Algorithmique Avancée 2018 École Centrale-Supélec

Examen écrit final de rattrapage

Christoph Dürr, Nguyễn Kim Thắng

Les documents distribués dans le cadre du cours et les notes personnelles sont autorisés, mais pas les calculatrices. Une correction sera publiée dans la page web du cours.

1 Encens sur un autel

C'est le nouvel an lunaire et vous devez avoir en permanence un encens qui se consomme sur votre autel, pendant exactement n heures d'affilé, pour un entier positif n. Il existe deux types d'encens. Un encens baton qui se consomme en une heure et un encent spirale qui se consomme en 24 heures. Combien de possibiltés s'offrent à vous pour mener à bien cette tâche ? Appelons A_n la réponse. Par exemple $A_{23} = 1$, car la seule possibilité est d'allumer à la suite 23 encens batons. Un autre exemple est $A_{25} = 3$, car vous avez la possibilté d'allumer 25 encens batons, ou un encens baton suivi d'un encens spirale, ou encore un encens spirale suivi d'un encens baton.

Donnez un programme dynamique de complexité en temps $O(n)$ pour ce problème.

Des cas de base et une récurence pour A_n sont demandés, pas un code Python par exemple.

2 Rendre la monnaie

Considérons le problème classique de rendu de monnaie suivant. Vous disposez d'une caisse avec des pièces de monnaie en quantité illimitée. Chaque pièce a une des valeurs $v_1, \ldots, v_k$, pour des entiers $v_1, \ldots, v_k$ donnés. Vous devez rendre la monnaie à un client, la somme de B unités pour être précis, et vous voulez le faire en lui rendant le moins de pièces possibles.

Considérons l'algorithme glouton suivant pour ce problème, qui tant que B est positif, donne au client une pièce de plus grande valeur v_i avec $v_i \leq B$ et décremente B de v_i.

Montrez que cet algorithme n'est pas optimal pour $v_1 = 1, v_2 = 5$, et $v_3 = 7$ ($k = 3$).

Montrez que cet algorithme est optimal pour des pièces de valeur

$$v_i = \begin{cases} 1 \cdot 10^j & \text{si } i = 3j + 1 \\ 2 \cdot 10^j & \text{si } i = 3j + 2 \\ 5 \cdot 10^j & \text{si } i = 3j + 3. \end{cases}$$

3 Kapla

Vous jouez à une variante de Kapla qui consiste en des blocs en bois de longueurs différentes, mais de largeur et profondeur identique et de densité uniforme et identique. On vous donne une description d'une construction à réaliser. Ce plan est décrit par une série de nombres $a_1, b_1, \ldots, a_n, b_n$. Il décrit un empilement de blocs, tous alignés sur l'axe x, tel que le i-ème bloc a une longueur $b_i - a_i$ et couvre l'intervalle $[a_i, b_i]$ de l'axe x. Les blocs sont numérotés de 1 à n, en partant du bas.

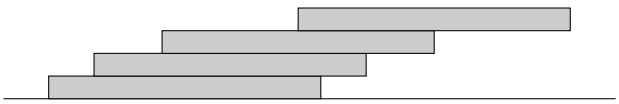
Maintenant on vous demande de déterminer la stabilité de cette construction. Celle-ci peut être stable, instable ou bancale, comme expliqué ci-dessous. Soit c_i la coordonnée x du centre de gravité de l'ensemble des blocs i à n, pour tout $2 \leq i \leq n$. Si $c_i \notin [a_{i-1}, b_{i-1}]$ pour un indice $i \geq 2$, alors la construction est instable. Si par contre $a_i < c_i < b_i$ pour tout $i \geq 2$, alors elle est stable. Et dans le cas échéant elle est bancale.

Un exemple, un exemple

D'accord, d'accord, voici des exemples.

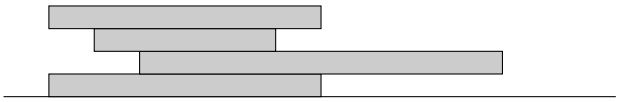
Bancale

i	a_i	b_i
4	11	23
3	5	17
2	2	14
1	0	12



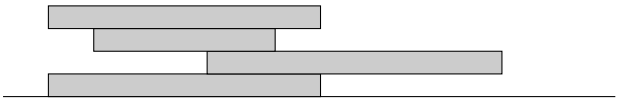
Stable

i	a_i	b_i
4	0	12
3	2	10
2	4	20
1	0	12



Instable

i	a_i	b_i
4	0	12
3	2	10
2	7	20
1	0	12



4 Planification de production

Vous disposez de n tâches à distribuer parmi m travailleurs. Les tâches sont numérotées de 1 à n, et les travailleurs de 1 à m. Le travailleur i est qualifié seulement pour les tâches de l'ensemble $S_i \subseteq \{1, \ldots, n\}$. Le but est d'affecter à chaque travailleur i un ensemble de tâches $T_i \subseteq S_i$, tel que tout $k \in \{1, \ldots, n\}$ appartient à exactement un ensemble T_i.

Le coût d'une affectation se calcule comme suit. Chaque tâche k affectée au travailleur i vous coutera a_i Euros tant que sa capacité c_i n'est pas dépassée et b_i Euros sinon, où $b_i > a_i$ inclu le prix des heures supplémentaires du travailleur i. Formellement le coût de l'affectation est

$$\sum_{i=1}^{n} a_i \cdot \min\{|T_i|, c_i\} + b_i \cdot \min\{0, |T_i| - c_i\}.$$

Modélisez ce problème comme une instance à un problème de flot de coût minimum. Ce dernier problème consiste en un graphe orienté $G(V, A)$, d'un sommet source $s \in V$, d'un sommet puits $t \in V$, d'une demande $d \in \mathbb{N}$, d'une capacité $f_{\max} : A \to \mathbb{R}^+$ et d'un coût par unité de flot $c : A \to \mathbb{R}^+$. Le but est de calculer un flot $f : A \to \mathbb{R}^+$ minimisant $\sum_{a \in A} c(a)f(a)$ tel que

$$\forall a \in A : 0 \leq f(a) \leq f_{\max}(a)$$

$$\forall v \in V : \sum_{a=(u,v) \in A} f(a) - \sum_{a=(v,u) \in A} f(a) = \begin{cases} d & \text{si } v = t \\ -d & \text{si } v = s \\ 0 & \text{sinon.} \end{cases}$$

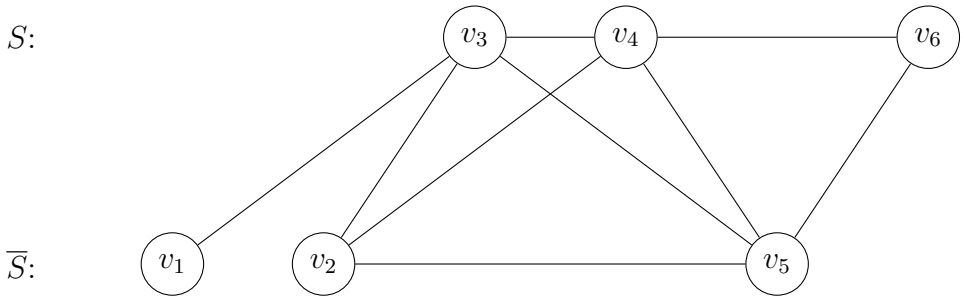
La première contrainte traduit le respect des capacités et la deuxième le respect de la demande de la valeur du flot.

5 Coupe maximale dans un graphe

Étant donné un graphe non orienté $G(V, E)$ une coupe est un ensemble $S \subseteq V$ et sa valeur est le nombre d'arêtes ayant une extrémité dans S et l'autre en dehors de S. Le but est de trouver une coupe plus grande valeur. Ce problème est NP-complet. Cependant il existe un algorithme très simple qui approxime ce problème.

Soit $V = \{v_1, \ldots, v_n\}$. Soit $S = \emptyset$ intialement. Pour chaque $i = 1, \ldots, n$, considérez les voisins v_j de v_i avec $j < i$. Si la majorité stricte n'est pas dans S, alors ajoutez v_i à S.

Sur l'exemple suivant l'algorithme calcule une coupe de valeur 6.



Montrez que l'algorithme produit toujours une coupe de valeur au moins la moitié de la coupe optimale.

A Corrections

A.1 Encens

Cas de base $A_n = 1$ pour tout $n = 0, \ldots, 24$ et $A_n = A_{n-1} + A_{n-24}$ pour tout $n \geq 24$.

A.2 Rendre la monnaie

Pour $B = 11$, glouton rendera 7+1+1+1+1 alors que l'optimum consite en 3 pièces 5+5+1.

Considérons une solution optimale. Elle peut contenir au plus une pièce de la forme 10^j, car sinon on pourrait remplacer deux de ces pièces par une seule de valeur $2 \cdot 10^j$, contredisant l'optimalité. Par le même argment elle peut contenir au plus deux pièces de la forme $2 \cdot 10^j$ et au plus une de la forme $5 \cdot 10^j$. Ceci implique que l'optimum est décrit de manière extrèmement simple. Pour chaque valeur $v \cdot 10^j$ dans la décomposition décimale de B, la solution optimale contient les pièces suivantes (le facteur 10^j a été omis pour plus de simplicité).

v	OPT
0	ϵ
1	1
2	2
3	$2+1$
4	$2+2$
5	5
6	$5+1$
7	$5+2$
8	$5+2+1$
9	$5+2+2$.

On observe que c'est également la solution produite par glouton. Si $v \geq 5$, glouton produit d'abord la pièce 5. Maintenant on est ramené dans le cas $v < 5$, car l'autre cas est symmétrique, etc.

A.3 Kapla

Le centre de gravité c_i est une valeur x tel que

$$\sum_{j=i}^{n} \max\{0, x - a_j\} = \sum_{j=i}^{n} \max\{0, b_j - x\}.$$

La différence des deux côtés de l'équation est linéaire par morceaux et convexe, on peut donc déterminer c_i par recherche dychotomique. Mais il y a une autre observation à faire.

La somme des deux côtés est indépendante de x, elle vaut

$$\sum_{j=i}^{n} b_j - a_j. \qquad \text{(total)}$$

La partie gauche est monotone croissante en x. On peut donc par un algorithme de balayage déterminer la valeur x tel que la partie gauche devient la moitié de (total).

Cependant il existe une manière plus directe de calculer le centre de gravité par programmation dynamique. Soit m_i la masse totale des blocs i à n et c_i la coordonnée x de leur centre. Pour simplfier notons $m_{n+1} = 0$.

Le bloc i a la masse $b_i - a_i$, et le centre $(b_i + a_i)/2$. Le centre des blocs i à n est la moyenne pondérée, calculée comme suit

$$\begin{aligned} m_i &= m_{i-1} + b_i - a_i \\ c_i &= \frac{c_{i-1} \cdot m_{i-1} + \frac{b_i + a_i}{2} \cdot (b_i - a_i)}{m_i}. \end{aligned}$$

Une fois ces valeurs calculées (ce qui se fait en temps linéaire), c'est juste une question de vérifier les inégalités $a_i \leq c_i \leq b_i$ et d'observer si l'une d'elle est satisfaite avec égalité.

Si vous avez aimé le problème et particulièrement l'exemple bancal, je vous invite à lire : *Overhang*, by Mike Paterson and Uri Zwick. The American Mathematical Monthly, Vol. 116, No. 1 (Jan., 2009), pp. 19-44.

A.4 Planification de production

On construit un graphe $G(V, A)$, comme suit. Il y a un sommet v_j pour chaque tâche et deux sommets u_i, u'_i pour chaque travailleur. Pour chaque $j \in S_i$ il a un arc (v_j, u_i) de capacité 1 et de coût a_i et un arc (v_j, u'_i) de capacité 1 et de coût b_i. Finalement il a un arc (s, v_j) de capacité 1 et coût 0 pour toute tache j. Pour tout travailleur i, il y a un arc (u_i, t) de capacité c_i et coût 0, et un arc (u_i, t) de capacité infinie et de coût 0. La demande est fixé à $d = n$. Ceci est juste une parmi les modélisations possibles.

A.5 Coupe maximale

Soit S une coupe arbitraire. Notons d_i le nombre de voisins v_j de v_i avec $j < i$. Notons $a_i(S)$ le nombre de voisins v_j de v_i avec $j < i$ tel que l'arête (v_j, v_i) compte dans la valeur de la coupe. La valeur de la coupe est $\sum a_i(S)$. Comme $a_i(S) \leq d_i$, la valeur de la coupe ne peut dépasser $\sum d_i$. L'algorithme par contre produit une coupe A avec $a_i(A) \geq d_i/2$, et donc a une valeur au moins $\sum d_i/2$.