Complexité

235711131723

14 octobre 2022

1 Avertissement

Ceci n'est qu'un résumé foireux rédigé par un étudiant sans aucune expérience dans le domaine.

À prendre avec des pincettes.

2 Comparaison de fonction

2.1 Relation de domination

Soit f est g deux applications de $\mathbb{N}$ dans $\mathbb{N}$. Alors on dit que f est dominée par g (noté $f = O(g)$ dans la notation de Landau) si et seulement si $\exists c \in \mathbb{R}_+^*, \exists N \in \mathbb{N}$ tel que $\forall n \geq N, f(n) \leq cg(n)$.

— On note $f = \Omega(g)$ si et seulement si $g = O(f)$.
— On note $f = \Theta(g)$ si et seulement si $f = O(g)$ et $f = \Omega(g)$.

Exemple. Posons $\begin{array}{rcl} f : \mathbb{N} & \to & \mathbb{N} \\ n & \to & 5n+6 \end{array}$ et $\begin{array}{rcl} g : \mathbb{N} & \to & \mathbb{N} \\ n & \to & n+3 \end{array}$. Montrons que $f = \Theta(g)$.

$\Rightarrow f = O(g)$. Posons $c = 5$. Étudions le signe de $f(n) - cg(n)$ pour tout entier naturel n. Alors :

$$\forall n \in \mathbb{N}, f(n) - cg(n) = 5n + 6 - (5n + 15) = -9 \leq 0$$

On prend alors $N = 0$. Alors on a bien $\forall n \geq N, f(n) - cg(n) \leq 0$. Donc $f = O(g)$.

$\Rightarrow f = \Omega(g)$. (autrement dit, $g = O(f)$). Posons $c = 1$. Étudions le signe de $g(n) - cg(n)$ pour tout entier naturel n. Alors :

$$\forall n \in \mathbb{N}, g(n) - cf(n) = n + 3 - (5n + 6) = -4n - 3 \leq 0$$

On prend alors $N = 0$. Alors on a bien $\forall n \geq N, g(n) - cf(n) \leq 0$. Donc $f = \omega(g)$.

Conclusion. On a bien $f = \Theta(g)$.

3 Complexité

3.1 Complexité temporelle

En algorithmique, la complexité en temps est une mesure du temps utilisé par un algorithme, exprimé comme fonction de la taille de l'entrée. Le temps compte le nombre d'étapes de calcul avant d'arriver à un résultat.

Usuellement le temps correspondant à des entrées de taille n est le temps le plus long parmi les temps d'exécution des entrées de cette taille ; on parle de complexité dans le pire cas. Les études de complexité portent dans la majorité des cas sur le comportement asymptotique, lorsque la taille des entrées tend vers l'infini, et l'on utilise couramment les notations grand O de Landau.

La complexité en temps étant la mesure la plus courante en algorithmique, on parle parfois simplement de la complexité d'un algorithme, mais il existe d'autres mesures comme la complexité en espace. [1]

4 Différents algorithmes de tri

4.1 Tri à bulle

Principe.

- On parcourt le tableau et on permute une case avec la suivante si elles ne sont pas dans le bon ordre
- À la fin du parcours, si on fait une permutation, on recommence le parcours.

Théorème. Le tri à bulle termine correctement en $\Theta(n^2)$.

Algorithme. Entrée : Tableau d'entiers T de taille n.

```
1  Bool Permutation = Vrai
2  Tant Que(Permutation):
3      Permutation = Faux
4      Pour(i = 0; i < n - 1; i = i + 1):
5          Si(T[i] > T[i + 1]):
6              temp = T[i]
7              T[i] = T[i + 1]
8              T[i + 1] = temp
9              Permutation = Vrai
10         Fin Si
11     Fin Pour
12 Fin Tant Que
```

1. Source : Wikipedia

Démonstration du théorème. Posons l'invariant ($j \in \mathbb{N}$) $H(i)$: "Après i exécution de la boucle **Tant Que**, les j dernières cases du tableau contiennent les plus grands éléments triés par ordre croissant."

— **Initialisation de l'invariant.** Quand on arrive à la première exécution de la ligne 3, on a $j = 0$ et l'invariant est totalement vrai.