspondant. Une représentation planaire (sans intersection) est possible.

Question 2

Un médecin de campagne habitant en **G** veut déterminer, en cas d'urgence médicale, les itinéraires les plus rapides le reliant à chacun des autres lieux du hameau. Calculer ces itinéraires : on définira :

- la nature de ce problème,
- la méthode employée,
- les détails de chacune des étapes.

Faites une représentation du sous réseau (graphe partiel) correspondant aux itinéraires obtenus.

Supposons maintenant que les habitants souhaitent renouveler l'alimentation en eau des huit maisons à partir d'un château d'eau situé en **H** par un réseau enfoui qui doit suivre certaines des routes existantes. Chaque nouvelle canalisation a un coût proportionnel à la longueur de la route correspondante, donc à la durée de son parcours. Pour cela, on veut construire un réseau d'alimentation en eau de coût minimal.

Question 3

Quelle est la structure du graphe partiel recherché ? Comment construire un tel réseau ?

Question 4

Dessinez le réseau obtenu. Le lieu où est situé le château d'eau a-t-il un impact (structure, coût) sur le réseau obtenu ?

Question 5

Des intempéries rendent la liaison GH inutilisable. Que deviennent alors les résultats des questions précédentes (question 2 et question 4) ?

Exercice 2 : DAG

L'algorithme des plus courts chemins trouve les plus courts chemins dans un graphe de topologie quelconque. Pour certaines familles de graphes, il est possible d'écrire des algorithmes plus performants (en temps de calcul). Dans cet exercice, nous allons voir comment « améliorer » l'algorithme des plus courts chemins dans le cas des graphes orientés **sans circuits** (en anglais : *Directed Acyclic Graphs* ou DAG).

Considérons $G = (V, E)$ un DAG pondéré.

Question 1

Dessinez un graphe de ce type avec quelques sommets.

Question 2

Dessinez **le même graphe** en positionnant les sommets sur une ligne de manière que tous les arcs vont dans le même sens. Qu'en déduisez-vous ?

Question 3

En utilisant la propriété observée dans la question précédente, proposez une variante de l'algorithme des plus courts chemins qui calcule les plus courts chemins en $\mathcal{O}(|V| + |E|)$. Écrivez votre algorithme en python ou en pseudo-code en décrivant précisément les structures de données utilisées.

Exercice 3 : Arborescence couvrante minimale (partie avancée)

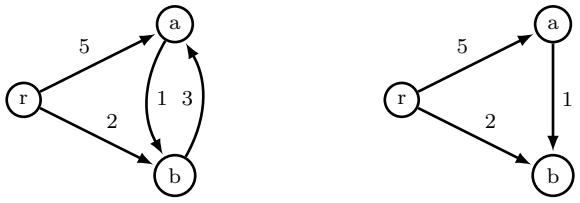
Attention : cet exercice est à réserver pour les groupes avancés.

Considérons un graphe orienté et pondéré G avec une racine r à partir de laquelle tous les sommets sont atteignables. Une arborescence couvrante de poids minimale de G à partir de r est un sous-graphe de G tel que tous les sommets sont toujours atteignables depuis r et que la somme des poids est minimale. Il s'agit de la version « orientée » d'un arbre couvrant de poids minimal vu en cours.

Considérons $G = (V, E)$ un graphe orienté et pondéré avec une racine r.

Question 1

Peut-on appliquer l'algorithme de Kruskal et l'algorithme de Prim sur G pour trouver une arborescence couvrante de poids minimal ? Si oui, expliquez pourquoi. Sinon, donnez un contre-exemple. Vous pouvez justifier votre réponse en utilisant les deux graphes suivants avec la racine r.



Nous allons maintenant étudier un autre algorithme pour trouver une arborescence couvrante de poids minimal.

Les étapes principales sont décrites ci-dessous :

1. supprimez tous les arcs dans le graphe G dont la destination est la racine r
2. pour chaque sommet v autre que la racine, prenez l'arc de poids le plus faible dont la destination est v, noté (u, v) pour lequel $min_u \omega(u, v)$
3. Considérez l'ensemble d'arcs choisis $S = \{(u, v) \mid v \in V \setminus \{r\} \wedge min_u \omega(u, v)\}$. Si S ne contient pas de cycles, alors S est une arborescence couvrante de poids minimal. Sinon, pour chaque cycle dans S, noté C, il faudrait le traiter comme suit.
   (a) pour chaque arc $(i, j)(i \in V \setminus C, j \in C)$ (un arc entrant dans le cycle), modifiez son poids : $\omega'(i, j) = \omega(i, j) - \omega(x(j), j)$, où $\omega(x(j), j)$ est le poids de l'arc dont la destination est j dans C.
   (b) ajoutez l'arc entrant dans le cycle $(i, j) \in E$ dont le poids modifié $\omega'(i, j)$ est le plus faible parmi tous les arcs entrant dans le cycle
   (c) enlevez l'arc dans C dont la destination est j

Question 2

Appliquez ce nouvel algorithme sur les graphes ci-dessus et donnez le résultat pour chaque étape.

Question 3

Maintenant vous comprenez chaque étape de l'algorithme, implémentez-le en python et teste-le à l'aide du code suivant. Votre fonction principale est $acm(root, graphe)$ avec $graphe$ représenté par un dictionnaire d'adjacence :

```
graphe={"r":{"a":1,"b":10},
            "a":{"b":8,"e":2},
            "b":{"d":6},
            "c":{"b":4,"a":7},
            "d":{"c":3},
            "e":{"c":9,"d":11}
    }
h = acm("r",graphe)
assert h == {'r': {'a': 1}, 'c': {}, 'a': {'e': 2, 'b': 8}, 'b': {'d': 6}, 'd': {'c': 3}, 'e': {}}
```