TD Info n°12 : Complexité d'un algorithme

ECE3 Lycée Carnot

9 février 2012

Complexité

Depuis le début de l'année, nous apprenons péniblement à écrire des algorithmes en Pascal servant à calculer des choses plus ou moins compliquées. Nous nous sommes jusqu'ici assez peu préocuppés d'optimiser nos algorithmes pour les rendre les plus efficaces possibles. C'est pourtant un souci prédominant de l'informatique actuelle. On peut en gros s'intéresser à deux choses quand on veut rendre un programme le plus performant possible :

- la quantité de mémoire utilisée par l'algorithme (ainsi, il sera toujours préférable d'utiliser le moins de variables possible dans un programme, surtout quand ce sont des variables gourmandes en mémoire comme des tableaux).
- le temps d'exécution de l'algorithme : on a intérêt à minimiser le nombre d'opérations effectuées au sein de l'agorithme pour qu'il tourne plus rapidement.

On parle de complexité en espace ou en temps pour mesurer ces deux paramètres. On ne s'intéressera pour l'instant qu'à la complexité en temps, c'est-à-dire qu'on cherchera à comparer le temps mis par différents algorithmes pour résoudre un même problème. Pour cela, on essaiera tout simplement de compter le nombre d'opérations effectuées par l'agorithme et de le comparer à la taille des données manipulées par l'algorithme. Comme nous travaillons pour l'instant avec des tableaux, la taille des données sera tout simplement le nombre d'éléments que nous utiliserons dans nos tableaux. Ainsi, si notre tableau représente un polynome, le nombre d'éléments utilisés est égal au degré du polynome plus un. On classera les algorithmes en différentes catégories selon le nombre d'opérations effectuées. On dira notamment qu'un algorithme est :

- linéaire si le nombre d'opérations est proportionnel à la taille des données (ainsi, avec des données dix fois plus volumineuses, l'algorithme mettra dix fois plus de temps).
- quadratique si le nombre d'opérations est proportionnel au carré de la taille des données (ainsi, avec des données dix fois plus volumineuses, l'algorithme mettra cent fois plus de temps).
- polynomial si le nombre d'opérations est proportionnel à une certaine puissance de la taille des données.
- exponentiel le nombre d'opérations est proportionnel à un certain nombre élevé à une puissance égale taille des données (ainsi, si le nombre d'opérations est proportiel à 2^n où n représente la taille des données, l'agorithme mettra deux fois plus de temps pour tourner à chaque fois qu'on rajoute un élément dans notre tableau, ce qui est absolument affreux).

Ainsi, si un algoritme met 1 seconde pour trier les données d'un tableau contenant 10 éléments, et qu'on veut lui faire trier un tableau à 100 éléments, il mettra :

- 10 secondes s'il est linéaire
- 100 secondes s'il est quadratique
- 1 000 secondes (soit un peu plus d'un quart d'heure) s'il est cubique
- 2^{90} secondes (soit un peu plus de 39 milliards de millards d'années) s'il est exponentiel de base 2
- 1.1^{90} secondes (environ une heure et demie) s'il est exponentiel de base 1.1

Petits exercices

Exercice 1

1. Estimer le nombre d'opérations (additions ou multiplications) effectuées par un algorithme calculant tous les coefficients binomiaux de la n-ème ligne du triangle de Pascal :
   - en utilisant la formule $\frac{n!}{k!(n-k)!}$ pour chaque valeur de k.
   - en utilisant cette même formule, mais en stockant les valeurs des factorielles de façon à ne calculer qu'un seule fois chaque factorielle différente.
   - en utilisant cette même formule mais en calculant les factorielles les unes après les autres en les stockant à chaque étape.
   - en utilisant la formule de Pascal (algorithme vu au TD10)
2. En considérant qu'une addition est en moyenne 10 fois plus rapide qu'une multiplication, comparer le temps d'exécution de chaque algorithme pour remplir la 50 ème ligne du triangle.

Exercice 2

Déterminer le nombre de comparaisons nécessaire pour trier dans l'ordre un tableau contenant 16 éléments quand on utilise chacune des méthodes suivantes :

1. on cherche le plus petit élément du tableau, on le supprime du tableau, puis on recommence jusqu'à ce qu'il n'y ait plus d'éléments dans le tableau.
2. on compare les deux premiers éléments du tableau (on les échange si besoin), puis les deux suivants et ainsi de suite jusqu'au deux derniers ; on recommence 15 fois cette manoeuvre (cette façon de trier porte la curieuse dénomination de tri à bulles).
3. on découpe de tableau en 8 paquets de 2 ; on trie chaque paquet, puis on regroupe les paquets de 2 en paquets de 4, puis de 8, et on finit par regrouper les deux paquets de 8 (je vous laisse déterminer combien de comparaisons il faut faire à chaque étape pour regrouper dans le bon ordre).