

# Enseirb Semestre 1

## Algorithmes et Structures de Données

Epreuve proposée par Robert Cori

22 janvier 2008, durée 1 heure 30

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

### **Exercice 1 : Récursivité. Barème envisagé : 4 points**

On considère la fonction récursive  $f$  sur l'ensemble des entiers positifs donnée par :

```
f(n) {  
    if (n == 1) return 0;  
    if (n == 2) return 1;  
    else return f(n-f(n-1)) + f(n-1 - f(n-2));  
}
```

**Question 1.** calculer les valeurs de  $f(n)$  pour les valeurs suivantes de  $n$  :  $n = 3$ ,  $n = 4$ ,  $n = 5$ ,  $n = 6$ .

**Question 2.** Ecrire une version itérative du calcul de  $f(n)$  qui pour obtenir ce résultat, remplit un tableau (`valf[ ]`) par les valeurs de  $f(i)$  pour  $1 \leq i < n$ .

## Exercice 2 : Piles. Barème envisagé : 6 points

On considère des suites finies de nombres  $u_0, u_1, \dots, u_{n-1}$  que l'on va traiter à l'aide de deux piles  $P$  et  $Q$ .

Pour cela on utilise un indice  $k$  initialisé à 0, ainsi que les 5 opérations élémentaires suivantes :

- $X$  Affiche  $u_k$ , puis incrémente  $k$ .
- $A$  Ajoute  $u_k$  à la pile  $P$  puis incrémente  $k$ .
- $\bar{A}$  Affiche la valeur en tête de la pile  $P$  et supprime cet élément de  $P$ .
- $B$  Ajoute  $u_k$  à la pile  $Q$  puis incrémente  $k$ .
- $\bar{B}$  Affiche la valeur en tête de la pile  $Q$  et supprime cet élément de  $Q$ .

Ainsi, pour la suite de nombres 13, 17, 8, 4, 10, les opérations suivantes (effectuées dans l'ordre) :

$$A, B, A, X, \bar{A}, X, \bar{A}, \bar{B}$$

affichent 4, 8, 10, 13, 17.

Comme les nombres sont affichés dans l'ordre croissant, on dit que la suite de nombres est *triée par la suite d'opérations*.

**Question 1.** Quelles sont les valeurs affichées pour la même suite de nombres pour la suite d'opérations

$$A, X, \bar{A}, A, B, X, \bar{A}, \bar{B}$$

**Question 2.** Donner une suite d'opérations qui permet de trier (d'afficher en ordre croissant) la suite 7, 12, 9, 3, 8.

**Question 3.** Parmi les 24 suites contenant les 4 nombres 1, 2, 3, 4 il y en a une seule que l'on ne peut pas trier à l'aide des deux piles  $P$  et  $Q$  en utilisant les opérations précédentes. Donner cette suite.

### Exercice 3 : Programmation dynamique. Barème envisagé : 10 points

On se donne un tableau  $\mathbf{t}$  de taille  $n$  et on suppose que tous les éléments de  $\mathbf{t}$  valent 0 ou 1. Une coupe  $\mathbf{t}[i..j] = \{ \mathbf{t}[i], \mathbf{t}[i+1], \dots, \mathbf{t}[j] \}$  de  $\mathbf{t}$ , pour  $0 \leq i < j \leq n-1$ , est dite *équilibrée* si elle comporte autant de 0 que de 1.

La longueur de la coupe  $\mathbf{t}[i..j]$  est son nombre d'éléments  $j - i + 1$ . Le but de l'exercice est d'obtenir un algorithme efficace qui calcule la longueur de la plus longue coupe équilibrée d'un tableau.

#### Question 1

1. Ecrire une fonction `equi1(i, j, t)` qui donne pour résultat *vrai* si la coupe  $\mathbf{t}[i..j]$  est équilibrée et *faux* sinon. Quel est le nombre d'opérations de cette fonction ?
2. Utiliser la fonction précédente `equi1` pour écrire une fonction `maxEqui1(t)` qui donne la longueur de la plus longue coupe équilibrée de  $\mathbf{t}$ .

#### Question 2

1. Ecrire une fonction `equi2(t, i)` qui donne la longueur de la plus longue coupe équilibrée de  $\mathbf{t}$  qui commence en  $i$ . On cherche donc

$$\max_j \{ j - i + 1, i \leq j \leq n - 1, \mathbf{t}[i..j] \text{ équilibrée} \}.$$

On veut que le nombre d'opérations de cette fonction `equi2` soit en  $O(n)$ .

2. Utiliser la fonction précédente `equi2` pour écrire une fonction `maxEqui2(t)` qui donne la longueur de la plus longue coupe équilibrée de  $\mathbf{t}$ . Cette fonction est-elle plus efficace que `maxEqui1(t)` ?

**Question 3** On cherche maintenant une fonction `maxEqui3()` qui donne la longueur de la plus longue coupe équilibrée de  $\mathbf{t}$ , et dont le nombre d'opérations soit linéaire en  $n$ . Pour cela, on introduit le déséquilibre `deseq(i, j)` d'une coupe  $\mathbf{t}[i..j]$ , i.e. la différence entre son nombre de 1 et son nombre de 0 :

$$\text{deseq}(i, j) = \text{card}\{k/t[k] = 1, i \leq k \leq j\} - \text{card}\{k/t[k] = 0, i \leq k \leq j\}.$$

Clairement, `deseq(i, j)` est un entier compris entre  $-n$  et  $n$ .

1. La première idée est de parcourir une fois le tableau  $\mathbf{t}$  et de calculer, pour chaque indice  $i$ ,  $0 \leq i \leq n-1$ , le déséquilibre `deseq(0, i)`. Plus précisément, on va remplir un tableau auxiliaire `aux[ ]` de taille  $2n + 1$ , dont les éléments auront auparavant été initialisés à  $-2$ , de la façon suivante :

$$\text{aux}[d+n] = i \text{ si et seulement si } i \text{ est le premier indice tel que } \text{deseq}(0, i) = d.$$

Ecrire une fonction `static void prepare(t)` qui effectue ce travail.

2. On remarque alors que si `deseq(0, i) = deseq(0, j)` pour deux indices  $i$  et  $j$  avec  $i < j$ , alors la coupe  $\mathbf{t}[i+1..j]$  est équilibrée. Expliquer comment modifier la fonction `prepare(t)` pour obtenir la fonction `maxEqui3(t)` qui calcule la longueur de la plus longue coupe équilibrée de  $\mathbf{t}$  avec un coût linéaire en  $n$  (ne pas oublier que le déséquilibre 0 est obtenu pour une coupe vide ; on pourra ainsi poser `aux[n] = -1`).