Année 2007-2008 - Semestre de printemps - 1re session
Étapes : CSB2/MHT2
UE : INF152 - Algorithmes et programmes
Date : 7 mai 2008 - Durée : 1h30
Seul document autorisé : 1 feuille manuscrite recto-verso
Épreuve de : A. Dicky, C. Gavoille, M. Zeitoun

Les candidats doivent traiter les deux exercices. Toutes les fonctions demandées doivent être écrites en langage Python. Le barème est indicatif.

Exercice 1 (7 points)

On rappelle que le *triangle de Pascal* est le tableau des coefficients binomiaux C_n^p :

n \ p	0	1	2	3	4 ...
0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
4	1	4	6	4	1
...					

et se construit en utilisant le fait que $C_n^0 = C_n^n = 1$, et la formule de récurrence

$$C_{n+1}^{p+1} = C_n^p + C_n^{p+1} \text{ pour } 0 \leq p < n.$$

Par exemple, $C_4^1 = C_3^0 + C_3^1 = 1 + 3 = 4$.

Question 1.1 *Écrire une fonction* `suivante(t)` *qui, si* `t` *est la n-ième ligne du triangle de Pascal, retourne la* $n+1$*-ième ligne. Par exemple,* `suivante([1,2,1])` *devra retourner la liste* `[1,3,3,1]`.

Question 1.2 *En utilisant la fonction précédente, écrire une fonction* `pascal(n)` *retournant la liste des* `n` *premières lignes du triangle de Pascal. Par exemple,* `pascal(4)` *devra retourner la liste* `[ [1] , [1,1] , [1,2,1] , [1,3,3,1] ]`.

Question 1.3 *En utilisant la fonction* `pascal`*, écrire une fonction* `C(n,p)` *qui retourne la valeur* C_n^p.

Exercice 2 (13 points)

Le but du problème est de déterminer un élément **médian** d'une liste de n nombres, c'est-à-dire un élément x de la liste tel que la liste contienne autant d'éléments $< x$ que d'éléments $> x$ (à 1 près). Par exemple, la liste `[1,5,2]` a pour élément médian `2`; la liste `[1,5,2,7]` a pour éléments médians `2` et `5`.

Question 2.1 *On souhaite montrer que le médian d'une liste à* $n = 3$ *éléments peut être calculé en faisant au plus trois comparaisons. Écrire une fonction* `median3(a,b,c)` *calculant l'élément médian de la liste* `[a,b,c]` *avec le plus petit nombre de comparaisons possibles.*

Si la liste `t` est triée, un élément médian de `t` est `t[n/2]` (où n est la longueur de `t`). On souhaite trouver cet élément sans avoir à trier complètement la liste.

Pour cela, on résout un problème plus général : étant donné un entier `k` $< n$, déterminer le `k+1`-ième plus petit élément de `t`, c'est-à-dire la valeur qu'aurait `t[k]` après avoir trié la liste `t`.

Question 2.2 *Écrire une fonction* `indiceMini(t,debut,fin)` *retournant l'indice du plus petit élément de la partie de la liste* `t` *comprise entre les indices* `debut` *(inclus) et* `fin` *(exclu) : on suppose qu'on a toujours* `debut` $<$ `fin`*. Si plusieurs éléments sont de valeur minimale, la fonction devra retourner le plus petit de leurs indices.*
Par exemple, si `t = [1,5,4,7,4]`, `indiceMini(t,1,5)` *devra retourner* `2`.

Pour déterminer le `k+1`-ième plus petit élément de `t`, on utilisera la fonction suivante :

```
def selection(t,k,debut,fin):
    if debut >= fin :
        return
    i = indiceMini(t,debut,fin)
    echanger(t,debut,i)
    if k == 0 :
        return t[debut]
    return selection(t,k-1,debut+1,fin)
```

(où `echanger(t,i,j)` échange les éléments d'indices `i` et `j` de la liste `t`).

Question 2.3 *Donner le résultat du programme suivant, en détaillant les appels de fonction effectués :*

```
t = [2, 5, 42, 1, 31, 0, 2, 42, 1]
n = len(t)
a = selection(t,2,0,n)
print t
```

Question 2.4 *Soit* `t` *une liste de* `n` *nombres. En utilisant une récurrence sur* `k`*, expliquer pourquoi un appel* `selection(t,k,0,n)` *se termine toujours (c'est-à-dire finit par renvoyer un résultat), pour tout* `k` ≥ 0.

Question 2.5 *Déterminer, en fonction de* `d = fin - debut`*, le nombre de comparaisons effectuées durant l'exécution de* `indiceMini(t,debut,fin)`.
En déduire, en fonction de la longueur `n` *de* `t` *et de* `k`*, le nombre de comparaisons effectuées durant l'exécution de* `selection(t,k,0,n)`.

Question 2.6 *Indiquer la complexité de l'algorithme de calcul du médian donné par*

```
def median(t) :
    n = len(t)
    return selection(t,n/2,0,n)
```

FIN.