ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Section d'Informatique et de Systèmes de Communication

Corrigé de la série 7 9 Novembre 2009

1. Sac à dos rationnel

a) Intuitivement, on sent que l'algorithme va nous donner une solution optimale, il est plus difficile de le prouver formellement. Nous allons montrer que toute solution autre que celle obtenue par l'algorithme sera moins bonne (c'est-à-dire que la valeur totale sera inférieure).

Supposons d'abord (sans perdre de généralité) que nos objets sont ordonnés selon leur valeur relative (i.e. prix au kilo) $\frac{v_i}{w_i}$ de façon décroissante. On a donc:

$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \ldots \geq \frac{v_n}{w_n} \tag{1}$$

Soit $S_1 = (x_1, \ldots, x_n)$ la solution obtenue avec l'algorithme glouton, c'est à dire qu'on prend une fraction x_i du produit i.

L'algorithme glouton va commencer par prendre autant de que possible du produit 1 (puisque c'est celui qui a le plus de valeur par kilo), lorsqu'il ne reste plus de produit 1 il prendra autant que possible du produit 2 etc.. Il s'arrête lorsque le poids total aura atteint W. Si $x_i = 1$ pour tout i, clairement cette solution est optimale. Sinon, il y a donc un produit m pour lequel il va prendre une quantité $x_m < 1$ et s'arrêter (puisqu'il aura atteint un poids total de W). L'algorithme ne prendra donc rien des produits $x_{m+1}, \ldots, x_n$. On a donc:

- $\forall i < m, x_i = 1$
- $\forall i > m, x_i = 0$
- $\sum_{i=1}^{n} x_i w_i = W$

Soit $S_2 = (y_1, \ldots, y_n)$ une solution quelconque, on a donc:

$$\sum_{i=1}^{n} y_i w_i \leq W = \sum_{i=1}^{n} x_i w_i \Longrightarrow \sum_{i=1}^{n} (x_i - y_i) w_i \geq 0 \tag{2}$$

On remarque aussi que

- Si $i < m$ donc $x_i = 1$ et on a $x_i - y_i \geq 0$ et $\frac{v_i}{w_i} \geq \frac{v_m}{w_m}$
- Si $i > m$ donc $x_i = 0$ et on a $x_i - y_i \leq 0$ et $\frac{v_i}{w_i} \leq \frac{v_m}{w_m}$

Et on remarque que dans les deux cas on aura

$$(x_i - y_i)\frac{v_i}{w_i} \geq (x_i - y_i)\frac{v_m}{w_m} \tag{3}$$

Calculons la différence entre la valeur totale de S_1 et de S_2:

$$\begin{aligned}
\text{val}(S_1) - \text{val}(S_2) &= \textstyle\sum_{i=1}^{n} x_i v_i - \sum_{i=1}^{n} y_i v_i \\
&= \textstyle\sum_{i=1}^{n} (x_i - y_i) w_i \frac{v_i}{w_i} \\
&\geq \textstyle\sum_{i=1}^{n} (x_i - y_i) w_i \frac{v_m}{w_m} \\
&= \textstyle\frac{v_m}{w_m} \sum_{i=1}^{n} (x_i - y_i) w_i \\
&\geq 0,
\end{aligned}$$

On passe de la $2^{\text{ème}}$ à la $3^{\text{ème}}$ ligne en utilisant (3) et de la $4^{\text{ème}}$ à la $5^{\text{ème}}$ ligne en utilisant (2).

Ainsi la solution S_1 obtenue par l'algorithme glouton est au moins aussi bonne que toute autre solution $S_2 = (y_1, \ldots, y_n)$, elle est donc optimale.

b) On commence par calculer et classer les valeurs relatives des produits:

produit	w	v	v/w	classement
A	200	1600	8	3
B	10	100	10	2
C	30	450	15	1
D	100	600	6	4

On prend d'abord autant que possible du produit 1 (C), et puisque $w_1 < W$ on peut prendre tout ce qui est disponible, donc 30 grammes. Ensuite on veut prendre du produit 2 (B), encore une fois, puisque $w_1 + w_2 < W$ on prend la totalité des 10 grammes disponibles. Nous voulons ajouter autant que possible de produit 3 (A) aux 40 grammes que nous avons dejà, nous en prenons donc 10 grammes et nous avons atteint la limite de 50 grammes.

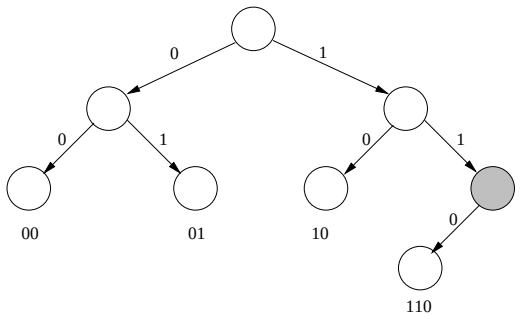
La solution optimale est donc $(30, 10, 10, 0)$ (en prenant les produits dans l'ordre donné par le classement dans le tableau ci-dessus), et on obtient une valeur totale de $30 \cdot 15 + 10 \cdot 10 + 10 \cdot 8 = 630\$$.

2. Le codage de Huffman

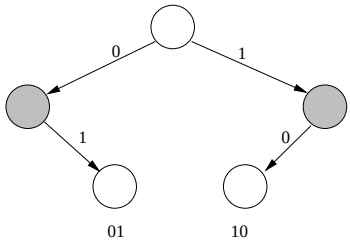
Le codage de Huffman est une méthode de construction d'un code optimal pour un alphabet avec des fréquences données. Autrement dit, le codage de Huffman nous fournit un code tel que la longueur moyenne des mots est minimale pour l'alphabet et les fréquences données.

a) Le code $(0, 10, 11)$ est un code de Huffman pour l'alphabet (a, b, c) et les fréquences $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$.

b) Le code $(00, 01, 10, 110)$ n'est pas un code de Huffman: en dessinant l'arbre correspondant, on voit qu'il y a un sommet interne qui n'a pas deux fils, ce qui n'arrive pas, par construction. Voici l'arbre correspondant à ce code avec le sommet en question dessiné en gris:

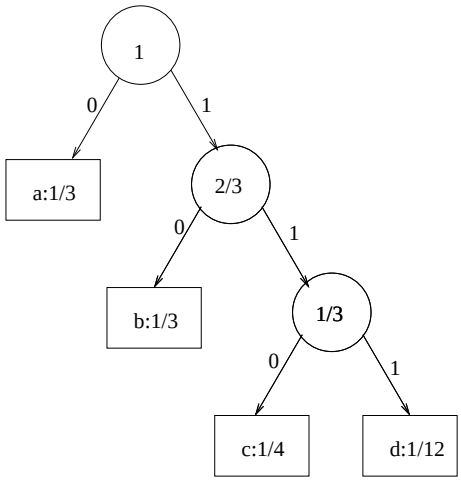


c) De même, le code $(01, 10)$ n'est pas de Huffman. L'arbre correspondant, avec les sommet internes n'ayant pas deux fils marqués en gris:



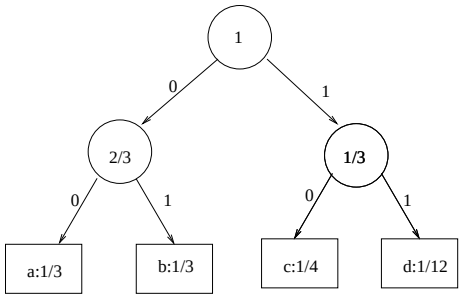
3. Le code optimal n'est pas unique

a) On peut construire un code de Huffman C_1 comme suit:



Les longueurs des mots de ce code sont $(1, 2, 3, 3)$.

b) Un autre arbre de Huffman C_2 pour le même alphabet et les mêmes fréquences est:



Les longueurs de ce code sont $(2, 2, 2, 2)$.

c) La longueur moyenne de C_1 est

$$\bar{L}_1 = 1 \cdot \frac{1}{3} + 2 \cdot \frac{1}{3} + 3 \cdot \frac{1}{4} + 3 \cdot \frac{1}{12} = 2.$$

Celle de C_2 est

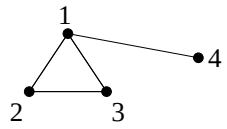
$$\bar{L}_2 = 2 \cdot \frac{1}{3} + 2 \cdot \frac{1}{3} + 2 \cdot \frac{1}{4} + 2 \cdot \frac{1}{12} = 2.$$

Les deux ont la même longueur moyenne et les deux sont optimaux.

4. Problème de la clique

a) Quelques exemples: $\{2, 4, 5\}, \{5, 9, 10\}, \{4, 5, 6, 7\}\{4, 5, 6\}, \{4, 5, 7\}, \{5, 6, 7\}$.

b) Considérons le graphe suivant:

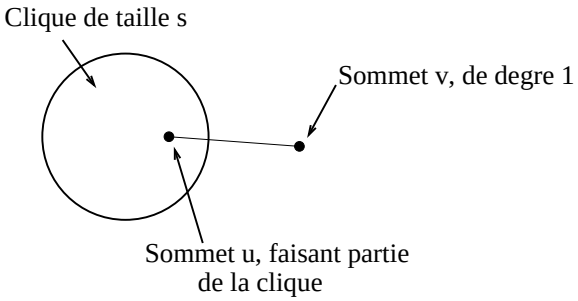


La plus grande clique est $\{1, 2, 3\}$, de taille 3. Considérons cependant le parcours suivant de FindClique:

- $U = \{1, 2, 3, 4\}$ au départ.
- Si l'algorithme choisit à la ligne 3 le sommet 2 (possible puisque 2 n'est pas connecté à 4), alors nous passons à $U = \{1, 3, 4\}$
- S'il choisit ensuite le sommet 3 nous passons à $U = \{1, 4\}$, qui est une clique.

L'algorithme retourne donc la clique $\{1, 4\}$ qui n'est pas la plus grande.

c) Nous généralisons l'idée ci-dessus. Considérons le graphe suivant avec $s + 1$ sommets:



Si FindClique choisit les sommets u et v en dernier il retournera une clique de taille 2, à savoir $\{u, v\}$.