

TD1

Exercice 1.

Pour chacun des algorithmes suivants, donnez la valeur de c en fonction de n à la fin de l'algorithme puis la complexité de l'algorithme.

<pre> Algo 1 : c ← 0 Pour i de 1 à n : Pour j de 1 à n : c ← c + 1 Algo 2 : c ← 0 Pour i de 1 à n : Pour j de 1 à i : c ← c + 1 Algo 3 : c ← 0 i ← 1 Tant que i < n : c ← c+1 i ← i*2 Algo 4 : c ← 0 i ← 2 Tant que i < n : c ← c+1 i ← i*i </pre>	<pre> Algo 5 : c ← 0 i ← 1 Tant que i < n : Pour j de 1 à i : c ← c+1 i ← i*2 Algo 6 : c ← 0 i ← 0 Tant que i < n : c ← c+1 i ← i+c Algo 7 : i ← n c ← 0 Tant que i > 1 : Pour j de 1 à i : Pour k de 1 à c : Afficher("ping") i ← ⌊i/2⌋ c ← c+1 </pre>
---	--

Exercice 2.

On considère des algorithmes de complexité : $\log_2(n)$, \sqrt{n} , n , $50 * n$, $n \log_2(n)$, n^2 , n^3 , 2^n .

1. Calculez chacune de ces fonctions pour les puissances de 10 de 10^1 à 10^{10} . Le résultat sera donné sous la forme approchée $x.10^k$ (Pour 2^n on ne calculera que jusqu'à $n = 10^3$).
2. En supposant que l'on dispose d'une machine capable de faire 10^6 (1 million) d'opérations par seconde, quelle est la taille des problèmes que l'on peut résoudre en une seconde, 1000 secondes (environ 17 minutes), 10 000 secondes (environ 2h47) ?

Exercice 3.

Un nombre premier est un nombre qui n'a pas de diviseurs autre que lui-même et 1. Écrivez un algorithme testant la primalité d'un nombre, la complexité doit être inférieure à $O(n)$ (sous-linéaire).

Exercice 4.

Écrivez un algorithme qui prend en paramètre un tableau d'entiers de taille n triés (les

valeurs sont dans l'ordre croissant) ainsi qu'un entier a et détermine si a est dans T . Quelle est la complexité de votre algorithme (on peut faire beaucoup mieux que linéaire) ?

Exercice 5.

Une matrice carrée de taille n est un double tableau de taille $n \times n$. Soient A et B deux matrices de taille n dont on écrit les entrées $a_{i,j} = A[i][j]$ et $b_{i,j} = B[i][j]$ avec $0 \leq i, j < n$. Le produit $A \times B$ est donné par la matrice C dont les entrées sont calculées par la formule

$$c_{i,j} = \sum_{k=0}^{n-1} a_{i,k} b_{k,j}.$$

1. Écrivez l'algorithme qui calcule la matrice C en fonction de A et B (on supposera qu'on possède une fonction `CreeMatrice(n)` qui crée une matrice carrée de taille n et l'initialise à 0).
2. Quelle est la complexité de votre algorithme ? La fonction `CreeMatrice` peut-elle l'influencer ?

Exercice 6.

Voici un algorithme qui calcule la valeur maximum d'un tableau \mathbf{t} de taille n .

```

m ← t[0]
Pour i de 1 à n-1 :
    Si t[i] > m :
        m ← t[i]
Retourner m
    
```

Supposons que l'on souhaite implémenter cet algorithme dans un langage dont l'opération élémentaire qui consiste à écrire en mémoire est extrêmement coûteuse. Nous cherchons donc à estimer la complexité de cet algorithme en fonction du nombre d'écritures en mémoire. Supposons que l'on exécute ce programme sur un tableau \mathbf{t} de taille n , trié aléatoirement.

1. Quelle est la complexité dans le meilleur des cas ? Dans le pire des cas ? Avec quelle probabilité est-on dans chacun de ces deux cas ?
2. On cherche maintenant à évaluer la complexité en moyenne, que l'on note $c(n)$. Quelles sont les valeurs de $c(1), c(2), c(3)$?
3. Supposons que l'élément maximum de \mathbf{t} est en i -ème position. Quelle est la complexité en moyenne sur un tel tableau, en fonction de $c(i-1)$?
4. Montrer que $c(n) = 1 + \frac{1}{n} \sum_{i=1}^{n-1} c(i)$.
5. En déduire que $c(n) = c(n-1) + \frac{1}{n}$.
6. En déduire que $c(n) \sim \ln n$.