Algorithmique et Structures de Données

Cours 1 - Algorithmique et Complexité

H. Djerroud

LIASD - Université Paris 8

26 November 2019

Objectif du cours

- Fournir une boîte à outils contenant :
  - Des structure de données permettant d'organiser et d'accéder efficacement aux données
  - Les algorithmes les plus utilisés pour résoudre des problèmes courants
- Implantation des structures de données et des algorithmes en langage C
- Utilisation des structures de données et des algorithmes définis dans la bibliothèque standard en C++ (STL)

Plan de cours

Cinq chapitres :

- Notions de base
- Les structures de données séquentielles
- Les structures de données linéaires
- Les arbres
- Les graphes

Organisation du cours

- Cours théoriques + TD/TP
  - 5 Chapitres et 10 séances de cours
  - Chaque chapitre sera traité en ($\pm$) deux séances de cours et deux séances de TD/TP
- TD/TP
  - 5 TP et 10 séances de TD/TP
  - Chaque TP est à rendre sur moodle à la fin de la dernière séance de TP de chaque chapitre (soit 5 TP à rendre)
- Projet
  - Individuels ou en binôme (pas de trinôme)
  - En langage C
- Évaluation : $Note\ finale = 0.4 * Projet + 0.6 * DE$

Notes de cours

- Cours + TD/TP disponible sur moodle avant chaque cours, clé d'inscription : **ALSTR2**
- Consultation de cours obligatoire avant chaque séance (au moins 1 jours avant).

Définition d'un algorithmique

Définition wikipedia

Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre une classe de problèmes.

- Un problème P peut avoir plusieurs solutions $S_1, S_2, \ldots, S_n$
- Comment comparer les solutions : $min(S_i)$?

Définition de la complexité

Définition

La complexité d'un problème P est la mesure de la quantité des ressources nécessaires pour résoudre le problème P.

Deux types de ressources :

- Complexité temporelle : le nombre d'étapes élémentaires nécessaire (temps nécessaire)
- Complexité spatiale : l'espace mémoire nécessaire

La complexité c'est quoi ?

A quoi ça sert :

- Classer les problèmes selon leur difficulté
- Classer les algorithmes selon leur efficacité
- **Comparer les algorithmes résolvant un problème donné afin de faire un choix éclairé sans devoir les implémentent**

Les cas possibles :

- **Pire cas**
- meilleur cas
- moyenne

Comment calculer la complexité

Étape 1 :

- Définir l'unité de mesure : la notion d'instructions élémentaires

Étape 2 :

- Calculer le nombre d'instructions élémentaires $T(n)$ en fonction de la taille des données n

Étape 3 :

- Calculer une estimation d'une borne supérieure $f(n)$ de $T(n)$

Exemple de calcul de la complexité

La fct Factorielle : $n! = n * (n-1) * (n-2) * \ldots * 1$ avec $0! = 1$

```
int factorielle(n) {
    fact = 1;              // initialisation(1)
    i = 2;                 // affectation (1)
    while (i <= n){        // comparaison (1)
        fact = fact*i;     // affectation + multip (2)
        i = i + 1;         // affectation + addition (2)
    }
    return fact            // retour (1)
}
```

Temps de calcul $= 1+1+(2+2+1)*(n-1)+1+1 = 5n-1$ op

Les opérations

- Affectation : 1 opération
- +, *,-,/ ... : 1 opération
- Condition : On choisit la partie la plus grande

La complexité asymptotique

Classes de complexité