Algorithmique 1

Devoir final

Les notes de cours et de TD sont autorisées. Pour les questions demandant de dérouler un algorithme, vous veillerez à faire apparaître les étapes de calcul permettant de juger de votre bonne compréhension (le résultat final est rarement suffisant). Le barème est indiqué à titre indicatif, il n'est pas définitif.

Question 1. Un *tournoi* est un graphe orienté obtenu à partir d'un graphe non orienté complet auquel on aurait ajouté un sens à chacun des arcs. Plus formellement, il s'agit d'un graphe orienté $G = (S, A)$ tel que pour toute paire de sommets $i, j \in S$, on a :

- soit $(i, j) \in A$ et $(j, i) \notin A$,
- soit $(i, j) \notin A$ et $(j, i) \in A$.

Par ailleurs, un *chemin hamiltonien* est un chemin passant exactement une fois par chacun des sommets du graphe.

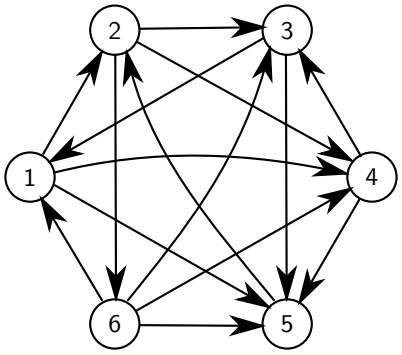

Figure 1: Un tournoi à 6 nœuds.

1.1.[0.5pt] Donner (sans justifier) un chemin hamiltonien pour le tournoi de la Figure 1.

1.2.[2pt] Montrer que tout tournoi admet au moins un chemin hamiltonien.

Question 2. Étant donné un graphe G non orienté et connexe, pondéré par une fonction w, on cherche à caractériser quelques propriétés d'un arbre couvrant de poids minimal. On suppose dans la suite que les pondérations sont deux à deux distinctes.

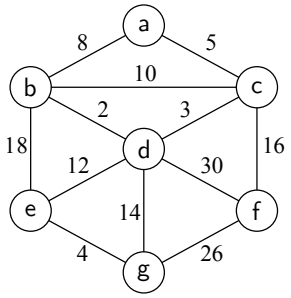

Figure 2: Un graphe non orienté, pondéré.

2.1.[1pt] Exécuter l'algorithme de Kruskal sur le graphe exemple de la Figure 2.

2.2.[2.5pt] Un arc (u, v) est dit *dangereux* s'il existe un cycle de G dans lequel (u, v) est l'arc de poids maximum. Montrer qu'un arbre couvrant minimum ne contient aucun arc dangereux.

2.3.[2.5pt] Un arc est dit *utile* s'il ne fait partie d'aucun cycle de G. Montrer qu'un arbre couvrant minimum contient nécessairement tous les arcs utiles.

Question 3. On considère un tableau A de taille n à valeurs dans $\mathbb{R}$, dont les entrées sont triées par ordre croissant.

3.1.[1.5pt] Proposer un algorithme qui permet de trouver deux nombres a et b dans le tableau tels que $a + b = 0$, ou qui renvoie FAUX dans le cas où un tel couple de nombres n'existe pas. La complexité dans le pire cas devra être en $O(n)$.

3.2.[3pt] Proposer un algorithme qui permet de trouver trois nombres a, b et c dans le tableau tels que $a + b + c = 0$, ou qui renvoie FAUX dans le cas où un tel triplet de nombres n'existe pas. La complexité dans le pire cas devra être en $O(n^2)$. Conseil : on pourra s'inspirer de la solution proposée la question précédente pour écrire une fonction auxiliaire.

Question 4. Soit A un tableau de nombres de taille $n \geq 3$, tel que $A[1] \geq A[2]$ et $A[n-1] \leq A[n]$. Un *minimum local* est un élément du tableau $A[i]$, tel que $A[i-1] \geq A[i]$ et $A[i] \leq A[i+1]$.

4.1.[1pt] Montrer que A contient au moins un minimum local.

4.2.[3pt] Proposer un algorithme en $O(\log n)$ permettant de trouver un minimum local dans A.

Question 5. En typographie, la *justification* d'un paragraphe consiste à répartir les mots entre les lignes de façon à ce que l'ensemble soit harmonieux. Pour ce problème, on utilise une police dont les caractères ont une largeur fixe. Formellement, on souhaite agencer les mots sur des lignes de longueur M. On dispose en entrée d'une séquence de n mots, de longueurs $l_1, l_2, \ldots, l_n$ (tous de taille inférieure à M). Étant donnée une ligne contenant les mots de i à j (inclus), le nombre d'*espaces résiduels* à la fin de la ligne est

$$e_{i,j} = M - \sum_{k=i}^{j} l_k - (j - i).$$

Le coût d'une telle ligne est défini comme $e_{i,j}^3$, sauf pour la dernière ligne dont le coût est nul. On cherche à trouver un découpage de coût total minimal.

5.1.[1pt] On note c_j le coût minimal d'un paragraphe contenant les j premiers mot (où toutes les lignes comptent). Proposer une relation de récurrence pour c_j.

5.2.[2pt] En déduire une méthode utilisant la programmation dynamique pour le calcul du coût optimal.