CHAPTER 1

Rappels de complexité (et d'algèbre linéaire)

PROBLEM 1.0.1. Quel est le meilleur méthode pour calculer le déterminant d'une matrice ?
Comment prédire le temps que va mettre un programme pour s'exécuter ?
Quelle est le meilleur algorithme pour trouver un nom dans un annuaire ?
Comment savoir, entre deux algorithmes, lequel est le plus rapide ?
Comment savoir si un algorithme est optimal ?

EXEMPLE 1.0.2. Je recherche le nom «Zorro» dans un annuaire en utilisant l'algorithme suivant:

(1) Je pars du début de l'annuaire;
(2) Je compare le nom avec «Zorro»;
(3) Si oui, j'ai terminé;
(4) Si non, je recommence en 2 avec le nom suivant;

Analyse de ce que l'on a fait:
On s'est donné un problème (rechercher un mot dans un dictionnaire), un algorithme pour le résoudre (recherche exhaustive). Puis on a introduit un *modèle de calcul*:

(1) Choix de la mesure de *la taille d'une instance problème* (le nombre de mots d'un dictionnaire donné)
(2) Choix des *opérations élémentaires* (comparer deux mots)

Dans ce modèle, on a cherché le nombre d'opérations élémentaires effectuées par l'algorithme pour un problème de taille n. C'est ce que l'on appelle la *complexité de l'algorithme*. En fait, on a vu deux variations:

(1) Complexité au pire (n opérations)
(2) Complexité en moyenne ($\frac{n}{2}$ opérations)

À partir de cette information, et en connaissant le temps nécessaire pour de petites instances du problème on peut évaluer le temps nécessaire pour résoudre n'importe quelle instance du problème.

EXEMPLE 1.0.3. Évaluons la complexité de la recherche dichotomique.

Analyse:
La plupart du temps, il suffit d'avoir un ordre de grandeur du nombre d'opérations: les constantes sont sans grande importance. Un algorithme en $1000 \log_2 n + 50$ sera meilleur qu'un algorithme en $\frac{n}{1000}$ dès que l'on s'intéressera à des instances suffisament grandes.

DÉFINITION 1.0.4. Soient f et g deux fonctions de $\mathbb{N}$ dans $\mathbb{N}$ (par exemple les complexités de deux algorithmes).

On note $f = O(g)$ si, asymptotiquement, f est au plus du même ordre de grandeur que g, i.e. s'il existe une constante a et un entier N tels que $f(n) \leq ag(n)$ pour $n \geq N$.

On note $f = o(g)$ si, assymptotiquement, f est négligeable devant g, i.e. si pour toute constante a il existe N tel que $f(n) \leq ag(n)$ pour $n \geq N$.

En gros: f

REMARQUE 1.0.5. Quelques règles de calculs sur les $O()$:

(1) $O(4n+3) = O(n)$
(2) $O(\log n) + O(\log n) = O(\log n)$
(3) $O(n^2) + O(n) = O(n^2)$
(4) $O(n^3)O(n^2 \log n) = O(n^5 \log n)$

Et de même pour les $o()$.

EXERCICE 1. Donner un algorithme pour chercher le plus grand élément d'une liste de nombres. Évaluer la complexité de cet algorithme. Existe-t'il un meilleur algorithme ?

DÉFINITION 1.0.6. La *complexité d'un problème* est la complexité du meilleur algorithme pour le résoudre.

On dit qu'un algorithme est *optimal* si sa complexité coïncide avec celle du problème.

EXERCICE 2. Donner un algorithme pour calculer la somme de deux matrices carrées, et évaluer sa complexité. Est-il optimal ?

Donner un algorithme pour calculer le produit de deux matrices carrées, et évaluer sa complexité. Est-il optimal ?

Rappeller l'algorithme du pivot de Gauss, et donner l'ordre de grandeur de sa complexité.

EXERCICE 3. On dispose d'un ordinateur pouvant exécuter 10^9 opérations élémentaires par seconde (1GHz). On a un problème (par exemple, chercher un mot dans une liste, calculer le déterminant d'une matrice), et des instances de taille $1, 10, 100, 1000$ de ce problème. Enfin, on a plusieurs algorithmes pour résoudre ce problème, dont on connaît les complexités respectives: $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$, $O(n^3)$, $O(n^{10})$, $O(2^n)$, $O(n!)$, $O(n^n)$. Évaluer dans chacun des cas le temps nécessaire.

EXERCICE 4. Comparaison de la complexité de deux méthodes de tri.

On a une liste que l'on veut trier, mettons $7, 8, 4, 2, 5, 9, 3, 5$.

- Méthode 1 (tri sélection)
  (1) On échange le premier élément avec le plus petit des éléments $2, 8, 4, 7, 5, 9, 3, 5$

  (2) On échange le deuxième élément avec le plus petit des éléments restants $2, 3, 4, 7, 5, 9, 8, 5$
  (3) ...
  (4) Au bout de k étapes, les k premiers éléments sont triés; on échange alors le $k+1$ème élément avec le plus petit des éléments restants.
  (5) À la fin, la liste est triée: $2, 3, 4, 5, 5, 7, 8, 9$.
- Méthode 2 (tri fusion)
  (1) On groupe les éléments par paquets de deux, et on trie chacun de ces paquets: $7, 8, 2, 4, 5, 9, 3, 5$.
  (2) On groupe les éléments par paquets de quatres, et on trie chacun de ces paquets: $2, 4, 7, 8, 3, 5, 5, 9$.
  (3) ...
  (4) Au bout de k étapes, les paquets de 2^k éléments sont triés; on les regroupe par paquets de 2^{k+1} que l'on trie.
  (5) À la fin, tous les éléments sont dans le même paquet et sont triés: $2, 4, 7, 8, 3, 5, 5, 9$.

Quelle est la meilleure méthode ?

EXERCICE 5. Évaluer au mieux la complexité des problèmes suivants:

(1) Calcul du n-ième nombre de Fibonacci;
(2) Calcul du déterminant d'une matrice;
(3) Calcul du rang d'une matrice;
(4) Calcul de l'inverse d'une matrice;
(5) Calcul d'un vecteur x solution de $Ax = b$, où A est une matrice et b un vecteur;
(6) Calcul du pgcd de deux nombres;
(7) Test de primalité de n;
(8) Recherche du plus court chemin entre deux stations de métro à Paris;
(9) Calcul de la n-ième décimale de $\sqrt{2}$;
(10) Calcul de l'inverse d'un nombre modulo 3;
(11) Recherche d'un échec et mat en 4 coups à partir d'une position donnée aux échecs.
(12) Problème du sac à dos: étant donné un ensemble d'objets d'épaisseur variable, et un sac à dos de hauteur donnée, peut-on remplir le sac à dos à bloc, sans déborder ?