

# CT - 1 - Algorithmique Élémentaire du I21

## Licence Informatique

17 mai 2021

```
1 SMULT( a : nombre, b: double, c : chiffre )
2 variable   i : indice;
3           ret : double
4 debut
5   ret ← 0
6   i ← 0
7   tant que i < taille( a )
8     ret ← c * a[ i ] + ret
9     a[i] ← ret div b
10    ret ← ret mod b
11    inc( i )
12  ftq
13    retourner ret > 0
14 fin
```

**Question 1** On considère l'algorithme `smult( a, b, c )` pour calculer sur place le produit du nombre `a` par le chiffre `c` dans la base `b`. Il doit renvoyer `FAUX` en cas de dépassement de capacité.

Corriger le code. Combien de lignes étaient erronées ?

-

En base octale, combien d'instances de taille 4 sont calculées sans débordement lors d'une multiplication par le chiffre 4 ?

-

```
1 def star( x, n ) :
2     if n == 0 :
3         return ''
4     res = star(x + x, n // 2 )
5     if n % 2 == 1 :
6         res = x + res
7     return res
```

**Question 2** Quelle est la nature de l'algorithme `star(x,n)` ?

- *récurif* *itératif*

$n$	$10^3$	$10^6$	$10^9$
$T(n)$	0	0.1 secondes	1 minute

**Question 3** Le tableau ci-dessus renseigne sur les temps de calcul d'une implantation de TeeSort, un algorithme de tri comparatif inventé par le professeur Tee. L'implantation est-elle :

-      quasilinéaire      quadratique      incorrecte

```

1 POSITION( x : nombre , t : tableau de n nombres )
2 variable i, j, q : indice
3 debut
4   i ← 1
5   j ← n
6   tant que ( i < j )
7     q ← ( i + j ) div 2
8     si ( x = t[q] ) alors
9       tantque q > 0 et t[q] = x
10        q = q - 1
11     ftq
12
13     retourner q + 1
14   fsi
15   si ( t[ q ] > x ) alors
16     j ← q - 1
17   sinon
18     i ← q + 1
19   fsi
20   ftq
21   retourner 0
22 fin

```

L'algorithme de recherche position( v, t) s'applique à un tableau d'entiers trié en ordre croissant.

**Question 4** Que peut-on affirmer en ligne 12 ?

-  
-

Préciser le rôle de l'algorithme.

-  
-

Donner une instance favorable pour  $n = 8$ .

-

Préciser la complexité du pire des cas.

-      logarithmique      linéaire      quasilinéaire

```

1 MAJOR( t : tableau de n objets )
2 variable i , j, k : indice
3 debut
4 i ← 1
5 tant que ( i < n div 2 )
6   j ← i
7   k ← 0
8   tant que j ≤ n
9     si t[i] = t[j] alors inc(k) fsi
10    j = j + 1
11  ftq
12  si k > n div 2 alors
13    retourner VRAI
14  fsi
15  i ← i + 1
16  ftq
17  retourner FAUX
18 fin

```

**Question 5** Une valeur  $v$  d'un tableau  $t$  de taille  $n$  est dite majoritaire si et seulement si le nombre d'occurrences de  $v$  dans  $t$  est strictement supérieur à  $n/2$ . L'algorithme major(t) possède une imprécision en ligne 5, expliquer.

-  
-

Quel est le nombre de passages par la ligne 9 pour une instance de taille  $n$  sans élément majoritaire ?

-

-

-

**Question 6** Soient  $0 < m \leq n$  deux entiers. Étant donné un tableau de  $n$  nombres tous distincts. On souhaite déterminer les  $m$  plus petites valeurs. On considère deux approches. La première approche (naïve) consiste à déterminer la plus petite valeur, puis la seconde plus petite valeur, la troisième etc... Donner le nombre de comparaisons réalisées par cette approche en fonction de  $m$  et de  $n$ .

-

-

La seconde approche consiste à trier le tableau au moyen d'un tri fusion et de retenir les  $m$  premières valeurs. Préciser le nombre de comparaison.

-

-

Quelle approche vous semble le plus adéquate pour traiter une instance de taille  $2^{32}$  et  $m = 8$  ?

-      *approche naïve*      *approche par tri*

Quelle approche vous semble le plus adéquate pour traiter une instance de taille  $2^{32}$  et  $m = 64$  ?

-      *approche naïve*      *approche par tri*

**Question 7** On considère la décomposition d'un entier  $n$  en produit de facteurs premiers :

$$n = p_1^{r_1} \cdot p_2^{r_2} \cdots p_k^{r_k}, \quad p_i \text{ premiers et distincts.}$$

La somme des exposants  $r_i$  est appelée valuation de  $n$ , elle est notée  $\text{val}(n)$ . Par exemple, la valuation de  $24 = 3 \cdot 2^3$  et  $\text{val}(24) = 3 + 1 = 4$ .

Compléter la table :

$n$	2	3	4	5	6	7	8
$\text{val}(n)$							

Que peut-on dire des nombres de valuation 1 ?

-

L'affirmation  $\text{val}(n) = \Omega(\log(n))$ , est-elle correcte :

-      *oui*      *non*

Compléter la fonction `valuation( n )` pour calculer la valuation de l'entier  $n > 0$ .

```

1 def valuation( n ) :
2     r = 0
3     d = 2
4     while n > 1 :
5         if n % d == 0 :
6             while           :
7
8
9
10            d = d + 1
11    return r

```

Préciser le temps de calcul.

-

**Question 8** Un tableau d'entiers  $t$  est rangé en ordre alternatif s'il existe un indice  $j$  tel que :

$$\forall i < j, \quad t[i] \neq t[i + 1] \quad \text{et} \quad \forall i \geq j, \quad t[i] = t[j]$$

Autrement dit, il existe un seuil  $j$  en dessous duquel les éléments consécutifs sont distincts, et au dessus duquel ils sont tous égaux. Par exemple, le tableau ci-dessous est en ordre alterné avec le seuil  $j = 5$ .

$$5 \quad 3 \quad 1 \quad 4 \quad 1 \quad 5 \quad 5 \quad 5 \quad 5$$
 $j$

```

1 ALTERNATIF( t: tableau d'entiers )
2 variable i, j, k, n : indice
3 debut
4   i ← 0
5   j ← 1
6   k ← 0
7   n ← taille( t )
8   tant que ( j < n )
9     si t[j] != t[i] alors
10      inc(i)
11      t[i] = t[j]
12      si k > 0 alors
13       inc(i)
14      t [i] ← t[i-2]
15       dec(k)
16     fsi
17   sinon
18     inc(k)
19   fsi
20   inc(j)
21
22
23
24
25 fin

```

Il s'agit de compléter l'algorithme `alternatif( t : tableau )` pour réaliser un rangement alternatif. Pour comprendre l'algorithme proposé, partons d'un sac de  $n$  balles que l'on range de gauche à droite sur une étagère au moyen d'une corbeille en procédant de la manière suivante. Prendre une balle dans le sac, la placer sur l'étagère. Tant que le sac n'est pas vide, prendre une balle dans le sac.

- Si la balle n'est pas de la même couleur que la dernière balle sur l'étagère, la ranger à côté, et si de plus la corbeille n'est pas vide, prendre une balle dans la corbeille et la ranger à côté sur l'étagère.
- Si la balle est de la même couleur que la dernière balle sur l'étagère, la mettre dans la corbeille.

A la fin de cette procédure, deux balles côte à côte sur l'étagère sont de couleurs différentes. Toutes les balles dans la corbeille sont de la même couleur.

Compléter l'algorithme. Préciser le temps de calcul.

Comment utiliser cet algorithme pour résoudre le problème de l'élément majoritaire en temps linéaire.