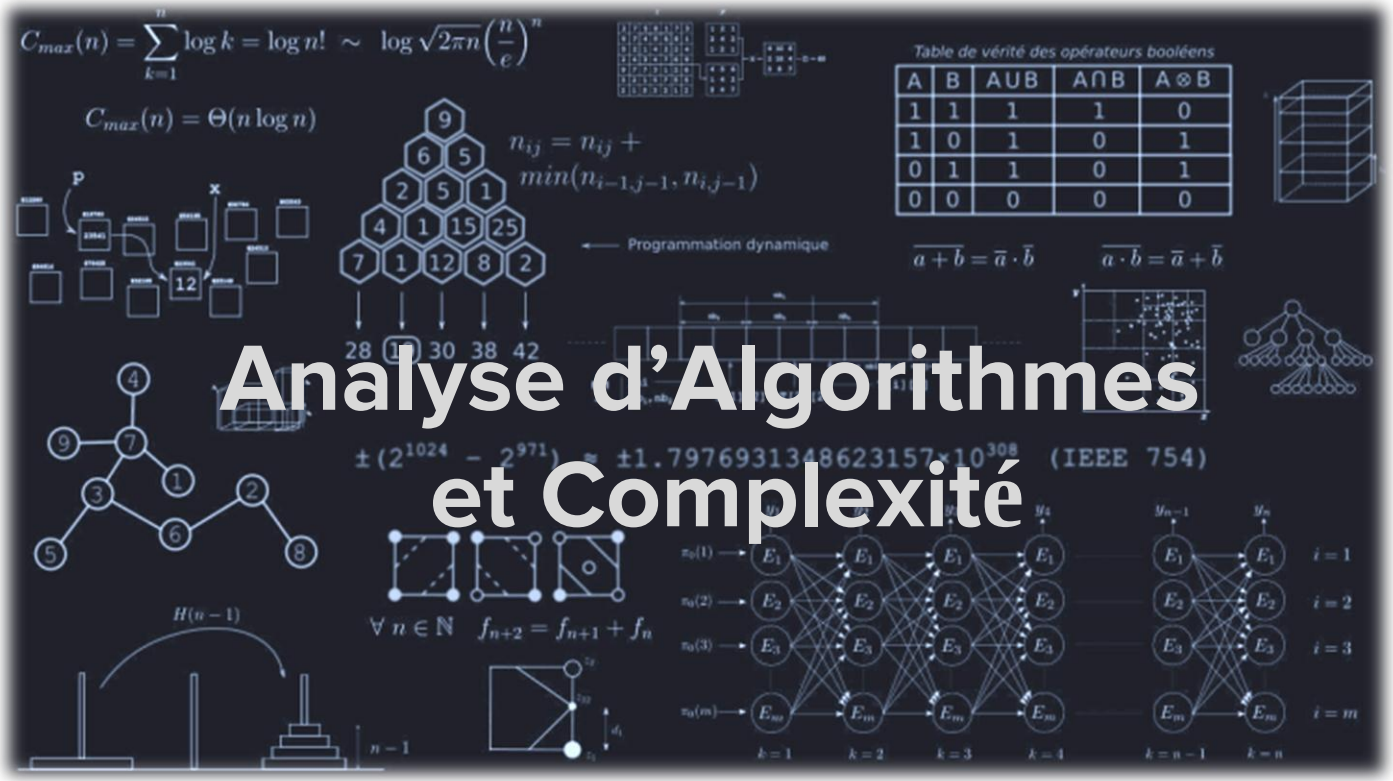

Analyse d'Algorithmes et Complexité

Pr Bekkari

Année universitaire 2021 - 2022

Objectifs du cours

- Concevoir des algorithmes simples, corrects et efficaces pour résoudre un problème;
- Analyser les performances d'un algorithme : notion de complexité;
- Algorithmes fondamentaux : description et complexité ;
- Structures de données performantes : linéaires et arborescentes.

Méthodologie d’enseignement et d’évaluation

- Cours magistraux (Diapositives). 2H présentiel (et à distance)
- Travaux dirigés. 2H
- Evaluation : 2 contrôles (80%) + TD/Devoir/Assiduité (20%)
- Tout est sujet à examen.
- Lectures personnelles :
  - Introduction to Algorithms par Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest
  - Types de données et algorithmes par Christine Froidevaux, Marie-Claude Gaudel, Michèle Soria

Planification du cours

Plan

- Introduction
- Notion de complexité
- Algorithmes de Tris
- Structures de données linéaires
  - Liste, Pile, file.
- Structures de données arborescentes
  - Arbres

Chapitre 1 : Introduction

Algorithmes classiques

- Chemin dans un graphe : votre GPS, votre téléphone IP, tout ce qui est acheminement (la Poste)
- Flots : affectations, emploi du temps, bagages (aéroport), portes d'embarquement
- Scheduling (ordonnancement) fabrication de matériel dans les usines, construction automobiles
- Compression/Décompression : MP3, xvid, divx, h264 codecs audio et vidéo, tar, zip.
- Cryptage : RSA (achat électronique)
- Internet : Routage des informations, moteurs de recherche
- Simulation : Prévision météo, Explosion nucléaire
- Traitement d'images (médecine) , effets spéciaux (cinéma)

Algorithmique : une science très ancienne

- Remonte à l'Antiquité:
  - Euclide : calcul du pgcd de deux nombres,
  - Archimède : calcul d'une approximation de π

- Le mot algorithme vient du mathématicien arabe du 9ème siècle Al Khawarizmi

- L'algorithmique reste la base dure de l'Informatique. Elle intervient dans le software (logiciel) et le hardware.

Algorithme

- Définition: Un algorithme est suite finie d'opérations élémentaires constituant un schéma de calcul ou de résolution d'un problème.

- Double problématique de l'algorithmique ?
  - Trouver une méthode de résolution (exacte ou approchée) du problème.

    Exemple: $ax^2 + bx + c$ *vs* $ax^5 + bx^4 + cx^3 + dx^2 + ex + f$
  - Trouver une méthode efficace.

=>Savoir résoudre un problème est une chose, le résoudre efficacement en est une autre, ou encore montrer qu'il est correcte …!!

Algorithme

- Un algorithme prend en entrée des données et fournit un résultat permettant de donner la réponse à un problème

- Un algorithme = une série d'opérations à effectuer :
  - Opérations exécutées en séquence => algorithme séquentiel.
  - Opérations exécutées en parallèle => algorithme parallèle.
  - Opérations exécutées sur un réseau de processeurs => algorithme réparti ou distribué.

- Mise en œuvre de l'algorithme
  - Implémentation
  - Ecriture de ces opérations dans un langage de programmation.

  Donne un programme.

Algorithmes

- Tous les algorithmes ne sont pas équivalents. On les différencie selon deux critères:
  - Temps de calcul : lents vs rapides
  - Mémoire utilisée : peu vs beaucoup
- On parle de complexité en temps (vitesse) ou en espace (mémoire utilisée)

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 les algorithmes
- Exprimée en fonction du nombre de données et de leur taille.
- A défaut de pouvoir faire mieux :
  - On considère que les opérations sont toutes équivalentes
  - Seul l'ordre de grandeur nous intéresse.
  - On considère uniquement le pire des cas
- Pour n données on aura : O(n) linéaire, $O(n^2)$ quadratique O(n log(n)),…

Pourquoi faire des algorithmes rapides ?

- Pourquoi faire des algorithme efficaces ? Les ordinateurs vont plus vite !
- Je peux faire un algorithme en n^2 avec n=10000
- Je voudrais faire avec n=100000. Tous les 2 ans la puissance des ordinateurs est multipliée par 2 (optimiste). Quand est-ce que je pourrais faire avec n=100000 ?
- Réponse:
  - p=100000 q=10000; p=10*q; donc $p^2 = 100*q^2$. Donc besoin de 100 fois plus de puissance
  - $2^6 = 64$, $2^7 = 128$ donc obtenue dans 7*2 ans=14 ans !!!

Algorithmes : vitesse

- On peut qualifier de rapide un algorithme qui met un temps polynomial en n (nombre de données) pour être exécuté. Exemple n^2, n^8

- Pour certains problèmes : voyageur de commerce, remplissage de sac-à-dos de façon optimum. On ne sait pas s'il existe un algorithme rapide. On connaît des algorithmes exponentiels en temps : 2^n.

Algorithme : preuve

- On peut prouver les algorithmes !
- Après la conception de chaque algorithme, le programmer doit impérativement se poser les trois questions suivantes :
  - Terminaison : le programme termine-t-il en un nombre fini d'opérations ?
  - Correction : le résultat retourné est-il toujours celui qui est attendu ?
  - Complexité : comment évoluent le temps de calcul et l'espace mémoire nécessaire en fonction de la taille du problème à traiter ?

Algorithme : preuve

- Un algorithme est dit totalement correct si pour tout jeu de données il termine et rend le résultat attendu
- C'est assez difficile, mais c'est important
  - Codage/Décodage des données. Si bug alors tout est perdu
  - Centrale nucléaire
  - Avion
- Attention : un algorithme juste peut être mal implémenté

Algorithme : résumé

- C’est ancien
- C’est fondamental en informatique
- Ça se prouve
- On estime le temps de réponse et la mémoire prise

Algorithmes et structures de données

- L'algorithme est souvent indépendant de l'ordinateur utilisé ainsi que du langage.
- L'algorithme est très lié aux données qu'il utilise et vice-versa.
- Suivant la taille du problème le but n'est pas toujours le même :
  - petit problème : Le but est d'arriver à la bonne solution.
  - gros problème : Il faut aussi arriver à la bonne solution mais rapidement et sans dépasser les capacités mémoire de l'ordinateur.
- Les algorithmes peuvent être simples ou difficiles ainsi que les structures de données.

Algorithmes et structures de données

- En informatique il y a deux types de personnes
  - Ceux qui écrivent les algorithmes
  - Ceux qui les implémentent
- Pour se comprendre on va améliorer le langage
  - On définit des choses communes bien précises.
  - On essaie de regrouper certaines méthodes ou techniques
- En Informatique
  - Langages : variables, boucle, incrémentation etc. . .
  - Regroupement : structures de données

Structures de données

- Une **structure de données** est une manière particulière de stocker et d'organiser des données dans un ordinateur de façon à pouvoir être utilisées plus efficacement.

- Différents types de structures de données existent pour répondre à des problèmes très précis :
  - B-arbres dans les bases de données
  - Table de hash par les compilateurs pour retrouver les identificateurs.

Structures de données

- Ingrédient essentiel pour l'efficacité des algorithmes.
- Permettent d'organiser la gestion des données
- Une structure de données ne regroupe pas nécessairement des objets du même type.

Structures de données et langage orienté objets

- Les structures de données permettent d'organiser le code
- Une Sdd correspond à une classe contenant un ensemble d'objets
- Deux parties:
  - Une visible correspondant aux opérations que la structure de données peut effectuée.
  - Une cachée qui contient les méthodes internes permettant de réaliser les opérations de la Sdd.
- La partie visible de la Sdd est parfois appelée API de la Sdd : Application Programming Interface, autrement dit l'interface de programmation de la Sdd qui permet son utilisation.

Conclusion

- Un bon algorithme est comme un couteau tranchant, il fait exactement ce que l'on attend de lui avec un minimum d'efforts.
- Concevoir un algorithme correct et efficace pour résoudre un problème permet d'avoir le résultat rapidement en utilisant le minimum de ressources.
- Le test de programmes peut être une façon très efficace de montrer la présence de bugs mais est désespérément inadéquat pour prouver leur absence.