

## Devoir d'informatique n°1. Corrigé.

### Exercice. Codage des matrices creuses par des dictionnaires

```
def somme(A,B) :  
    n = A["dim"] ; C = {"dim":n}  
    for x in A :  
        if (x != "dim") and (x in B) :  
            v = A[x] + B[x]  
            if v != 0 :  
                C[x] = A[x] + B[x]  
    # il reste à traiter les cas où  $A[i,j] = 0$  et  $B[i,j] \neq 0$   
    for x in B :  
        if not(x in A) : C[x] = B[x]
```

### Problème. Recherche du médian et problème de sélection

1) On tri le tableau en  $O(n \log n)$  opérations et on renvoie l'élément d'indice  $p$  du tableau trié.

2)

```
def triPartiel(p,L) :  
    n = len(L)  
    for k in range(p+1) :  
        j = k  
        for i in range(k+1,n) :  
            if L[i] < L[j] : j = i  
        L[k],L[j] = L[j],L[k]  
    return L[p]
```

3)

```
def moyenne(g,d,L) :  
    x = 0  
    for i in range(g,d) : x = x + L[i]  
    return x/(d-g)      ### ici, ne pas utiliser la division entière //
```

4) a)

```
01. def partition(v,g,d,L) :  
02.     i = g ; j = d-1  
03.     while i <= j  
04.         if L[i] <= v : i = i+1  
05.         elif : L[j] > v : j = j-1
```

```
06.           else :  L[i],L[j] = L[j],L[i] ; i = i+1
```

```
07.         return i
```

b) L'invariant est : A chaque étape, les éléments de  $L[g : d]$  d'indices  $< i$  ont une valeur  $\leq v$ .

Et de même les éléments d'indices  $> j$  ont une valeur  $> v$

La propriété est vraie initialement car il n'y a aucun élément de  $L[g : d]$  d'indice  $< g$  ou  $\geq d$ .

Lors de l'exécution de la ligne 04, la propriété est conservée car  $L[i] \leq v$ . De même pour la ligne 05. Sinon, on a au départ  $L[i] > v$  et  $L[j] \leq v$ , donc après l'échange, on a bien  $L[i] \leq v$ .

A chaque itération,  $j - i$  décroît de 1, donc l'algorithme aboutit en au plus  $(d - g)$  itérations.

A la sortie, on a  $j = i - 1$  ; les éléments d'indices  $< i$  sont donc exactement ceux de valeur  $\leq v$ .

5)

```
def selection(p,g,d,L) :
```

```
    if d-g == 1 : return L[d]    ### ici, on a nécessairement p = 0
```

```
    v = moyenne(g,d,L) ; k = partition(v,g,d,L)
```

```
    if g+p < k : return selection(p,g,k,L)
```

```
    else : return selection(p-(k-g),k,d,L)
```

*Remarque* : Si  $d - g \geq 2$ , comme les réels sont distincts, il y a au moins dans  $L[g : d]$  un réel  $< v$  et un réel  $> v$ , donc on a nécessairement  $g < k < d$ , donc les deux sous-tableaux  $L[g : k]$  et  $L[k : d]$  sont de longueur strictement inférieure à  $d - g$ . **Ce qui permet d'assurer la terminaison de l'algorithme.**

6) Si le pivot  $v$  est toujours la valeur médiane des sous-tableaux étudiés, le coût  $c(n)$  vérifie :

$$c(n) \leq Kn + \max \left( c \left( \left\lfloor \frac{n}{2} \right\rfloor \right), c \left( \left\lceil \frac{n}{2} \right\rceil \right) \right)$$

Avec  $n = 2^N$ , on obtient  $c(n) \leq Kn + c \left( \frac{n}{2} \right)$ . Donc  $c(n) \leq K \left( n + \frac{n}{2} + \frac{n}{4} + \dots + \frac{n}{2^{N-1}} \right) + c(1)$ .

Comme  $\sum_{k=0}^{+\infty} \frac{1}{2^k} = 2$ , on obtient bien  $c(n) = O(n)$ .

7)

```
def medians(g,d,L) :
```

```
    n = len(x) ; M = []
```

```
    i = g
```

```
    while i+4 < d :
```

```
        M.append(TriPartiel(2,L[i,i+5]))
```

```
        i = i+5
```

```
    return M
```

Variante :

```
def medians(g,d,L) :
```

```
    n = len(x) ; M = []
```

```

for i in range(g,d,5) :
    M.append(TriPartiel(2,L[i,i+5]))
return M

```

8)

```

def selection2(p,g,d,L) :
    if d-g < 5 : return TriPartiel(p,g,d,L)
    M = medians(g,d,L) ; q = len(M)
    v = selection2(q//2,0,q,M) ; k = partition(v,g,d,L)
    if g+p < k : return selection2(p,g,k,L)
    elif g+p > k : return selection2(p-(k-g),k,d,L)

```

9) a) Considérons par exemple le nombre  $n'$  de termes qui sont  $\leq v$ , où  $v$  est le médian de  $M$ .

Il y a au moins  $\left\lfloor \frac{q}{2} \right\rfloor$  indices  $k$  tels que  $y_k \geq v$ .

Pour chacun de ces  $y_k$ , il y a au moins 3 termes de  $\{x_{5k}, \dots, x_{5k+4}\}$  qui sont  $\geq y_k$ , donc a fortiori  $\geq v$ .

Donc  $n' \leq n - 3 \times \left\lfloor \frac{q}{2} \right\rfloor$ . On procède de même pour les éléments  $> v$ .

b) La recherche du médian demande  $O(n) + c(q)$  opérations, où  $q = \left\lfloor \frac{n}{5} \right\rfloor$ .

Donc  $\forall n \geq 5$ ,  $c(n) \leq K n + c(q) + c(n')$ , et par a),  $c(n') \leq c\left(n - 3 \times \left\lfloor \frac{q}{2} \right\rfloor\right)$ .

c) On a  $q \geq \frac{n}{5} - 1$ , d'où  $\left\lfloor \frac{q}{2} \right\rfloor \geq \frac{q}{2} - 1 \geq \frac{n}{10} - \frac{3}{2}$ . Donc  $n' \leq \frac{7n}{10} + \frac{9}{2}$ .

Supposons que  $c(r) \leq E r$  pour tout  $1 \leq r < n$ .

Pour  $c(n) \leq K n + E \left( \frac{n}{5} + \frac{7n}{10} + \frac{9}{2} \right) \leq \left( K + \frac{9}{10} E \right) \times n + \frac{9}{2} E$ .

On peut choisir  $E \geq E_0$  assez grand de sorte que  $\left( K + \frac{9}{10} E \right) < E$ .

Ainsi, pour  $n \geq n_0 \geq 5$  assez grand, on a alors  $c(n) \leq E n$ .

On choisit  $E$  assez grand pour avoir en plus  $c(n) \leq E n$  pour  $n \leq n_0$ , en prenant  $E = \max \left( E_0, \sup_{1 \leq n \leq n_0} \frac{c(n)}{n} \right)$ .

Ayant choisi  $E$  ainsi, on peut alors conclure  $M(n) \leq E n$  par récurrence forte sur  $n$ .