

# Recherche par dichotomie et complexité - Cours

- Mars 2023

## 1 Complexité

### Définition : Ordre de complexité

On dit qu'un algorithme est d'une **complexité de l'ordre de  $f(n)$**  si il existe une constante positive  $K$  telle que, quelle que soit la taille  $n$  de l'entrée, le nombre d'opérations élémentaires est plus petit que  $K \times f(n)$ .

On dit alors que l'algorithme est en  $\mathcal{O}(f(n))$ .

## 2 Etude de la complexité

### 1. Recherche du maximum d'une liste

```
def maximum(liste):
    candidat = liste[0]
    for element in liste:
        if element > candidat:
            candidat = element
    return candidat
```

Calcul de la complexité

### 2. Moyenne

```
def maximum(liste):
    candidat = liste[0]
    for element in liste:
        if element > candidat:
            candidat = element
    return candidat
```

Calcul de la complexité

### 3. Table de multiplication

```
def table(liste):
    resultats = []
    for element1 in liste:
        for element2 in liste:
            resultats.append(element1 * element2)
    return resultats
```

Calcul de la complexité

### 4. Recherche par dichotomie

```
def dichotomie(liste, x):
    g = 0
    d = len(L) - 1
    while g <= d:
        m = (g + d) // 2
        if L[m] < x:
            g = m
        elif L[m] > x:
            d = m
        else:
            return m
    return None
```

Calcul de la complexité

### 3 Les complexités courantes

Dans la pratique, on ne choisira que des fonctions  $f(n)$  simples prise dans la liste suivante (classé par ordre croissant de rapidité) :

- $O(1)$  : complexité constante. Le temps d'exécution est indépendant de  $n$  (la taille de l'entrée).

Exemples :

- $O(\log(n))$  : complexité logarithmique. Le temps d'exécution augmente d'une quantité constante quand la taille de l'entrée ( $n$ ) est doublée.

Exemple :

- $O(n \log(n))$  : complexité linéaire. Le temps d'exécution est proportionnel à la taille de l'entrée ( $n$ )

Exemples :

- $O(n^2)$  : complexité quadratique. Le temps d'exécution est multiplié par 4 quand la taille de l'entrée est multipliée par 2.

Exemples :

- $O(n^k)$  : complexité polynomiale. Le temps d'exécution est majoré par un polynôme de la taille d'entrée.

Exemples :

- $O(k^n)$  : complexité exponentielle. Le temps d'exécution croit trop rapidement pour que la taille d'entrée puisse être grand.

Exemples :

#### Temps d'exécution en fonction de la taille d'entrée

Temps de calcul sur un ordinateur personnel actuel (10 milliards d'opérations par seconde)

nombre d'opérations / taille de l'instance	$n$	$n^2$	$n^3$	$2^n$	$2^{n^2}$
$n = 10$	1 ns (nanoseconde)	10 ns	100 ns	102 ns	4000 milliards d'années
$n = 50$	5 ns	250 ns	12,5 $\mu$ s (microseconde)	1 j 7 h	🤔
$n = 100$	10 ns	1 $\mu$ s	0,1 ms (milliseconde)	4000 milliards d'années	🤔
$n = 10.000$	1 $\mu$ s	10 ms	1 min 40 s	🤔	🤔
$n = 10^6$ (un million)	0,1 ms	1 min 40 s	3 ans 2 mois	🤔	🤔
$n = 10^9$ (un milliard)	0,1 s	3 ans 2 mois (aïe)	3 milliards d'années	🤔	🤔

« instantané »

« oups » (l'univers a 14 milliards d'années)