École polytechnique
Promotion 1998

Majeure informatique

Cours « Conception et analyse d'algorithmes »
Correction du contrôle classant du 21 mars 2001.

Problème 1.

1. On classe les tâches par ordre de durée d'exécutions croissantes, et on les réalise dans cet ordre. Le temps disponible pour réaliser les tâches étant le même pour toutes, il est clair qu'il vaut mieu par commencer par les tâches ayant une faible durée pour maximiser le nombre de tâches réalisées.

On peut effectivement voir ce problème comme un cas particulier du problème du sac-à-dos : chaque tâche a un poids correspondant à sa durée et un intérêt de 1, et la capacité du sac est le temps total disponible. Dans le cas ou tous les intérêts sont égaux, on obtient un algorithme polynomial puisque le classement des tâches demande un temps $n \log n$, et qu'il faut s'arrêter dès que l'on a dépasser la capacité du sac.

2. L'ensemble X vaut successivement $\{1\}$, $\{1, 2\}$, $\{1, 3\}$, $\{1, 3, 4\}$, $\{1, 3, 4, 6\}$.

3. On stocke X dans une file de priorité, en ordonnant par durée ; les tâches initiales sont ordonnées quant à elle dans un tableau. Le tri initial prend donc $n \log n$; pour chaque itération, il suffit de maintenir la durée totale utilisée jusqu'alors ; l'insertion de X dans la file prend $O(\log n)$, la suppression éventuelle de la tâche de durée la plus grande $O(\log n)$. Le coût total de l'algorithme est donc $O(n \log n)$.

4. On montre qu'à chaque étape k l'algorithme glouton donne l'ensemble de tâches $A_k = \{U_1, U_2, \ldots, U_j\}$ (chosies parmi les $\{T_1, T_2, \ldots, T_k\}$), réalisables dans les délais. telles que j soit maximal et que la somme des durées est minimale parmi les ensembles de tâches vérifiant ces 2 conditions. La preuve se fait par récurrence sur k. Il est clair que la propriété est satisfaite pour $k = 1$ puisque seule la tâche T_1 est considérée.

On suppose la proprété vraie à l'étape k on montrequ'elle est aussi satisfaite à l'étape $k + 1$. En effet, si on ajoute simplement T_{k+1}, on a un ensemble de cardinal maximal qui satisfait la condition de minimalité des sommes de durées. Si l'ajout de T_{k+1} provoque une impossibilité, il est alors clair que le cardinal maximal est le même pour l'étape k et l'étape $k+1$, la minimalité de la somme des durées s'obtient par l'hypothèse de récurrence.

5 Si le graphe G contient une clique de taille k, et qu'on suppose que l'on réalise les tâches sommets de la clique, arêtes de la clique, on peut réaliser dans les temps les T_x pour x dans la clique, puis les T_e pour e arête de la clique, puis les T_x pour x hors de la clique (car $m > k(k-1)/2$), soit au total $n + k(k-1)/2$.

Inversement, si cela est possible, on peut réaliser au moins $k(k-1)/2$ arêtes qui peuvent être précédées par au plus k sommets. Vu les précédences, cela implique que ces sommets forment une clique de taille k.

Problème 2.

I.1. La version de décision consiste à chercher un J tel que $\cup_{j \in J} S_j = E$ et $\sum_{j \in J} w(S_j) \leq w$.

I. 2. Il faut chosir S_v égal à l'ensemble des arêtes incidentes à v. A un recouvrement de E par une famille d'ensembles S_v on associe l'ensemble des sommets v qui apparaissent comme indice d'un S_v de recouvrement ; toute arête du graphe est clairement incidente à l'un de ceux-ci et on a un recouvrement par sommets de G. La réciproque est aussi immédiate.

II. 3. A une instance de COUVERTURE PAR SOMMETS on a asoocié une instance de COUVERTURE PAR ENSEMBLE en temps polynomial (les ensembles S_v sont représentés par les listes de voisins de chaque sommet dans le graphe) ; l'un admet une solution si et seulement si c'est le cas pour l'autre. C'est une réduction polynomiale.

II. 1. Soit $S_{i_1}, \dots S_{i_k}$ les sous-ensembles de la solution optimale. Si pour tout j on a :

$$\frac{w(S_{i_j})}{|S_{i_j}|} > \frac{W_0}{n}$$

alors $w(S_{i_j}) > \frac{W_0 |S_{i_j}|}{n}$ en sommant on obtient (comme on a un recouvrement de E : $\sum |S_{i_j}| \geq n$) $W_0 > \frac{W_0 n}{n}$ Qui est une contradiction.

II. 2. L'algoritme glouton chosit T_1 tel que $\frac{w(T_1)}{|T_1|}$ soit minimum, il est en particulier inférieur ou égal aux ratios des sous-ensembles intervenant dans la solution optimum, d'où le résultat en appliquant II. 1.

II. 3. A l'étape i on note $\overline{F} = E \setminus F$, on peut restreindre le problème à $\overline{F}$, considérer les $S'_i = S_i \setminus F$ et supposer que l'on est au début de l'algorithme. Dans cette sitaution $n - j + 1$ est un minorant de $|\overline{F}|$, d'où le résultat en utilisant la question précédente.

II. 4. Il y a au pire n étapes, et en sommant les inégalités de la question précédente, le coût obtenu par l'algorithme est inférieur ou égal à $W_0 H_n$. L'algorithme est donc H_n approché par définition de l'approximation.

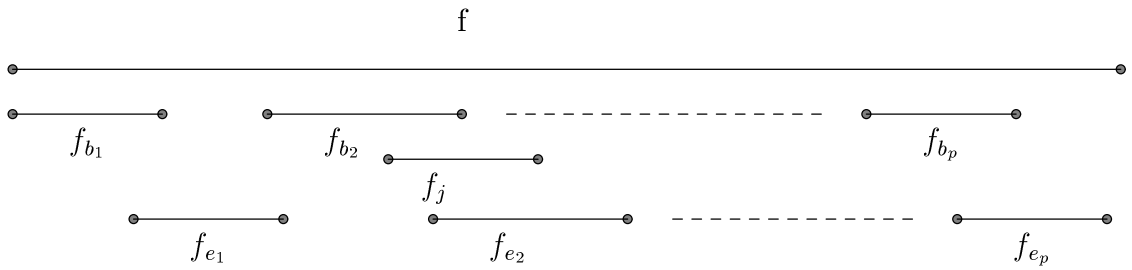
II. 5 . On peut prendre comme poids $1, 1/2, 1/3 \dots 1/n$ et $((1+\varepsilon)/n)$, l'algorithme chosisit S_n puis S_{n-1} et ainsi de suite jusqu'à S_1 obtenant un poids H_n ; alors que la solution optimale est S_{n+1} de poids $((1+\varepsilon)/n)$. Noter que l'énoncé imposait des poids entiers, il convient donc de multiplier tous les poids par $n!$ pour satisfaire cette condition.

III. 1. σ_{12k} est égal à $ab^k c$ et le f optimal est $ab^{k+1}c$.

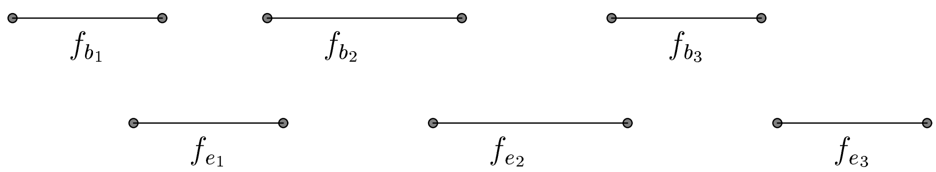
III. 2. L'algorithme à égalité de longueur de chevauchements pour l'exemple précédent peut choisir σ_{12k} qui donne un ensemble $S = \{ab^kc, b^{k+1}\}$ composé de deux mots qui n'ont qu'un 0-chevauchement d'où la solution obtenue ab^kc, b^{k+1}, celle-ci est de longeur $2k+3$ alors que la solution optimale est de longeur $k+2$ le seuil est supérieur ou égal à $\frac{2k+3}{k+2}$ qui aussi proche que l'on veut de 2.

III. 3. Il suffit de prendre pour f la concaténation des mots $\theta_1, \theta_2, \ldots \theta_p$. Comme les S_{θ_i} recouvrent S, tout f_j est élément d'un des S_{θ_i}, donc facteur du θ_i correspondant. Le poids W_0 de la solution optmale au problème Q est égal à la somme des longueurs des mots θ_i, mais c'est exactement la longuer de f.

III. 4. Comme on suppose que les f_i ne sont pas facteurs l'un de l'autre, on obtient que chaque f_j est facteur d'un segment de f situé entre le début d'un f_{b_i} et la fin d'un f_{e_i} (voir figure). Mais ce segment est justement égal à $\pi_i = \sigma_{b_i e_i k_i}$; ainsi tout f_j est facteur d'un π_i et il appartient au S_{π_i} correspondant. Les S_{π_i} constituent donc un recouvrement de S ; la somme de leurs poids est $\sum_{i=1}^{t} \ell(\pi_i)$



III. 5. Il suffit de montrer que π_1 et π_3 ne s'intersectent pas ou ce qui revient au même que tel n'est pas le cas pour f_{e_1} et f_{b_3}. Si f_{e_1} et f_{b_3} s'intersectent alors comme on le voit sur la figure ci-dessous, f_{b_2} serait facteur de f_{e_1} et l'énoncé exclut qu'un mot f_i soit facteur d'un f_j.



III.6. On considère l'algorithme qui remplace le problème P par le problème Q correspondant, puis qui résout Q par l'algorithme glouton de la partie II, enfin qui concatène les mots u correspondants aux S_u de la solution trouvée. On perd un facteur 2 en remplaçant P par Q, et un facteur H_n en se contentant de l'algorithme glouton. On a donc un algorithme $2H_n$ approché.

Problème 3.

I. 1. On peut utiliser la construction de l'arbre couvrant par l'algorithme glouton : quand a de V_{min} est examinée, x n'est pas encore dans l'arbre couvrant (les arêtes sont examinées par poids croissants, donc l'ajout de l'arête ne crée pas de cycle, et l'arête est donc ajoutée.

I. 2. Il y a au plus une arête minimum incidente par sommet, donc on peut prendre $\beta_n = n$, et même $n - 1$, car l'arête minimale du graphe est minimale pour ses deux extrémités. Inversement, une arête reliant deux sommets, une arête minimum ne peut correspondre qu'à deux sommets, donc $\alpha_n = n/2$. Pour la borne inférieure, il suffit de prendre la suite de poids suivants : 1, 10, 2, 9, 3, 8. Pour la borne supérieure, on peut prendre les poids 1, 2, 3, 4, 5, 6.

I. 3. Il suffit de rajouter à l'arbre recouvrant pour $Contr(G)$ les arêtes de V_{min}. Tous les sommets appartiendront bien au graphe correspondant. En outre, comme toutes les arêtes d'un cycle ne peuvent être dans V_{min}, l'ajout de ces arêtes ne crée pas de cycle. Enfin, un arbre de poids minimal pour G, en vertu de 1, contient toutes les arêtes de V_{min}. Comme il y a correspondance bijective (et préservant les poids, à un facteur additif près) entre arbres couvrants de G contenant V_{min} et arbres couvrants de $Contr(G)$, le résultat s'ensuit.

4. On détermine l'ensemble V_{min} en temps $O(m)$ (il suffit d'énumérer les arêtes en stockant pour chaque sommet la plus courte des arêtes incidentes rencontrée), puis on contracte lesdites arêtes en temps $O(m + n)$, et on répète le processus. Chaque étape du processus divise le nombre de sommets par 2, donc le nombre total d'étapes est majoré par $\log n$ et le temps de chaque étape est en $O(m+n)$ car on crée des arêtes multiples par contraction. En tout, on obtient donc une complexité $O(\log n(m + n))$.

II 1. On trouve trivialement $|E|/2$.

2. En vertu de la construction de l'algorithme glouton, une arête lourde n'est jamais dans l'arbre couvrant de G (car on a ajouté un chemin de x à y constitué d'arêtes de poids inférieur auparavant). Pour ce qui est des arêtes légères, on ne peut rien garantir : en effet, il peut exister dans G un chemin de x à y meilleur que dans $G_{/2}$.

II. 3. On considère les arêtes par poids croissant et on tire un bit à pile ou face pour chaque si $e_i \in G_{/2}$, si e_i ne crée pas de cycle avec les e_j pour $j < i$ et si l'on a tiré un 1, alors e_i appartient à F. Si e_i ne crée pas de cycle avec les e_j pour $j < i$ et si l'on a tiré un 0, alors l'arête est légère.

II. 4. Le processus peut être considéré comme majoré en moyenne par un tirage aléatoire de 0 ou 1 qui s'arrête lorsqu'on a trouvé $k \leq n - 1$ fois la valeur 1. Il est bien connu que le nombre de tirages moyen est $2k < 2n$. La majoration en moyenne implique le résultat.

III. 1. Le nombre de sommets n_2 de G_2 est au plus $n/8$ son nombre d'arêtes est en moyenne majoré par $m/2$.

G_3 a le même nombre de sommets que G_2 et son nombre d'arêtes est majoré par $2n_2$ d'après la question précédente.

III. 2.

On appelle récursivement `MSTProba` pour G_2 et G_3 ce qui donne

$$T(m,n) \leq c(m+n) + T(m/2, n/8) + T(n/4, n/8)$$

III. 3.

Si on suppose que $T(m',n') \leq 2c(m',n')$ pour $m' < m$ et $n' < n$, on obtient d'après l'inégalité trouvée plus haut

$$T(m,n) \leq c(m+n) + 2c(m/2 + n/8) + 2c(n/4 + n/8)$$

Mais ceci a le bon goût de se simplifier en : $c(m+m+n+n/4n/2+n/4) = c(2m+2n)$.