ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
Section d'Informatique et de Systèmes de Communication

Corrigé de la série 6 26 October 2009

1. Algorithmes par induction: Le problème de la célébrité

On précise que dans nos graphes non-orientés on suppose qu'un sommet n'est jamais connecté à lui-même, c'est-à-dire que la diagonale de la matrice d'adjacence ne contient que des 0.

a) Le sommet 2 est une célébrité puisqu'il est connu de tout le monde mais ne connait personne (son in-degree vaut $n-1$ alors que son out-degree vaut 0).

b) Supposons que nous avons un graphe avec deux célébrités (que nous appelons les sommets i et j). i ne connait personne, donc en particulier i ne connait pas j. j ne peut donc pas être une célébrité et nous avons une contradiction. Un graphe ne peut donc pas avoir deux célébrités.

c) On rappelle que pour tout graphe orienté, avec comme matrice d'adjacence A, A_{ij} vaut 1 s'il y a un arc du sommet i au sommet j, et 0 sinon. Ainsi la ligne k montre les arcs qui sortent du sommet k, et la colonne k ceux qui vont vers k.

Si le sommet k est une célébrité, la ligne k ne contient donc que des 0. De même, la colonne k ne contient que des 1, sauf sur la diagonale (l'entrée A_{kk}) qui vaut 0 (puisque nous supposons qu'il n'y a jamais d'arc d'un sommet à lui-même).

d) L'algorithme naïf serait de prendre chaque sommet un par un et de regarder s'il s'agit d'une célébrité. Etant donné un sommet k, pour déterminer si c'est une célébrité nous devons:

- (1) vérifier que tous les autres sommets connaisent k
- (2) vérifier qu'aucun autre sommet n'est connu de k

Pour (1) nous devons vérifier que $A_{ik} = 1$ pour tout $i \in \{1, \ldots, n\} \backslash \{k\}$ *($n-1$ questions).* Pour (2) nous devons vérifier que $A_{kj} = 0$ pour tout $j \in \{1, \ldots, n\} \backslash \{k\}$ *($n-1$ questions).* Pour vérifier si k est une célébrité il nous faut donc $2 \cdot (n-1)$ questions.

Puisque nous devons répeter cette procedure pour tous les $k \in \{1, \ldots, n\}$ on obtient un total de $2 \cdot n \cdot (n-1)$ questions dans le pire des cas, donc un algorithme $O(n^2)$.

e) Rappelons qu'on sait qu'il existe une célébrité dans le graphe qui nous est donné.

- *(i)* Si on a deux sommets a et b on peut poser la question "a connait-il b?". Si la réponse est "oui" on sait que a ne peut pas être une célébrité, et puisque l'un des deux doit être une célébrité, il s'agit forcément de b. De même, si la réponse est "non", on sait que b ne peut pas être une célébrité, et donc la célébrité est a.
- *(ii)* Nous supposons qu'étant donné un groupe de n personnes contenant une célébrité, nous pouvons la retrouver en posant $T(n)$ questions.
  Etant donné un groupe de $n+1$ personnes nous procédons comme suit:

- nous choisissons deux personnes a et b parmi les $n+1$.
- nous posons la question "a connait-il b?"
- Si la réponse est "oui" nous savons que a n'est pas la célébrité. Celle-ci se trouve donc parmi les n autres personnes (dont b). Nous pouvons donc la retrouver en posant $T(n)$ questions. Nous avons donc dû poser $T(n)+1$ questions au total.
- Si la réponse est "non" nous savons que b n'est pas la célébrité. Nous pouvons donc de la même façon la retrouver parmi les n personnes restantes en posant $T(n)$ questions. Nous avons donc dû poser $T(n)+1$ questions au total.

- *(iii)* L'algorithme récursif a été décrit dans la partie (*ii*). En pseudo-code on obtient:

Call: FindCelebrity
Input: $S = \{s_1, \ldots, s_n\}$: Un ensemble de n personnes, dont une célébrité
Output: c: la célébrité
if $n = 2$ **then**
  if s_1 connait s_2 **then**
  else
if s_1 connait s_2 **then**
else

Si on appelle $T(n)$ le nombre de questions dont il a besoin pour trouver la célébrité parmi n personnes on a donc $T(2) = 1$ et $T(n) = T(n-1) + 1$ pour $n \geq 2$. Il est assez clair que cela nous donne

$$T(n) = n - 1$$

En effet par récurrence on voit que l'égalité est vraie pour $n = 2$ (base), et si elle est vraie pour n alors

$$T(n+1) = T(n) + 1 = (n-1) + 1 = n$$

Ainsi c'est un algorithme $O(n)$.

2. Sac à dos 0/1

a) La valeur maximale est 45. Il y a deux solutions correctes: les ensembles qui atteignent ce maximum sont:

$$\{A, C, E\} \qquad \text{et} \qquad \{B, D, E\}$$

Pour les détails voir la figure 1.

Les flèches diagonales corréspondent à la branche passant par le pas 9 de l'algorithme du polycopié et les flèches verticales aux branches passant par 11 ou 14. Les flèches en gris sont les décisions rejetées parce que l'autre chemin est mieux.

Les flèches en gras donnent des chemins optimaux (flèche diagonale à la $i$ème étape: on prend l'objet i; flèche verticale: on ne le prend pas).

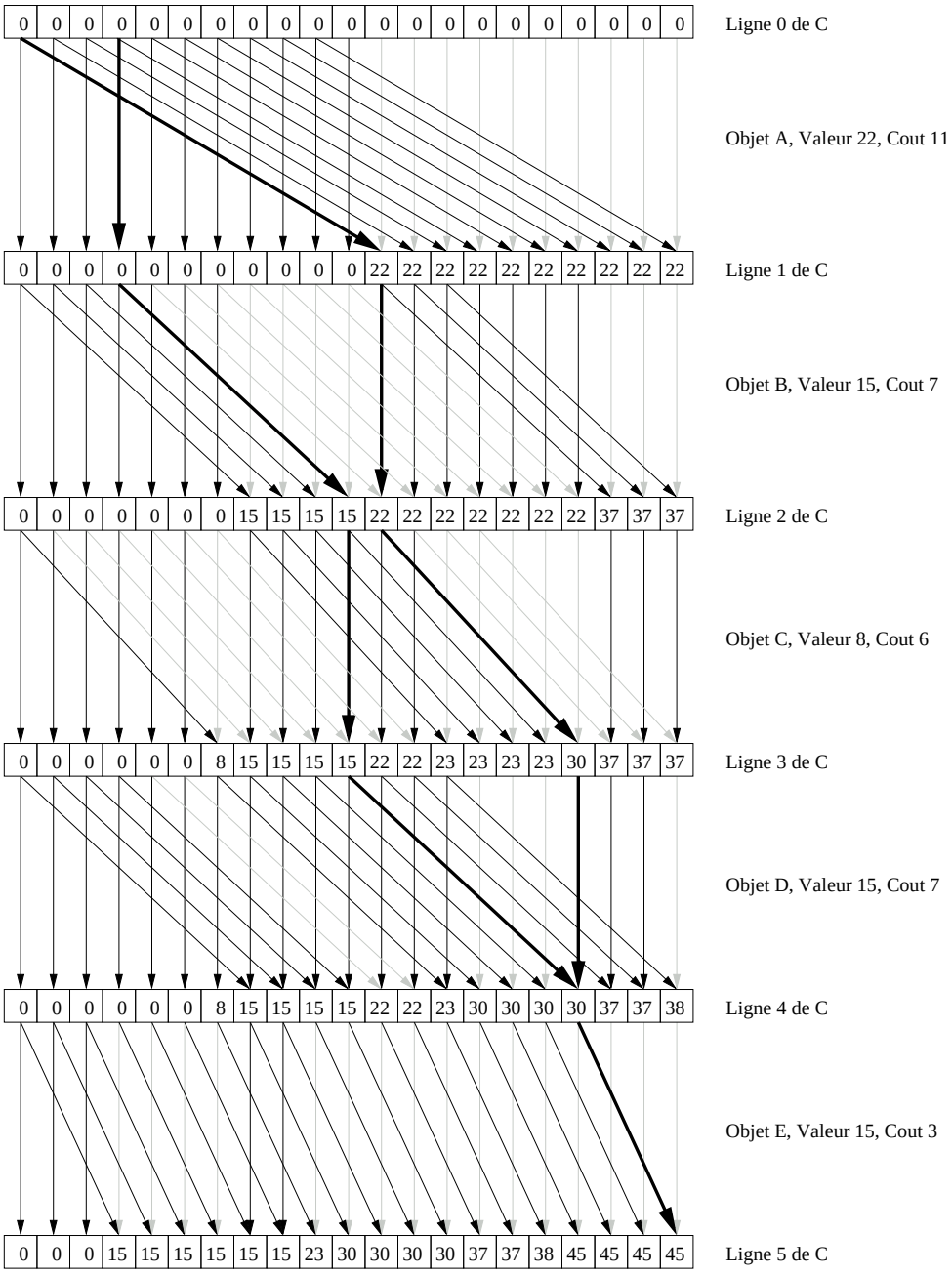

Figure 1: Knapsack 0/1

b) La solution optimale n'est pas unique, c.f. le corrigé du point précédent. On peut adapter l'algorithme pour trouver toutes les solutions optimales. La méthode la plus simple est d'utiliser une récursion pour le faire (Initialement, nous appélons GetOptimalChoice2(W, n, C)).

```
Call: GetOptimalChoice2(w, i, C)
Input: L'input à Knapsack et le C calculé. x[] est statique.
Output: Un choix optimal.
  if i ≥ 1 then
    if c_{i-1,w} = c_{i,w} then
      x[i] ← 0
      GetOptimalChoice2(w, i − 1, C)
    if c_{i-1,w-w_i} + v_i = c_{i,w} then
      x[i] ← 1
      GetOptimalChoice2(w − w_i, i − 1, C)
  else
    Print "Solution:" x[]
```

c) Chaque solution (optimale ou non, admissible ou non) est un sous-ensemble de

$$\{1, 2, \ldots, n\}.$$

Donc, l'affirmation est vraie parce que $|\text{Pot}(\{1, 2, \ldots, n\})| = 2^n$.

d) On peut choisir $v_i = 1$ et $w_i = 1$ pour $i = 1, \ldots, n$ et $W = n/2$. Les solutions optimales sont alors des sous-ensembles de $\{1, \ldots, n\}$ de taille $n/2$, et il y en a $\binom{n}{n/2}$. Il est facile de vérifier que

$$\binom{n}{n/2} > 2^{n/2},$$

d'où le résultat.

3. Multiplication de plusieurs matrices

Rappelons que $m_{i,j}$ est le nombre minimal de multiplications scalaires pour le calcul de $A_{i\ldots j}$ et on a:

$$m_{ij} = \begin{cases} 0 & \text{si } i = j = 0, \\ \min_{i \le k < j}\{m_{ik} + m_{k+1,j} + p_{i-1} \cdot p_k \cdot p_j\} & \text{si } 1 \le i \le j \le n. \end{cases}$$

Rappelons aussi que s_{ij} est la valeur de k pour laquelle le minimum de la formule est

atteint. Considerons que $M[i,j] = m_{ij}$ et $S[i,j] = s_{i,j}$.

$$\begin{aligned}
m_{1,2} &= 5 \cdot 10 \cdot 3 = 150 \\
m_{2,3} &= 10 \cdot 3 \cdot 12 = 360 \\
m_{3,4} &= 3 \cdot 12 \cdot 5 = 180 \\
m_{1,3} &= \min \begin{cases} m_{1,2} + 5 \cdot 3 \cdot 12 = 330 \\ m_{2,3} + 5 \cdot 10 \cdot 12 = 960 \end{cases} \\
m_{2,4} &= \min \begin{cases} m_{2,3} + 10 \cdot 12 \cdot 5 = 960 \\ m_{3,4} + 10 \cdot 3 \cdot 5 = 330 \end{cases} \\
m_{1,4} &= \min \begin{cases} m_{2,4} + 5 \cdot 10 \cdot 5 = 580 \\ m_{1,2} + m_{3,4} + 5 \cdot 3 \cdot 5 = 405 \\ m_{1,3} + 5 \cdot 12 \cdot 5 = 630 \end{cases}
\end{aligned}$$

Les matrices M et S sont alors:

$$M = \begin{pmatrix} 0 & 150 & 330 & 405 \\ & 0 & 360 & 330 \\ & & 0 & 180 \\ & & & 0 \end{pmatrix} \qquad S = \begin{pmatrix} 0 & 1 & 2 & 2 \\ & 0 & 2 & 2 \\ & & 0 & 3 \\ & & & 0 \end{pmatrix}$$

$S[1,4] = 2$ signifie que $A_1 \cdot A_2 \cdot A_3 \cdot A_4$ est construit de manière optimale à partir de $A_1 \cdot A_2$ et $A_3 \cdot A_4$. Le nombre de multiplications nécessaires est donc 405.