**IFT 2121 H06** **Geňa Hahn**

Introduction à l'algorithmique

# DEUXIEME EXAMEN INTRA

le 23 mars 2006

Durée: 55 minutes

Valeur: 15% de la note totale

Réponses

1. **(20 points)** Enoncez TRES précisément le problème de l'arbre couvrant (sous-tendant) minimum. Donnez toutes les définitions nécessaires (vous pouvez supposer connue celle d'un graphe $G = (V, E)$).

   *Etant donné un graphe $G = (V, E)$ pondéré par $w : E \longrightarrow \mathbb{R}$, trouver un arbre couvrant minimum, c'est-à-dire un sous-graphe connexe sans cycle $T = (V, F)$ tel que pour tout autre arbre couvrant $T' = (V, F')$, $\sum_{e \in F} w(e) \leq \sum_{e \in F'} w(e)$.*

   *On pourrait aussi énoncer le problème dans le format :*

   *DONNEES: graphe $G = (V, E)$, $w : E \longrightarrow \mathbb{R}$*
   *BUT: un sous-graphe connexe sans cycle $T = (V, F)$ tel que que pour tout sous-graphe connexe sans cycle $T' = (V, F')$, $\sum_{e \in F} w(e) \leq \sum_{e \in F'} w(e)$.*

2. **(20 points)** En utilisant le théorème vu en cours, prouvez que l'algorithme de Prim fonctionne correctement sur un graphe connexe (si vous êtes plus confortable avec la Propriété 3.2.1. du livre, vous pouvez utiliser celle-ci à la place du théorème). Voir ANNEXE pour les deux énoncés.

   *L'algorithme commence avec un ensemble d'arêtes vide. La partition initiale du théorème est alors $P = \{v\}_{v \in V}$, i.e., chaque sommet est dans une classe à part. Après la première itération, une arête de poids minimum sera choisie par Prim et sera contenue dans un ACM du graphe, par le théorème. Si, après $k$ itérations, Prim obtient un ensemble d'arêtes $F'$ contenu dans un ACM, cet ensemble induit un sous-graphe connexe (c'est ce que Prim fait!) et il donnera une classe $C_0$ de la partition. Les autres classes seront chacune formée d'un sommet isolé. Prim choisira une arête $uv$ de poids minimum parmi celles reliant $C_0$ aux autres classes; une telle arête existera toujours car le graphe est connexe. Par le théorème, il existe un ACM de $G$ contenant $F' \cup \{uv\}$. Donc après $n - 1$ itération, on aura un ensemble d'arêtes qui induit un arbre couvrant contenu dans un ACM; cet arbre sera donc un ACM.*

3. **(20 points)** Soit $A = [2, \sqrt{3}, 5, -\frac{4}{\sqrt{2}}, 3, \sqrt{3}, 5, 3\sqrt{2}]$. Pour calculer la transformée de Fourier discrète de $A$, on commence par remplir le tableau $T$ pour que tous les calculs subséquents puissent être faits sur place, en n'utilisant que deux variables $X$ et $Y$ et le tableau $\Omega$ qui peut être obtenu à partir de celui donné dans l'annexe.

   (a) (8 points) Remplissez le reste du tableau:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | 3 | 5 | 5 | $\sqrt{3}$ | $\sqrt{3}$ | $-2\sqrt{2}$ | $3\sqrt{2}$ |

   (b) (12 points) Quel est le contenu de $T[5]$ après la troisième itération?
   *Les tableaux ci-dessous sont donnés pour vous faciliter les calculs et non pas pour la réponse. Certaines valeurs y sont déjà en guise d'indications.*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | 3 | 5 | 5 | $\sqrt{3}$ | $\sqrt{3}$ | $-2\sqrt{2}$ | $3\sqrt{2}$ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 5 | $-1$ | 10 | 0 | $2\sqrt{3}$ | 0 | $\sqrt{2}$ | $-5\sqrt{2}$ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 15 | $-1$ | $-5$ | $-1$ | $2\sqrt{3}+\sqrt{2}$ | $-i5\sqrt{2}$ | $2\sqrt{3}-\sqrt{2}$ | $i5\sqrt{2}$ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | $5i-6$ | | |

**Réponse:** $5i-6$

4. (**20 points**) Soit $A$ un algorithme de tri par comparaison et soit $t(n)$ le nombre de comparaisons faites par $A$ sur un fichier de $n$ clés dans un pire cas. Prouvez que $t(n) \in \Omega(n \lg n)$.

   *Soit $A$ un tel algorithme. Son fonctionnement sur un fichier de $n$ clés peut être modélisé par une arborescence de décision : chaque sommet correspond à une comparaison de deux clés $a$ et $b$ et ses enfants aux comparaisons des clés choisies suivant le résultat, si $a \leq b$ la comparaison suivante peut être différente de celle faite si $a > b$. Quand l'algorithme s'arrête, le fichier devrait être trié. Si le fichier de départ contient les clés $c_1, \ldots, c_n$ dans un ordre inconue, toute permutation $c_{\sigma(1)}, \ldots, c_{\sigma n}$ doit pouvoir être retrouvée.*
   *On peut aussi voir cet arborescence comme ceci: chaque sommet correspond aux permutations possible en ce moment ($n!$ à la racine, une seule dans chaque feuille) et si le nombre de permutations possible est $> 1$, on choisi $a, b$ comparer et on "élimine" les permutation incompatibles avec le résultat de la comparaison* **merci à un de vous pour avoir suggéré ce point de vue**.
   *Dans tous les cas, cet arbre binaire aura au moins $n!$ feuilles, donc une profondeur d'au moins $\lg n! \in \Theta(n \lg n)$. Mais le nombre de comparaison faites au pire par $A$ est précisément la profondeur de cet arbre. On a alors $t(n) \in \Omega(n \lg n)$.*

5. (**20 points**) Prouvez que $n \lg n \notin \Omega(n^2)$

   *Méthode 1. $lim_{n \longrightarrow \infty} \frac{n \lg n}{n^2} = 0$, donc, par notre théorème, $n \lg n \in O(n^2)$ mais $n^2 \notin O(n \lg n)$, i.e. $n \lg n \notin \Omega(n^2)$.*

   *Méthode 2. $lim_{n \longrightarrow \infty} \frac{n^2}{n \lg n} = \infty$, donc (par définition de la limite) pour tout $c > 0$ il existe un $n_0 \in \mathbb{N}$ tel que pour tout $n \geq n_0$, $\frac{n^2}{n \lg n} > c$, i.e. $\frac{n^2}{c} > n \lg n$, i.e. $n \lg n \not\geq dn^2$ pour tout $d > 0$ car un tel $d$ définirait un $c = \frac{1}{d}$ contredisant le limite.*

   *Méthode 3. $lim_{n \longrightarrow \infty} \frac{n \lg n}{n^2} = 0$, i.e. pour tout $c > 0$, $\frac{n \lg n}{n^2} < c$ à partir d'un $n_c \in \mathbb{N}$. Donc pour tout $c > 0$, $n \lg n < n^2 c$, i.e. $n \lg n \notin \Omega(n^2)$.*