

Complexité

Fiche de synthèse

① TROIS QUESTIONS

Devant un algorithme, on se demande s'il s'arrête, c'est la **terminaison** ; s'il donne le résultat attendu, c'est la **correction** ; et en combien de temps, c'est la **complexité**.

Définition 1 (Complexité en temps) La **complexité en temps** d'un algorithme mesure le nombre d'**opérations élémentaires** qu'il effectue, en fonction de la **taille** n des données. Une opération élémentaire, comparaison, affectation, opération arithmétique, accès à un élément, compte pour une unité. Cette mesure ne dépend pas de la vitesse de l'ordinateur.

Pour une même taille, le nombre d'opérations dépend des données : on distingue le **meilleur cas**, le **cas moyen** et le **pire cas**. On étudie surtout le pire cas, qui garantit que l'algorithme ne sera jamais plus lent. Chercher un élément dans une liste de n éléments demande une comparaison s'il est en tête, n s'il est à la fin ou absent.

② LA NOTATION O

On ne garde qu'un **ordre de grandeur** : le terme qui croît le plus vite, sans son coefficient.

- $4n + 3$ opérations : $O(n)$.
- $2n^2 + 7n + 4$ opérations : $O(n^2)$.
- $n/3$ opérations : $O(n)$.

Classe	Nom	Exemple
$O(1)$	constante	accès à $L[i]$
$O(\log n)$	logarithmique	recherche dichotomique
$O(n)$	linéaire	parcours d'une liste
$O(n \log n)$	quasi linéaire	tri fusion, <code>sort</code>
$O(n^2)$	quadratique	tris par sélection et par insertion
$O(2^n)$	exponentielle	essayer tous les sous-ensembles
$O(n!)$	factorielle	essayer tous les ordres

$\log n$ est le logarithme de base 2 : le nombre de fois qu'on peut diviser n par 2 avant d'atteindre 1. $\log 1024 = 10$, et $\log$ d'un million vaut environ 20.

③ CONSÉQUENCES PRATIQUES

À un milliard d'opérations par seconde, pour un million de données, un algorithme en $O(n)$ met 1 ms, en $O(n \log n)$ 20 ms, en $O(n^2)$ environ 17 minutes. En $O(2^n)$, cent données suffisent à dépasser l'âge de l'Univers.

Quand la taille des données est multipliée par 10, le temps est multiplié par 10 pour une complexité linéaire, par 100 pour une complexité quadratique ; il augmente seulement d'une constante pour une complexité logarithmique.

④ COMPTER LES OPÉRATIONS

```
def recherche(L, x):
    for i in range(len(L)):
        if L[i] == x:
            return True
    return False
```

Dans le pire cas, x est absent : la boucle fait n tours, avec un accès et une comparaison à chaque tour, soit $2n$ opérations. La complexité est en $O(n)$.

```
def rencontres(eq):
    matchs = []
    for i in range(len(eq)):
        for j in range(i):
            matchs.append((eq[j], eq[i]))
    return matchs
```

Pour chaque i , la boucle intérieure fait i tours : en tout,

$$0 + 1 + \dots + (n-1) = \frac{n(n-1)}{2}.$$

Deux boucles imbriquées donnent ici une complexité en $O(n^2)$.

🔍 **Déterminer une complexité.** Compter le nombre de tours de chaque boucle dans le pire cas, multiplier pour des boucles imbriquées, puis ne garder que le terme dominant, sans coefficient.

📌 À retenir.

- La complexité compte les opérations élémentaires en fonction de la taille n , dans le pire cas.
- On ne garde que le terme dominant : $2n^2 + 7n + 4$ est en $O(n^2)$.
- Une boucle sur n éléments donne $O(n)$, deux boucles imbriquées souvent $O(n^2)$.