

# Enseirb, Filière Electronique, Semestre 1

## Algorithmes et Structures de Données

Epreuve proposée par Robert Cori

20 janvier 2008, durée 1 heure 30

*Le polycopié, les notes de cours et de travaux dirigés sont autorisées, à l'exclusion de tout autre document.*

### **Exercice 1 : Algorithme glouton. Barème envisagé : 10 points**

On s'intéresse au problème suivant : afin de réaliser un produit une entreprise doit acquérir tous les objets d'un ensemble de  $n$  éléments  $A = \{e_1, e_2, \dots, e_n\}$ , ces objets ne sont pas vendus séparément mais par des paquets chaque paquet constituant un sous-ensemble  $S_i$  de  $E$ .

Chacun de ces paquets a un coût  $c_i$  et on doit déterminer le coût minimal permettant d'acquérir tous les objets. Un exemple est le suivant :

$$S_1 = \{1, 2\}, S_2 = \{3, 4\}, S_3 = \{1, 3, 4\}, S_4 = \{1, 4\}, S_5 = \{2, 4\}$$

avec les coûts :

$$c_1 = 5, c_2 = 3, c_3 = 5, c_4 = 4, c_5 = 2$$

Plusieurs solutions sont possibles par exemple :

- Acquérir  $S_1$  et  $S_2$  dont le coût total est 8
- Acquérir  $S_1$  et  $S_3$  de coût 10
- Acquérir  $S_2$ ,  $S_4$  et  $S_5$  de coût 9
- Mais la meilleure solution consiste à retenir  $S_3$ , et  $S_5$  de coût 7.

Un premier algorithme glouton consiste à classer les paquets par coûts moyens croissants ; le coût moyen d'un sous-ensemble  $S_i$  est le quotient de son coût par son nombre  $n_i$  d'éléments. On pose ainsi  $u_i = \frac{c_i}{n_i}$ . Par cet algorithme on retient d'abord l'ensemble  $S_i$  tel que  $u_i$  est minimum, puis à chaque étape on retient un nouvel ensemble ayant la plus petite valeur de  $u_i$  parmi ceux qui n'ont pas été retenus et ceci jusqu'à obtenir que chaque  $e_j$  soit dans un  $S_i$  retenu.

**Question 1.** Quels sont les sous-ensembles retenus par l'algorithme glouton dans l'exemple considéré plus haut ? Cet algorithme glouton donne-t-il l'optimum ?

**Question 2.** On suppose que les ensembles  $S_i$  sont donnés par un tableau (noté `appartient`) à deux indices et à valeurs booléennes. Ainsi `appartient[j][i]` est égal à `true` si et seulement si l'élément  $e_j$  appartient à l'ensemble  $S_i$ . On vous demande d'écrire un algorithme détaillé du calcul du tableau `nb` tel que `nb[i]` soit le nombre d'éléments de l'ensemble  $S_i$ .

On modifie l'algorithme glouton de façon à ne pas tenir compte au cours des étapes des éléments déjà acquis. Ainsi après avoir sélectionné un  $S_j$ , on recalcule le coût moyen de chaque ensemble  $S_i$  en ne tenant compte que des éléments qui n'appartiennent pas à un  $S_k$  déjà retenu. On pose alors

$$u_i = \frac{c_i}{n'_i}$$

où  $n'_i$  est le nombre d'éléments de  $S_i$  qui ne sont pas dans un ensemble déjà retenu.

**Question 3.** Appliquer le nouvel algorithme à l'exemple donné plus haut : vous donnerez la valeur des  $n'_i$  une fois qu'un premier sous-ensemble a été retenu ainsi que la valeur des  $u_i$  après la première étape. Qu'en est-il de la deuxième étape.

**Question 4.** Ecrire en détail l'algorithme

`reCalcule(nb, j)`

qui modifie la valeur des `nb[i]` une fois retenu l'ensemble  $S_j$  (d'indice `j`), en utilisant le tableau `appartient`.

**Question 5.** Donner un exemple pour lequel ce nouvel algorithme ne donne pas le résultat optimal.

## Exercice 2 *Barème envisagé : 10 points*

On se propose de calculer une plus longue sous-suite croissante d'une suite de nombres en utilisant des techniques inspirées de la programmation dynamique.

Pour ceci on commence par un algorithme simple qui sera utilisé par la suite.

**Question 1.** Soit  $u$  un tableau de  $n$  nombres entiers ( $u[0], u[1], \dots, u[n-1]$ ) tous distincts. On vous demande d'écrire en détail la fonction `precedent(i)` (où  $i$  est un indice compris entre 0 et  $n-1$ ), dont le résultat est  $j$ , indice de l'élément du tableau situé avant  $u[i]$  inférieur ou égal à  $u[i]$  et le plus proche de lui. Le résultat doit être -1 si tous les éléments situés avant  $u[i]$  lui sont supérieurs. Ainsi on doit avoir (si  $u[i] = j$  est différent de -1) :

$$j < i \quad u[j] < u[i] \quad u[k] > u[i] \text{ pour } j < k < i$$

Par exemple, pour le tableau 6, 2, 5, 4, 9, 7, 8 on a  $u[3] = 4$ ; ainsi `precedent(3)` = 1, car  $u[1] = 2 < 4$  et  $u[2] = 5 > 4$ . On a aussi `precedent(1)` = -1.

**Question 2.** Une sous-suite croissante d'une suite  $u_0, u_1, \dots, u_{n-1}$  est donnée par  $u_{i_1}, u_{i_2}, \dots, u_{i_p}$  tels que

$$i_1 < i_2 \dots < i_p \quad u_{i_1} < u_{i_2} < \dots < u_{i_p}$$

par exemple 2, 5, 9 est une sous-suite croissante de la suite considérée à la Question 1 mais ce n'est pas la plus longue car 2, 5, 7, 8 est plus longue.

On se propose de donner un algorithme qui calcule une sous-suite de longueur maximale. Tout d'abord on calcule cette longueur; pour cela on note  $\ell_i$  la longueur de la plus longue sous-suite croissante dont le dernier élément est  $u_i$ , quelle est la valeur de  $\ell_i$  si `precedent(i)` = - 1? On suppose calculé  $\ell_j$ , quelle est la valeur de  $\ell_i$  si `precedent(i)` =  $j$ ?

**Question 3.** Donner un algorithme qui calcule les  $\ell_i$  en utilisant la fonction `precedent` que vous avez donné à la Question 1.

**Question 4.** Comment compléter l'algorithme précédent afin qu'il affiche une suite de longueur maximale?