

INF550. Conception et analyse d'algorithmes

Examen du mercredi 7 décembre 2011. 3 heures

Documents autorisés : poly, transparents du cours, énoncés et corrigés de PC, notes manuscrites

On vous demande d'accorder une attention particulière à une rédaction soignée, qui sera un critère important dans l'évaluation. Les quatre parties du sujet sont indépendantes et peuvent être traitées dans l'ordre de votre choix. Aborder deux parties peut suffire à avoir une bonne note. Vous trouverez en fin de texte un appendice contenant quelques rappels de terminologie.

**I. Exercice de remplissage de salles
(Recherche de flot maximal, programmation linéaire)**

Vous êtes embauché pour aider le service de la scolarité d'une école à gérer l'inscription des élèves aux cours en tenant compte d'une série de contraintes plus ou moins réalistes. La semaine type est divisée en k créneaux horaires et dans le i ème créneau est proposé un ensemble $C_i = \{C_{i,1}, \dots, C_{i,\ell_i}\}$ de cours. Chaque cours $C_{i,j}$ a lieu dans une salle qui peut accueillir $c_{i,j}$ élèves. Par ailleurs chacun des n élève se déclare intéressé par un sous-ensemble $E_\ell \subset \bigcup_{i=1}^k C_i$ de cours (ici ℓ est le matricule de l'élève).

1. Les élèves n'étant de toute façon pas terriblement assidus, le responsable du service se dit dans un premier temps qu'il n'est pas gênant qu'un élève soit inscrit à plusieurs cours durant un même créneau horaire. Proposer un algorithme polynomial basé sur un problème de flot pour affecter chaque élève à au plus M cours parmi ceux qu'il a choisis, en maximisant le nombre d'inscriptions sans dépasser les capacités des salles.
2. À la réflexion, le responsable souhaite faire en sorte que chaque élève puisse effectivement suivre l'ensemble des cours auxquels il est inscrit. Proposez un nouvel algorithme d'affectation tenant compte de cette contrainte.
3. Dans les affectations recherchées aux deux questions précédentes un élève peut se retrouver inscrit à aucun cours. Pour éviter cela, le responsable du service se propose de maximiser le minimum des nombres de cours suivis par chaque étudiant (au lieu du nombre total d'inscriptions). Proposer une modélisation de ce problème par un programme linéaire. L'algorithme obtenu est-il polynomial? *Indication : on pourra par exemple associer une variable $x_{i,j,\ell}$ à chaque paire $(C_{i,j}, \ell)$.*

Dans le cadre la révision générale des politiques publiques (RGPP), le responsable du service est remplacé par un spécialiste de la rationalisation des ressources, chargé de s'assurer que chaque cours fonctionne à plein effectif.

3. Afin d'arriver au plein effectif, le spécialiste se propose naturellement de renoncer à prendre en compte les choix des élèves. Il vous demande un critère pour vérifier

rapidement, étant donnée la répartition des cours en créneaux horaires, s'il existe une affectation des n élèves permettant de remplir chaque salle de cours, sous les seules contraintes qu'un élève ne peut être présent dans 2 salles simultanément et ne peut suivre qu'au plus M cours ?

II. Élimination de triangles (NP-complétude, complexité paramétrique)

On rappelle tout d'abord que le problème RECOUVREMENT DES ARÊTES PAR SOMMETS suivant est NP-complet (ce problème est aussi appelé VERTEX COVER) :

Donnée : un graphe $G = (V, E)$ de taille $n = |V|$, et un entier k ;

Question : existe-t-il un ensemble X de k sommets tel que pour toute arête xy de G , soit $x \in X$ soit $y \in X$.

On considère maintenant le problème RECOUVREMENT DES TRIANGLES suivant :

Donnée : un graphe $G = (V, E)$ de taille $n = |V|$ et un entier $k \geq 0$

Question : existe-t-il un sous-ensemble X d'au plus k sommets de G tel que le sous-graphe de G dans lequel on a supprimé les sommets de X , noté $G[V \setminus X]$ ne contienne pas 3 sommets deux-à-deux voisins ;

Autrement dit on veut savoir si on peut intersecter tous les triangles de G avec au plus k sommets, ou encore on cherche un ensemble de k sommets "couvrant" tous les triangles.

1. Montrer que le problème RECOUVREMENT DES TRIANGLES est NP-complet.

On considère que le problème est paramétré par le nombre de sommets dans l'ensemble couvrant, on appelle k -RECOUVREMENT DES TRIANGLES cette version paramétrique.

2. Montrer par exploration arborescente bornée que le problème k -RECOUVREMENT DES TRIANGLES est FPT, c'est-à-dire qu'il existe une fonction $f(k)$, une constante $c > 0$ et un algorithme de complexité $O(f(k) \cdot n^c)$ le résolvant.
3. On va maintenant chercher un noyau polynomial au problème.
 - (a) Montrer que si l'arête xy est commune à $k+1$ triangles alors x ou y appartiennent à tout ensemble couvrant de cardinal $\leq k$.
 - (b) Montrer que si aucune arête n'est commune à $k+1$ triangles, alors un sommet commun à au moins $k^2 + 1$ triangles appartient à tout ensemble couvrant de cardinal $\leq k$.
 - (c) En déduire le problème k -RECOUVREMENT DES TRIANGLES admet un noyau de taille $O(k^3)$.
4. Quelle est la complexité de l'algorithme obtenu en combinant les questions 2 et 3 ?

III. Arbre couvrant de degré minimum (NP-complétude, recherche locale, approximation)

On rappelle que le problème CHEMIN HAMILTONIEN suivant est NP-complet :

Donnée : un graphe $G = (V, E)$ connexe de taille $n = |V|$;

Question : existe-t-il un chemin passant une et une seule fois par chaque sommet de G ?

On considère le problème ARBRE-COUVRANT-DEGRÉ-MIN suivant :

Donnée : un graphe $G = (V, E)$ connexe de taille $n = |V|$ et un entier k

Question : existe-t-il un arbre couvrant T de G dont les sommets soient de degré au plus k dans T ?

Étant donné un sous-ensemble d'arêtes $F \subset E$ d'un graphe $G = (V, E)$, on note $\delta_F(x)$ le degré de x dans F , c'est-à-dire le nombre d'arêtes de F incidente à x et $\Delta(F) = \max_{x \in V} \delta_F(x)$ le degré maximal dans F . Enfin on note $\Delta^*(G)$ le minimum de $\Delta(T)$ pour T dans l'ensemble des arbres couvrants de G . Le problème ARBRE-COUVRANT-DEGRÉ-MIN(G, k) est alors de décider si $\Delta^*(G) \leq k$.

1. Montrer que si $k \geq \Delta(E)$ alors la réponse au problème ARBRE-COUVRANT-DEGRÉ-MIN(G, k) est positive.
2. Montrer que le problème k -ARBRE-COUVRANT-DEGRÉ-MIN, pour lequel k est fixé et on doit savoir dire pour tout graphe G si ARBRE-COUVRANT-DEGRÉ-MIN(G, k), est NP-complet. (Indication : on pourra commencer montrer la NP-complétude dans le cas $k = 2$).
3. Montrer que si $X \subset V$ est un ensemble de sommets tel que $G[V \setminus X]$ ait t composantes connexes alors $\Delta^*(G) \geq \lceil \frac{r+t-1}{r} \rceil$, où $r = |X|$. *Indication : calculer le degré moyen des sommets de X dans un arbre couvrant T de G .*

Un arbre couvrant T de G est *localement optimal* si pour toute arête (u, v) de $G \setminus T$ et tout sommet w sur le chemin $T[u, v]$ le degré $\delta_T(w)$ de w dans T est au plus égal à $\max(\delta_T(u), \delta_T(v)) + 1$.

4. Proposer un algorithme pour trouver un arbre couvrant localement optimal dans un graphe connexe G . *Indication : montrer que si l'arbre couvrant T de G n'est pas localement optimal alors il existe un autre arbre couvrant T' de G tel que $\Delta(T') \leq \Delta(T)$ et tel que T' contient un sommet de degré $\Delta(T)$ de moins que T .*
5. Soit S_i l'ensemble des sommets de degré au moins i d'un arbre couvrant T d'un graphe G à n sommets. Montrer que pour toute constante $c > 1$ il existe $i \geq \Delta(T) - \log_c n$ tel que $0 < |S_{i-1}| \leq c|S_i|$.
6. Montrer que si la suppression des sommets de S_i découpe T en t composantes connexes, alors la suppression des sommets S_{i-1} découpe G en au moins t composantes connexes.
7. Montrer que le degré maximal d'un arbre localement optimal T est au plus $c\Delta^*(G) + \lceil \log_c n \rceil$.

IV. Gestion de mémoire (Algorithme online)

On considère un modèle de mémoire dans lequel on peut consulter les données non seulement en mémoire centrale, mais aussi directement dans la mémoire externe : les données sont découpées en pages, la consultation d'une page en mémoire centrale est gratuite, celle d'une page en mémoire externe à un coût unitaire et le chargement d'une page en mémoire centrale coûte D fois le coût d'une consultation. Nous nous intéressons à la conception d'un gestionnaire de mémoire qui décide au fur et à mesure de l'arrivée des requêtes s'il vaut la peine de charger la page demandée en mémoire ou pas.

On se concentre ici (sauf à la dernière question) sur le cas où le découpage en pages des données est calibré sur la taille de la mémoire centrale : il y a à tout moment une unique page en mémoire centrale, toutes les autres sont en mémoire externe.

L'algorithme de gestion de la mémoire centrale que nous étudions associe à chaque page p (qu'elle soit en mémoire externe ou en mémoire centrale) un compteur $c(p)$. Au départ tous les compteurs sont mis à zéro, puis le gestionnaire de mémoire effectue les opérations suivantes en boucle :

- a. Réceptionner la prochaine requête p .
- b. Incrémenter le compteur $c(p)$.
- c. Si $c(p) = D$ alors
 - Charger la page p en mémoire centrale si elle n'y est pas déjà,
 - Tant que $c(p) > 0$
 - Réceptionner la prochaine requête q
 - Décrémenter le compteur $c(p)$ chaque fois que $q \neq p$.
- d. Retourner en a.

La réception des requêtes est bloquante : lorsqu'il n'y pas de requête à traiter le gestionnaire de mémoire attend. Remarquer que durant la boucle interne "Tant que..." les compteurs $c(q)$ pour $q \neq p$ sont bloqués.

1. Compléter l'évolution des compteurs pour $D = 3$ dans l'exemple ci-dessous d'une suite $s = (s_1, s_2, \dots)$ de requêtes portant sur les pages 1, 4 et 6 :

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	...
s_i	1	1	4	6	4	1	4	6	1	1	4	1	4	6	1	4	6	6	...
$c(1)$	1	2	2	2	2	3	2												
$c(4)$	0	0	1	1	2	2	2												
$c(6)$	0	0	0	1	1	1	1												

Pour étudier cet algorithme, divisons la séquence $s = (s_1, s_2, \dots)$ en sous-ensembles de requêtes appelées *phases*, associées à chaque remise à zéro d'un compteur : la phase I_j^p contient les indices i des requêtes réceptionnées entre la j ème et la $(j + 1)$ ème remise à zéro du compteur $c(p)$ et satisfaisant l'une des deux conditions suivantes :

- soit la requête s_i provoque l'incrément du compteur $c(p)$,

- soit la requête s_i est réceptionnée après la requête ayant amené le compteur $c(p)$ à D pour la $(j + 1)$ ème fois.

Dans l'exemple de la question 1, vérifier que les premières phases sont

$$\begin{aligned} I_0^1 &= (1, 2, 6, 7, 8, 9, 10, 11), & I_1^1 &= (12, \dots), & \dots \\ I_0^4 &= (3, 5, 13, 14, 15, 16, 17), & & \dots \\ I_0^6 &= (4, 18, \dots), & & \dots \end{aligned}$$

On remarque que deux phases portant sur des requêtes distinctes peuvent s'entrelacer et que la requête I_j^p commence par une suite de requête à la page p et se termine par une suite d'au moins $D + 1$ requêtes consécutives.

2. Montrer que le coût d'une phase est au plus $3D$.

On va comparer ce coût payé par notre gestionnaire de mémoire à celui d'un algorithme arbitraire qu'on appellera OPT.

3. Étant donnée la séquence de requêtes s , montrer qu'on peut répartir le coût payé par OPT sur les différentes phases I_j^p précédemment définies de sorte que coût ainsi attribué à chaque phase soit au moins D .

(Indication : attribuer le coût payé par OPT pour le chargement d'une page qui fait sortir p du cache à la phase I_i^p durant laquelle ce chargement est effectué, et analyser le coût payé par OPT pour le traitement des autres requêtes de chaque phase.)

4. En déduire le facteur de compétitivité de notre algorithme.
5. **Question subsidiaire difficile.** Que dire de la généralisation du problème précédent au cas d'une mémoire cache de k pages.

Appendice : rappels de terminologie

- Un noyau d'une instance (G, k) de taille n d'un problème est une instance (G', k') de taille au plus $g(k)$, obtenue en temps polynomial en n à partir de (G, k) et tel que les réponses aux deux instances soient identiques.

- Un graphe $G = (V, E)$ est *connexe* si, pour tout x et y dans V s'il existe un chemin de x à y dans G .

Les *composantes connexes* d'un graphe G sont les classes d'équivalence de sommets pour la relation "être lié par un chemin" : les composantes connexes de G forment une partition $C_1 \cup \dots \cup C_k$ de V telle qu'il existe un chemin de x à y si et seulement si x et y sont dans la même composante C_i .

- Étant donné un sous-ensemble X de l'ensemble des sommets d'un graphe $G = (V, E)$, on note $G[X]$ le graphe ayant X pour ensemble de sommets et contenant les arêtes de E qui ont leurs deux extrémités dans X .
- Un arbre est un graphe connexe sans cycle. Un arbre qui possède n sommets possède $n - 1$ arêtes. Un arbre couvrant d'un graphe $G = (V, E)$ est un ensemble T d'arêtes tel que (V, T) soit un arbre. Si T est un arbre couvrant de G et x, y sont deux sommets de G , alors $T[x, y]$ désigne l'unique chemin de x à y dans T (on dit aussi la branche de x à y dans T).