Algorithmique : les grandes lignes

l'Algorithmique, une science très ancienne.

- Antiquité. Euclide (calcul du pgcd), Archimède (approximation de π)
- le mot Algorithme vient du mathématicien arabe du 9ème siècle Al Khou Warismi
- L'algorithmique est, avec l'électronique, la base scientifique de l'informatique. Elle intervient dans
  - le logiciel (software)
  - le matériel (hardware). Un processeur est un câblage d'algorithmes simples (addition, multiplication, portes logiques etc.)

Algorithme : une définition.

- Un algorithme prend en entrée des données et fournit, en un nombre fini d'étapes, la réponse à un problème.
- Un algorithme est une série d'opérations à effectuer :
  - Opérations en séquence (algorithme séquentiel)
  - Opérations en parallèle (algorithme parallèle)
  - Opérations exécutées sur un réseau de processeur (algorithme réparti / distribué)
- Mise en œuvre de l'algorithme
  1. implémentation (plus général que le codage)
  2. écriture de ces opérations dans un langage de programmation (= codage)

  On obtient un programme

Algorithme vs Programme

- Un programme implémente un algorithme
- (Turing-Church) : les problèmes ayant une solution algorithmique sont ceux résolvables par une machine de Turing (théorie de la calculabilité)
- On ne peut pas résoudre tous les problèmes avec des algorithmes (indécidabilité algorithmique) – problème de l'arrêt (Turing 1936)
  - cet algorithme va-t-il s'arrêter ?
  - cet algorithme est-il juste ? (Rice 1951)

Complexité

Qualité d'un algorithme

- Deux algorithmes résolvants le même problème ne sont pas équivalents. 2 critères :

– Temps de calcul : lents vs rapides
– Mémoire utilisée : peu vs beaucoup

• On parle de complexité en temps (vitesse) ou en espace (mémoire)

Pourquoi faire des algorithmes rapides ?

• Pourquoi faire des algos efficaces ? Les ordinateurs vont très vite !

• C'est faux, on n'a toujours pas assez de puissance de calcul (météo, mécanique des fluides, IA, trafic routier, séquençage du génome etc.)

Quelques précisions

Seconde Loi de Moore (Gordon Moore - 1975) :

• le nombre de transistors des microprocesseurs sur une puce double tous les deux ans.

Elle est restée vraie de 1960 à 2000.

Depuis la dissipation thermique limite la taille des puces, on a plutôt tendance à multiplier le nombre de cœurs de processeurs.

La puissance de calcul est la capacité à effectuer un certain nombre de calcul en un temps donné.

Nous discuterons plus en détail des conséquences économiques et environnementales.

Exemple • Hypothèse optimiste (Loi de Moore) : la puissance de calcul double tous les deux ans.

• Mon programme en n^2 se termine en un temps satisfaisant avec n = 10.000

Quand pourrais-je le faire avec n = 1.000.000 ?

- quelques précisions

• Mon programme en n^2 : cela signifie qu'il réalise n^2 opérations pour une donnée de taille n.

• se termine en un temps satisfaisant avec n = 10.000 : il a donc réalisé n^2 = (10.000)² = 10^8 = 100.000.000 opérations en un temps satisfaisant.

- correction

• Hypothèse optimiste (Loi de Moore) : la puissance de calcul double tous les deux ans.

• Mon programme en n^2 se termine en un temps satisfaisant avec n = 10.000 Quand pourrais-je le faire avec n = 100.000 ?

– p = 100.000 q = 10.000 ; $p = 10 \times q$ donc $p^2 = 100 \times q^2$.

On a besoin de 100 fois plus de puissance.

– $2^6 = 64$ et $2^7 = 128$. Elle sera obtenue dans 7 × 2 = 14 ans !!!

- correction - suite

Mon autre algorithme est en n log n

• log n (logarithme de n) est une fonction croissante vers l'infini, comme n mais "très lente" Entre q = 10.000 et p = 100.000 On passe de 100.000 opérations à 1.000.000 d'opérations. Il ne faudra que 4 ans ($2^3 = 8$ $2^4 = 16$).

Complexité des algorithmes

• But :

– Avoir une idée de la difficulté des problèmes

– Donner une idée du temps de calcul ou de l'espace nécessaire pour résoudre un problème

• Cela va permettre de comparer des algorithmes.

• La complexité est exprimée en fonction du nombre de données et de leur taille.

• C'est très difficile…

– On considère que toutes les opérations sont équivalentes

– Seul l'ordre de grandeur nous intéresse.

– On considère uniquement le pire des cas (souvent ~ cas moyen)

Pourquoi faire des algos rapides ?

• Dans la vie réelle, ça n'augmente pas toujours !

• OUI et NON :

– Certains problèmes sont résolus.

– Pour d'autres, on simplifie moins et donc la taille des données à traiter augmente ! Réalité virtuelle : de mieux en mieux définie.

• Dès que l'on résout un problème on le complexifie !

Algorithme : vitesse

Rapide un algorithme qui met un temps polynômial en n (nombre de données) pour être exécuté : n^2, n^8

Lent un algorithme qui met un temps exponentiel : 2^n

Pour certains problèmes (voyageur de commerce, remplissage de sac à dos de façon optimale) on ne sait même pas s'il existe un algorithme rapide.

On connait des algorithme exponentiels en temps 2^n.

• 100^2 = 10.000 100^3 = 1.000.000

• 2^{100} = 1267650600228229401496703205376