Examen L3 – Algorithmique

Yves Robert

8 Janvier 2014, durée 3h

Tous les exercices sont indépendants. Je vous conseille commencer par les amusettes, et de ne pas chercher les trucs marqués **(difficile)** avant d'avoir fini le reste. Chaque algorithme donné doit être accompagné de sa complexité, ainsi que d'une explication (plus ou moins brève) expliquant pourquoi il renvoie la bonne réponse. Enjoy and good luck !

1 Amusettes

1.1 Stations d'essence

Bob roule sur l'autoroute en voiture. Il part de Lyon (kilomètre 0) avec le plein d'essence et va, euh, très loin, à Tombouctou. La i-ème station-service est au kilomètre k_i. Son autonomie (plein d'essence) est de d kilomètres. Quel est son algorithme pour minimiser le nombre d'arrêts pour faire le plein ?

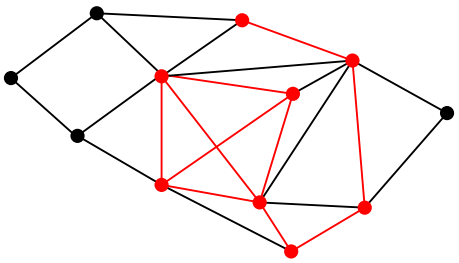
1.2 Analyse du cours d'actions

On regarde le cours passé d'une action pendant n jours consécutifs et on se demande quel profit on aurait pu faire. Autrement dit, étant donnés n nombres $p_1, p_2, \ldots, p_n$, ou p_i est le cours de l'action au i-ème jour, on veut trouver $M = \max_{1 \leq i < j \leq n}(p_j - p_i)$ (acheter le jour i, revendre un jour j ultérieur), ou renvoyer 0 si $M < 0$ (pas de profit possible).

1. Donner un algorithme diviser-pour-régner de coût $O(n \log n)$
2. **(difficile)** Donner un algorithme de coût $O(n)$

1.3 NP-complétude : cerf-volant

Un cerf volant de taille g est un graphe de $2g$ sommets dont g forment une clique et g forment une chaîne qui s'attache à l'un des sommets de la clique. Voici un exemple avec $g = 4$:



Formellement, en notant les sommets $v_1, \ldots, v_{2g}$, les arêtes sont $(v_i, v_j,\ 1 \leq i < j \leq g$ (la clique) et $v_i, v_{i+1},\ g \leq i \leq 2g-1$ (la chaîne attachée en v_g).

Montrer que le problème suivant est NP-complet : étant donné un graphe $G = (V, E)$ et un entier g, G contient-il un cerf-volant de taille g ?

1.4 NP-complétude : ordonnancement

On exécute n tâches indépendantes T_i, $1 \leq i \leq n$, sur un seul processeur. La durée de T_i est a_i, et T_i doit s'exécuter sans interruption dans l'intervalle $[r_i, d_i]$ (r_i est la date de disponibilité ou *release time* et d_i est l'échéance ou *deadline*). Tous les nombres a_i, r_i et d_i sont entiers. On donne une constante entière D et on veut décider si on peut exécuter les n tâches dans l'intervalle $[0, D]$. Montrer que ce problème est NP-complet.

2 Problème qui ressemble au cours

On considère l'algorithme glouton pour ordonnancer n tâches indépendantes T_i de durée a_i sur p processeurs. A chaque étape, l'algorithme glouton alloue la tâche courante au processeur le moins chargé. Rappel, on a étudié en cours le cas $p = 2$. On a toujours les deux versions, *online* où les a_i ne sont pas triés, et *offline* où ils le sont.

1. Montrer que glouton *online* est une $2 - \frac{1}{p}$ approximation, et donner un exemple où ce ratio est atteint
2. Montrer que glouton *offline* est une $\frac{4}{3} - \frac{1}{3p}$ approximation, et donner un exemple où ce ratio est atteint

3 Placement de serveurs dans les arbres

On s'intéresse au placement de serveurs dans les arbres. Un arbre a N noeuds internes $n_1, \ldots, n_N$ (dont la racine) et F feuilles $f_1, \ldots, f_F$. Chaque feuille f_i, $1 \leq i \leq F$ porte un nombre entier r_i de requêtes. On peut placer un serveur de capacité W sur n'importe quel noeud interne (y compris la racine). Chaque feuille f devra être associée à un serveur serveur(f) qui doit être un noeud interne sur le chemin (unique) qui mène de la feuille à la racine. On va chercher à minimiser le nombre de serveurs avec la contrainte de capacité suivante : la somme des requêtes servies par un serveur donné ne doit pas dépasser W : si on a placé un serveur sur le noeud n_j, alors

$$\sum_{\text{serveur}(f_i)=n_j, 1 \leq i \leq F} r_i \leq W$$

.

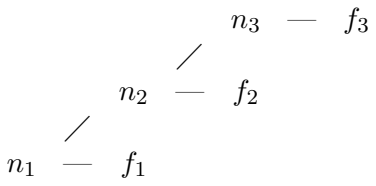
Pour choisir le serveur de chaque feuille, on considère deux règles possibles

CLOSEST dans cette stratégie, le serveur de chaque feuille doit être le premier serveur rencontré quand on remonte le chemin (unique) de la feuille vers la racine

UP dans cette stratégie moins contraignante, le serveur de chaque feuille peut être n'importe quel serveur rencontré quand on remonte le chemin (unique) de la feuille vers la racine

Dans chaque cas, on cherche à placer un nombre minimal de serveurs, tout en respectant la contrainte de capacité et la règle de choix de la stratégie. Pour un arbre donné $\mathcal{A}$, on note $CLOSEST(\mathcal{A})$ le nombre minimal de serveurs à placer avec la stratégie CLOSEST. On définit de même $UP(\mathcal{A})$ pour la stratégie UP.

1. On considère l'arbre $\mathcal{A}$ suivant, à 3 sommets et 3 feuilles, de racine n_1, et on pose $W = 10$:



   — Montrer que $CLOSEST(\mathcal{A}) = 3$ et $UP(\mathcal{A}) = 2$ si $r_1 = 3$, $r_2 = 9$ et $r_3 = 7$
   — Déterminer $CLOSEST(\mathcal{A})$ et $UP(\mathcal{A})$ si $r_1 = 3$, $r_2 = 9$ et $r_3 = 8$

2. **(difficile)** Donner un algorithme polynomial (en la taille $N + F$ de $\mathcal{A}$) pour déterminer $CLOSEST(\mathcal{A})$

3. Montrer que le problème suivant est NP-complet : étant donné un arbre $\mathcal{A}$ et un entier K, est-ce que $UP(\mathcal{A}) \leq K$?

4. **(difficile)** Pour tout entier K arbitrairement grand, monter qu'il existe un arbre $\mathcal{A}$ tel que $\frac{CLOSEST(\mathcal{A})}{UP(\mathcal{A})} \geq K$.