Complexité algorithmique

1 Définitions

La compléxité est l'une des performances d'un programme. Elle quantifie les ressources nécessaires pour mener l'exécution du programme. Elle est exprimée en fonction de n la taille des données traitées.

1. Compléxité en temps :

2. Compléxité en mémoire :

2 Détermination du nombre d'opérations élémentaires

Pour déterminer le coût total en opérations C_t d'un programme, nous utiliserons en général le modèle suivant :

1. Une opération élémentaire (une affectation, une comparaison, une incrémentation de boucle, et.) est considéré comme l'unité de base du coût d'un algorithme, noté C_e.
2. Le coût C_A de la séquence d'instructions A est la somme des coûts des instructions.
3. Le coût C_{si} d'un test "si *condition* alors A sinon B" est de $C_{si} = max(C_A, C_B) + 1$
4. Le coût C_{boucle} d'une boucle exécutée k fois est la multiplication du coût C_A du corps de boucle par k : $C_{boucle} = k \cdot C_A$.

Ce modèle très simplifié permet comparer des algorithmes entre eux.

3 Catégorisations : Notation O

3.1 Définition mathématique

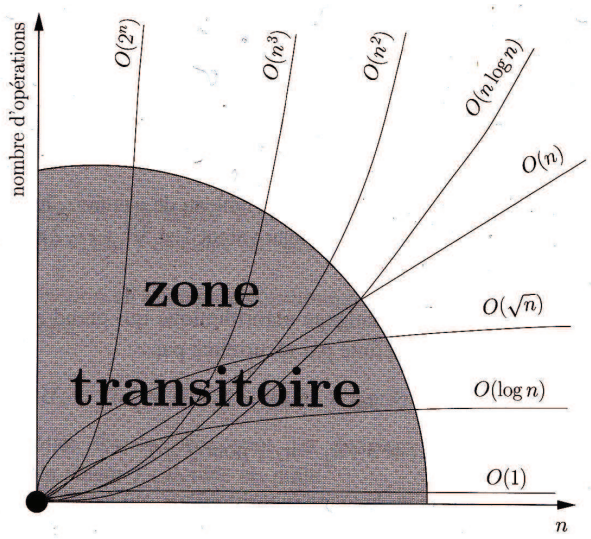
Soit $f(n)$ et $g(n)$ deux expressions dépendant de n.
On dit que f est dominée par g et on note $f(n) = \mathrm{O}(g(n))$ lorsque :

$$\exists N_0 \in \mathbb{N} \ / \ n \geq N_0 \Rightarrow \frac{f(n)}{g(n)} \text{ est borné}$$

En d'autres termes on dit que f est en $\mathrm{O}(g)$ si, à partir d'un certain seuil N_0, f ne croît pas plus vite que g. Ou que g croit plus vite que f.

3.2 Aspect pratique : les catégories

On dit qu'un algorithme nécessitant $C_t(n)$ opérations élémentaires est en $O(g(n))$ lorsque $C_t(n)$ est dominée par $g(n)$. En pratique, g sera une expression simple en n. Nous retiendrons 8 catégories, ou 8 expressions pour $g(n)$. Ces expressions de n que nous utiliserons sont résumées dans la figure suivante :



On s'interesse aux cas où le nombre de données est grand (n au moins 1000). On choisira la catégorie de l'algorithme en choisissant l'expression en O qui domine $C_t(n)$ ayant le plus faible accroissement.

Nous éviterons ainsi d'écrire si possible des dominations trop grossières de C_t. Ainsi si un algorithme en $O(n^2)$ est également en $O(n^3)$, mais nous écrirons qu'il est en $O(n^2)$ et non en $O(n^3)$.

Ordre de grandeur	Nom courant	Temps pour n=50	Temps pour n=10^6	Exemple
O(1)	Temps constant	10 ns	10 ns	
O($\log n$)	Logarithmique	20 ns	60 ns	Dichotomie
O(n)	Linéaire	500 ns	10 ms	Parcours d'une liste
O($n \log n$)	Linéarithmique	1 μs	60 ms	Tri par fusion
O(n^2)	Quadratique	25 μs	3h	Parcours Tableaux 2d
O(n^3)	Cubique	1.25 ms	300 ans	Multiplication Matrices naïve
O(2^n)	Exponentielle	135 jours	¿ 10^{300000} années	Problème du Sac à dos
O($n!$)	Factorielle	10^{48} ans	...	Voyageur de Commerce naïf

4 Différentes nuances de complexité

4.1 Complexité en temps dans le pire des cas

4.2 Complexité en temps dans le meilleur des cas

4.3 Complexité en mémoire