Complexité des algorithmes

Dans ce cours, il sera question de complexité d'algorithmes, c'est-à-dire du nombre d'opérations élémentaires (affectations, comparaisons, opérations arithmétiques) effectuées par l'algorithme. Elle s'exprime en fonction de la taille n des données. On dit que la complexité de l'algorithme est $O(f(n))$ où f est d'habitude une combinaison de polynômes, logarithmes ou exponentielles. Ceci reprend la notation mathématique classique, et signifie que le nombre d'opérations effectuées est borné par $cf(n)$, où c est une constante, lorsque n tend vers l'infini.

Considérer le comportement à l'infini de la complexité est justifié par le fait que les données des algorithmes sont de grande taille et qu'on se préoccupe surtout de la croissance de cette complexité en fonction de la taille des données. Une question systématique à se poser est: que devient le temps de calcul si on multiplie la taille des données par 2?

Les algorithmes usuels peuvent être classés en un certain nombre de grandes classes de complexité.

- Les algorithmes sub-linéaires, dont la complexité est en général en $O(\log n)$. C'est le cas de la recherche d'un élément dans un ensemble ordonné fini de cardinal n.
- Les algorithmes linéaires en complexité $O(n)$ ou en $O(n \log n)$ sont considérés comme rapides, comme l'évaluation de la valeur d'une expression composée de n symboles ou les algorithmes optimaux de tri.
- Plus lents sont les algorithmes de complexité située entre $O(n^2)$ et $O(n^3)$, c'est le cas de la multiplication des matrices et du parcours dans les graphes.
- Au delà, les algorithmes polynomiaux en $O(n^k)$ pour $k > 3$ sont considérés comme lents, sans parler des algorithmes exponentiels (dont la complexité est supérieure à tout polynôme en n) que l'on s'accorde à dire impraticables dès que la taille des données est supérieure à quelques dizaines d'unités.

La recherche de l'algorithme ayant la plus faible complexité, pour résoudre un problème donné, fait partie du travail régulier de l'informaticien. Il ne faut toutefois pas tomber dans certains excès, par exemple proposer un algorithme excessivement alambiqué, développant mille astuces et ayant une complexité en $O(n^{1,99})$, alors qu'il existe un algorithme simple et clair de complexité $O(n^2)$. Surtout, si le gain de l'exposant de n s'accompagne d'une perte importante dans la constante multiplicative: passer d'une complexité de l'ordre de $n^2/2$ à une complexité de $10^{10}\ n \log n$ n'est pas vraiment une amélioration. Les critères de clarté et de simplicité doivent être considérés comme aussi importants que celui de l'efficacité dans la conception des algorithmes.