Algorithmique II
Graphes – Arbres

(Correction)

Exercice 1 :

Quizz : vrai ou faux avec justifications ! Dans tout ce qui suit MST ("Minimum Spanning Tree") désignera un arbre couvrant de poids minimum d'un graphe connexe donné.
Donner une réponse justifiée (preuve directe ou preuve algorithmique) aux questions suivantes.

1. Est-il vrai que dans un graphe pondéré g, l'arête de poids minimum si elle est unique est toujours incluse dans le MST de g ?

▷ *Soit a l'arête unique de poids minimum parmi les arêtes de g.*
Si a n'appartient à aucun cycle alors il est forcément dans le MST de g.
Si a appartient à un cycle et qu'il n'est pas dans le MST alors sur ce cycle là en enlevant une arête de poids supérieur à celui de a et en le substituant par a, on arrivera à un MST de poids strictement inférieur à celui trouvé.

En conclusion, une arête de poids minimum est forcément dans tous les MSTs si elle est unique.

2. Qu'en est-il si cette arête n'est pas unique ?

▷ *Dans un graphe ou toutes les arêtes ont le même poids, il est évident qu'une telle arête ne pourra pas nécéssairement appartenir au MST.*

3. Dans un graphe ou toutes les arêtes ont des poids 2 à 2 distincts on a un seul et unique MST.

▷ *VRAI, on utilise l'unicité sur le minimum récursivement.*

4. Soit e une arête de poids maximum sur un des cycles de $G = (S, A)$. Soit $G' = (S, A - \{e\})$. Est-il vrai que $\mathrm{MST}(G') = \mathrm{MST}(G)$?

▷ *Oui, il suffit d'appliquer l'algo à l'envers (vu en cours) qui fixe un cycle et élimine l'arête de poids le plus fort.*

5. Les algorithmes de Boůvka, Kruskal et Prim continuent à trouver les MST sur des graphes avec des arêtes de poids négatifs.

▷ *OUI : preuve en rajoutant un poids constant positif sur toutes ces arêtes négatives on retrouvera les mêmes arbres.*

Exercice 2 :

Ecrire un algorithme qui trouve l'arbre couvrant de poids *maximum* d'un graphe pondéré. Quelle est sa complexité ?

▷ *On transforme le graphe en "flippant" les poids des arêtes (positive → négative). Puis on applique Kruskal de complexité $O(|A| \log |A|)$.*

Exercice 3 :

Etant donné un graphe pondéré g et un ensemble d'arêtes $\mathcal{E} = \{e_1, e_2, \cdots\}$ de g, écrire un algorithme qui calcule le MST de g incluant forcément $\mathcal{E}$.

▷ *On peut par exemple utiliser l'algorithme renversé. On fixe un cycle, on enlève l'arête de poids minimum de ce cycle qui n'est pas dans $\mathcal{E}$. On boucle tant qu'il y aura des cycles.*

Exercice 4 :

Etant donné un graphe pondéré g, on sait calculer son MST via différents algorithmes. Ecrire un algorithme dit de mise-à-jour qui calcule un (nouveau ?) MST si le poids d'une arête (u, v) de g est modifié. Quelle est la comlexité de votre algorithme ?

▷ *4 cas*

(i) $(u, v) \in$ *MST et on décrémente la valeur de* (u, v) *: rien à faire*

(ii) $(u, v) \notin$ *MST et on incrémente sa valeur : rien à faire.*

(iii) $(u, v) \notin$ *MST et on décrémente la valeur de* (u, v)*. On rajoute* (u, v) *dans le MST : on a un cycle. Sur ce nouveau cycle, on élimine l'arête de poids le plus fort : temps linéaire en utilisant un parcours BFS ou DFS.*

(iv) $(u, v) \in$ *MST et on incrémente sa valeur : on enlève* (u, v) *de MST on a alors* 2 *composantes connexes qu'on doit relier. Chaque composante peut-être calculée via un BFS ou un DFS en temps linéaire. On doit maintenant trouver l'arête de poids minimal reliant ces deux composantes : on le fait en temps linéaire sur le nombre d'arêtes.*

Votre graphe g s'est enrichi avec un nouveau sommet v et de nouvelles arêtes $a_1, a_2, \cdots$. Ecrire un algorithme de mise-à-jour du nouveau MST.

▷ *On rajoute* $a_{\min} = \min(a_1, a_2, \cdots)$ *et* v *dans le MST. Puis on rajoute* a_i *dans le MST : on crée un cycle. On fait un parcours pour le trouver et on enlève l'arête de poids maximal. On itère sur toutes les* a_i*. La complexité est alors* O*(le nombre de nouvelles arêtes* $\times$ *parcours DFS ou BFS*$)$*.*

Exercice 5 :

Une arête du MST est dit *critique* si sa suppréssion et son remplacement dans le MST augmente le poids de ce dernier. Ecrire un algorithme qui trouve toutes les arêtes critiques. Quelle est sa complexité ?

▷ *Rappel : l'algo de Kruskal construit un arbre couvrant minimum en sélectionnant des arêtes par poids croissant. Plus précisément, l'algorithme trie d'abord en pratique les arêtes par poids croissant et pour chacune d'entre elles l'algo la rajoute au MST si elle ne crée pas de cycle(s).*

Justement, au moment où l'insertion d'une arête e *crée un cycle et que ce cycle contient une arête* e' *de même poids que* e *alors* e' *n'est pas une arête critique (on peut la remplacer par* e *sans rien augmenter). Toutes les arêtes peuvent donc être marquées critique ou non-critique en temps* $O(|A| \log |A|)$ *pour le tri +* $O(|A|)$ *pour le parcours et détéction du cycle* $\times$ $O(|A|)$ *par arête à rajouter.*

Exercice 6 :

Le problème dit du *voyageur de commerce* ("Traveling Salesman Problem" ou TSP) consiste à trouver un tour (un cycle) de poids minimum entre n villes. Etant donné n villes et les coûts de voyage entre chaque pair de ville $c(v_i, v_j)$.

1. Montrer sur un exemple qu'il y a un nombre exponentiel de solutions potentielles au problème du TSP.

▷ *Si les poids sont tous les mêmes, alors il y aura* $n!$ *tours possibles. Ce problème est algorithmiquement très difficile (NP-difficile).*

2. Donner un algorithme qui donne une borne inférieure du coût du tour minimum.

▷ *Un MST est une borne inférieure d'un tel tour (cycle).*