École Polytechnique
Promotion 1999

Majeure informatique

Cours « Conception et analyse d'algorithmes »
Correction du contrôle du 20 mars 2002.

Problème 1. *Ordonnancement*

1. Une solution possible est la suivante :
le processeur 1 effectue les tâches de durées 4, 3, 7, le processeur 2 celles de durées 2, 2, 1, 6, 3, enfin le dernier processeur celles de durées 5, 6, 3.

2. La version sous forme de problème de décision se présente comme suit : on se donne un ensemble de durées de tâches $d_1, d_2, \ldots, d_n$ un nombre p de processeurs et un entier D, il s'agit de savoir s'il est possible de répartir les d_i en p sous-ensembles disjoints de façon telle que la somme des éléments dans chacun des sous-ensembles soit inférieure ou égale à D.

Soit $\{a_1, a_2, \ldots, a_k\}$ une instance de PARTITION, nombres qu'il s'agit de partager en deux sous-ensembles de sommes égales. On pose

$$2N = \sum_{i=1}^{k} a_i$$

et on considère l'instance de p-affecte ; $\{a_1, a_2, \ldots, a_k, N, \ldots N\}$ où le nombre N intervient $p-2$ fois, et on cherche à savoir s'il y a une affectation sur p processeurs dont la durée ne dépasse pas N sur chacun des processeurs.

3. On a pour tout $i \neq h$, en raison de l'algorithme de choix du processeur P_h :

$$u_i \leq u_h - d_j$$

4. On utilise le fait que

$$\sum_{i=1}^{p} u_i = \sum_{j=1}^{n} d_j$$

et que

$$pD_{min} \geq \sum_{j=1}^{n} d_j$$

Ainsi

$$pD_{min} \geq u_h + \sum_{j \neq h} (u_h - d_j) \geq pu_h - (p-1)d_j$$

Soit

$$pu_h \leq pD_{min} + (p-1)d_j$$

mais $d_j \leq D_{min}$, ainsi

$$pu_h \leq (2p-1)D_{min}$$

qui est le résultat demandé.

5. On considère $p(p-1)$ taches de durées 1 et une seule de durée p que l'on examine en dernier. Pour la solution optimale, on consacre l'un des processeurs a l'exécution de la dernière tache et les autres processeurs à celles des $p(p-1)$ autres tâches, ce qui donne une durée de p au total. L'algorithme glouton répartit les $p(p-1)$ tâches équitablement entre les p processeurs, puis la dernière est attribuée à l'un d'entre eux donnant une durée de $2p-1$ pour le processeur qui l'exécute et $p-1$ pour les autres.

6. Si toutes les tâches ont une durée strictement supérieure à $\frac{D_{min}}{3}$, il est facile de vérifier que l'algorithme glouton attribue au plus deux taches par processeur de façon optimale. Par contre s'il existe une tâche de durée inférieure ou égale à $\frac{D_{min}}{3}$, la durée de la dernière tâche considérée vérifiera la même inégalité. Ainsi en reprenant les calculs de la Question 4 , on a :

$$pu_h \leq pD_{min} + (p-1)\frac{D_{min}}{3}$$

Soit :

$$u_h \leq (\frac{4}{3} - \frac{1}{3p})D_{min}$$

7. On introduit les variables $x_{i,j}$ prenant la valeur 1 si le processeur i effectue la tache j et 0 sinon. On a alors les contraintes suivantes modélisant le problème de décision :

$$\sum_{i=1}^{p} x_{i,j} = 1 \quad \sum_{j=1}^{n} x_{i,j} \leq D \quad 0 \leq x_{i,j} \leq 1$$

Problème 2. *Sous-séquences communes maximales*

1. Les sous-séquences communes maximales de $ACATG$ et $TGACG$ sont ACG et TG.

Les deux mots demandés sont par exemple $a^{2n}b^n$ et $b^n a^{2n}$.

2. Si $(u, f', g') \in S(f, gb)$ alors u est sous-séquence commune maximale de f et de gb ; si u n'était pas sous-séquence commune maximale de fa et de gb c'est que ua serait sous-séquence commune de fa et de gb donc de fa et g ; ainsi $u \notin SSM(fa, g)$.

3. Un exemple est

$$(c, bc, c) \in S(bc, cab) \quad (c, bc, c) \notin S(bca, cab)$$

4. Si $(u, fa, g') \in S(fa, g)$ c'est que $u = u'a$ et qu'il est sous-séquence maximale commune à fa et g. Si u n'est pas sous-séquence commune maximale de fa et gb, ub serait sous-séquence commune à fa et gb donc à f et gb et $\sigma(u, fa)$ serait différent de fa.

5. Si (u, f', g') est un élément de $S(f, g)$, alors en posant $f'' = N_{fa}(f', a)$ et $g'' = N_{ga}(g', a)$ on a $(u, f'', g'') \in S(f, g)$.

Ainsi

$$S(fa, ga) = \bigcup_{(u,f',g') \in S(f,g)} (u, N_{fa}(f', a), N_{ga}(g', a))$$

6. Soient f et g de longueurs respectives n et m, pour chaque i, j tels que $1 \leq i \leq n$ et $1 \leq j \leq m$, on construit les triples d'entiers

$$S'(i, j) = \{(r, p, q) \mid (u, f', g') \in S(f_i, g_j), |u| = r, |f'| = p, |g'| = q\}$$

Dans cette notation f_i désigne le facteur gauche de f de longuer i, g_j le facteur gauche de g de longueur j et $|w|$ la longueur du mot w.

On a alors les formules suivantes qui permettent de donner un algorithme de calcul de $S'(i, j)$ en fonction de $S'(i, j-1), S'(i-1, j)$ et $S(i-1, j-1)$ dans le style programmation dynamique :

Si la i-ème lettre de f est égale à la j-ème lettre de g :

$$S'(i, j) = \{(r, p, q) \mid (r-1, p', q') \in S'(i-1, j-1), p = |N_f(f_{p'}, a)|, q = |N_g(g_{q'}, a)|\}$$

Et si elles sont différentes :

$$S'(i, j) = \{(r, p, j) \in S'(i-1, j)\} \cup \{(r, i, q) \in S'(i, j-1)\} \cup (S'(i-1, j) \cap S'(i, j-1))$$

Problème 3. *Couplage maximal en* $O(n^{5/2})$

1. Dans $C' = C \oplus P$ on ajoute dans P les arêtes qui n'étaient pas dans C et et on supprime celles qui y étaient ; les extremités de P deviennent incidentes à une arête de C' et les autres sommets internes à P restent incidents à une seule arête de C'. On a donc toujours un couplage.

Comme P avait une arête de moins en dehors de C que dans C on améliore le couplage d'une arête.

$$|C \oplus P| = |C| + 1$$

2. On calcule $C \oplus C'$, dans ce graphe tous les sommets on degré 2 ou 1, il est ainsi union de chemins et de cycles (voir le poly page 51). Comme C' contient q éléments de plus que C, il y a au moins q chemins qui contiennent une arête de plus dans C' que dans C, ces chemins sont disjoints puisqu'on a des couplages et ils constituent des chemins d'amélioration pour C.

3. Soit $P_1, P_2 \dots P_{s-r}$ les chemins d'amélioration dont l'existence a été prouvée à la question précédente. La longueur $\ell(P_i)$ du chemin P_i est $2n_i + 1$ où n_i est le nombre d'arêtes de P_i qui sont dans C. On a alors ;

$$\sum_{i=1}^{s-r} \ell(C_i) = s - r + 2\sum_{i=1}^{s-r} n_i$$

Mais la somme des n_i est certainement inférieure au nombre r d'éléments de C, ce qui donne :

$$\sum_{i=1}^{s-r} \ell(C_i) \leq 2r + (s - r)$$

D'où le résultat demandé.

4. A chaque étape on cherche le chemin d'amélioration de longueur minimum, il peut être obtenu en temps $O(nm)$ au pire par un algorithme élémentaire consistant à examiner les chemins qui partent d'un sommet qui n'est pas incident à une arête du couplage courant, une borne pour la complexité est donc $O(nms)$.

5. On sait que C_r contient r arêtes ; aussi, d'après la Question 3, la longueur de P_r est majorée par $2\lfloor \frac{r}{s-r} \rfloor + 1$, soit comme $r = s - \lceil \sqrt{s} \rceil$ on obtient :

$$\ell(P_r) \leq 2\lfloor \sqrt{s} \rfloor + 1$$

Les longueurs des chemins considérés pour atteindre C_r sont toutes inférieures à $2\lfloor \sqrt{s} \rfloor + 1$, comme il s'agit de nombres tous impairs, il ne peut y avoir que la moitié des nombres compris entre 1 et $2\lfloor \sqrt{s} \rfloor + 1$, soit $\lfloor \sqrt{s} \rfloor + 1$ valeurs possibles ; mais il reste encore $s - \lceil \sqrt{s} \rceil$ étapes à accomplir pour arriver à C_s et donc au pire $\lfloor \sqrt{s} \rfloor + 1$ nouvelles valeurs de longueurs soit au total $2\lfloor \sqrt{s} \rfloor + 2$ valeurs différentes.

6. On obtient que $P \oplus P'$ a plus d'éléments que $P_1 \oplus P_2$ car la différence symétrique peut contenir des chemins et des cycles. Ainsi

$$|P| + |P'| - 2|P \cap P'| \geq |P_1| + |P_2|$$

Mais P est de longueur minimale comme chemin d'amélioration de |C|, ainsi $|P_1| + |P_2| \geq 2|P|$ et on obtient le résultat.

7. La suite des inégalités demandées découle immédiatement de la question précédente, de plus on peut dire que si $|P| = |P'|$, et P, P' sont consécutifs alors $P \cap P'$ est vide.

8. L'algorithme consiste à trouver un ensemble de chemins d'amélioration tous de longueur minimale à chaque phase, ceci prend un temps $O(m)$ par phase. De plus à chaque phase la longueurs des plus courts chemins d'amélioration augmente strictement : puisque deux chemins d'amélioration de même longueur sont disjoints, ils interviennent dans la même phase. A la Question 6 on a vu que le nombre de longueurs différentes, donc le nombre de phases, est au plus en $O(\sqrt{s})$ donc en $O(\sqrt{n})$, ainsi l'algorithme est en $O(n^{2.5})$.