Chapitre 1 : Complexité

Un alogrithme effectue un traitement sur des données. Il doit être correct mais aussi efficace. La théorie de la complexité explore comment le temps de calcul dépend de la taille des données. Elle aide au choix de l'algorithme le plus efficace pour résoudre un problème.

1 Mesure des performances avec `time`

La bibliothèque Python `time` fournit un chronomètre qui permet de mesurer le temps de calcul. Il suffit d'inclure dans le programme la ligne

```
from time import time
```

lancer le chrono au début du programme et l'arrêter à la fin. On fait ensuite la différence entre les temps de départ et d'arrivée.

Par exemple pour chronométrer la création d'un tableau :

```
n = 10000                 # la taille du tableau voulu
tic = time()            # top départ

t = []
i = 0
while i < n:
    t.append(i)
    i+=1

tac = time()            # arrêt du chrono

#print(tac-tic)                    # en secondes
print(round(1000*(tac-tic),2))     # on convertit en ms et on arrondit au centième
```

Le temps obtenu varie à chaque exécution car d'autres processus peuvent mobiliser le processeur dans l'intervalle. Il faut plusieurs essais pour se faire un idée du temps de calcul réel.

2 Nombre d'opérations et taille des données

Pour estimer théoriquement le temps de calcul, on compte le nombre d'opérations simples (comparaisons, additions, affectations...) en fonction de n (la taille des données).

Parcourir les données avec une boucle à nombre d'opérations constant fera Cn opérations, en négligeant celles hors de la boucle. En effet, on veut prévoir l'évolution des performances quand n augmente.

3 Complexités linéaire et quadratique

L'estimation du nombre d'opérations en fonction de la taille des données n pour n grand s'appelle *performance asymptotique* ou **complexité** de l'algorithme.

Pour un programme à Cn opérations, le temps de calcul est proportionnel à n, autrement dit dépend linéairement de n.

- On dit que la complexité est *linéaire* ou en $O(n)$, prononcé *grand Ô de n*.

Pour un programme à Cn^2 opérations (c'est souvent le cas des boucles imbriquées), le temps de calcul est proportionnel à n^2.

- On dit que la complexité est *quadratique* ou en $O(n^2)$.

4 Complexité dans le pire des cas

Un jeu de données (ou *instance*) favorable peut réduire le temps de calcul, mais c'est un "heureux hasard". Quand la donnée peut être quelconque, il faut prévoir la complexité *dans le pire des cas*.