Ordre du jour :
- Cours (1h)
- Puis TP 1 (binaire, hexa...)

Développement informatique

Cours 2 :

-Introduction aux algorithmes

-Les langages de programmation

Définition d’un algorithme

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

La notion de problème peut être vue dans un sens large, il peut s'agir d'une tâche à effectuer, comme trier des objets, assigner des ressources, transmettre des informations

Exemple commun : la recherche d'un mot dans le dictionnaire

Propriété d’un algorithme

Un algorithmes présente les propriétés suivantes :

- Des entrées (de d'information, un jeu de donnée ; un type (int, float…))
- Des sorties
- Une définition précise : Chaque étape d'un algorithme doit être définie précisément, les actions à transposer doivent être spécifiées rigoureusement et sans ambiguïté pour chaque cas. (ex: résolution d'une équation du second degré avec un déterminant négatif → utilisation des nombres complexes ? Calcul matriciel/conditionnement des matrices)
- Il doit être effectif : toutes les opérations doivent pouvoir être simulées par un homme en un temps fini.
- Une finitude (pour chaque jeu de données d'entrée, l'algorithme doit fournir un résultat grâce à un nombre fini d'opérations.)

=> Notion de complexité, choix de l’implémentation

Notion de complexité algorithmique

Deux grands type de complexité :

- Complexité temporelle (ou en temps) : temps de calcul ;

La complexité (temporelle) d'un algorithme est le nombre d'opérations élémentaires (affectations, comparaisons, opérations arithmétiques) effectuées par un algorithme. Ce nombre s'exprime en fonction de la taille n des données.

On distingue 3 formes :

- la complexité dans le meilleur des cas
- complexité dans le pire des cas
- la complexité en moyenne

- Complexité spatiale (ou en espace) : l'espace mémoire requis par le calcul

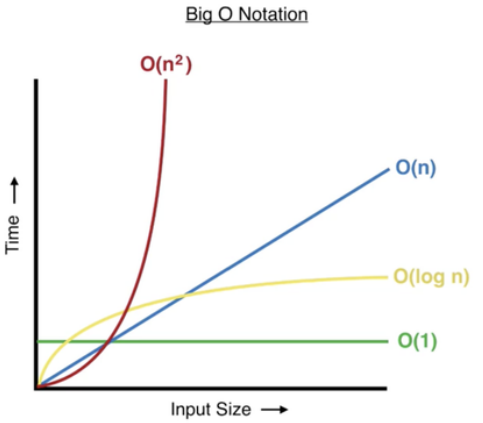
Notion de complexité algorithmique

On défini aussi :

- La complexité pratique qui est une mesure précise des complexités temporelles et spatiales pour un modèle de machine donné.

- La complexité (théorique) qui est un ordre de grandeur de ces coûts, exprimé de manière la plus indépendante possible des conditions pratiques d'exécution.

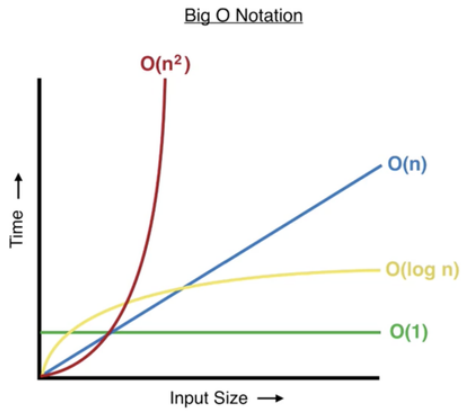
Classes de complexité.



- 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(nlog n) sont considérés comme rapides, comme l'évaluation de la valeur d'une expression composée de n symboles ou les algorithmes optimaux de tri.

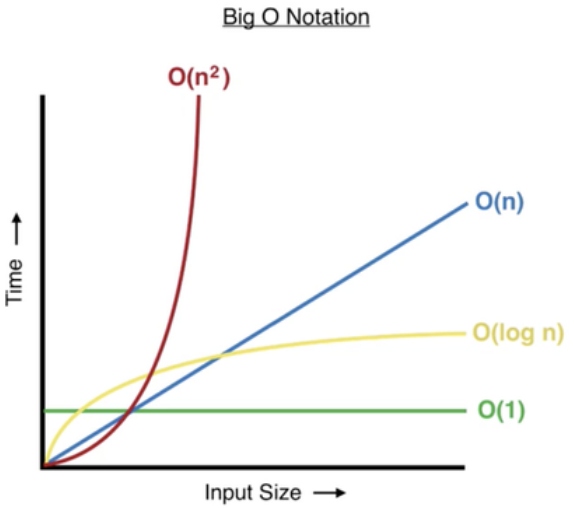
Classes de complexité.



- 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.

- 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.

Classes de complexité.



Quelque exemples :

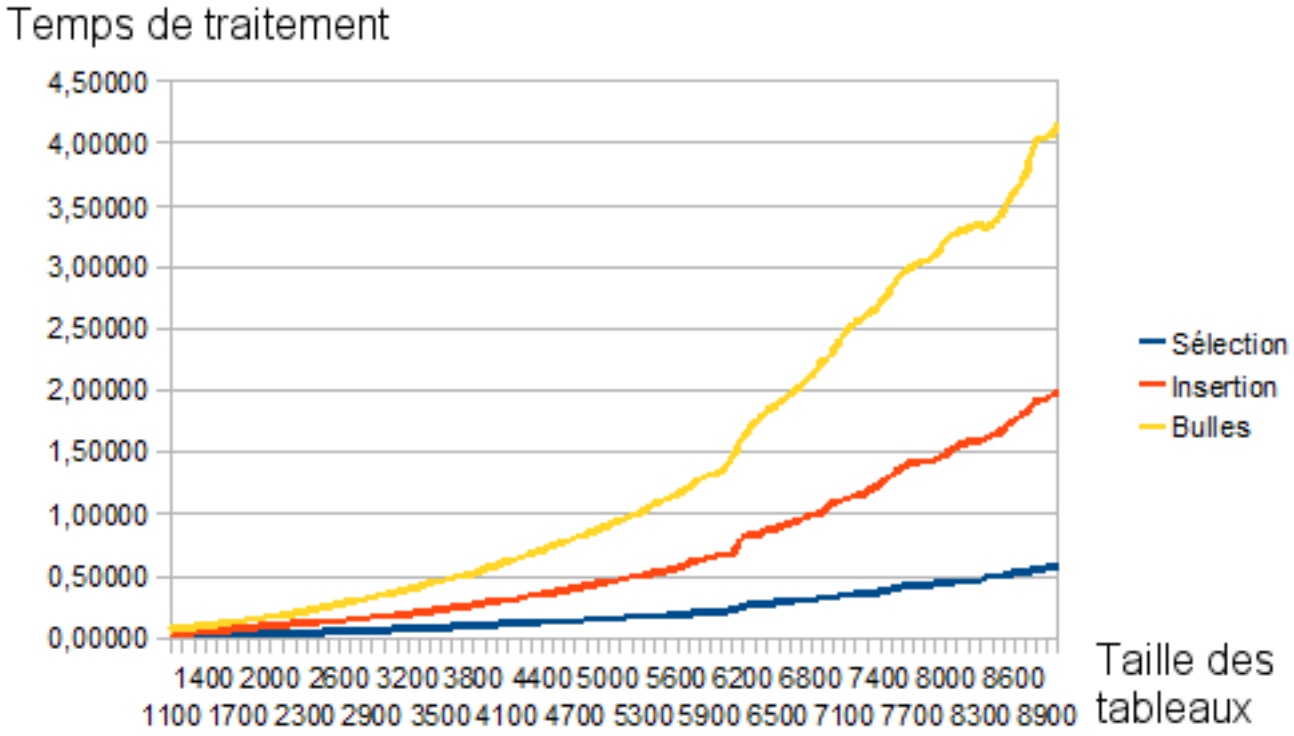
- O(n*log n) : tris par échanges, par fusion, tri rapide, tri par tas

- O(n^k) quadratique, polynomial : tri a bulle, tri par insertion

Cf https://fr.wikipedia.org/wiki/Algorithme_de_tri

Classes de complexité.

Illustration de la complexité de quelques algorithmes de tri



Classes de complexité.

Exercices :

- Déterminer la classe d'un algorithme donnant toutes les permutations de lettres

-Estimer le temps d'exécution de l'algorithme pour n=20 (en considérant que votre algorithme permet de faire un permutation par nanoseconde)

Classes de complexité.

Déterminer la complexité algorithmique de ces deux fonctions de recherche :

```
static int plusPetit (int[] x){
    int k = 0, n = x.length;

    for(int i = 1; i < n; i++)
        // invariant : k est l'indice du plus
petit
        // élément de x[0..i-1]
        if(x[i] < x[k])
            k = i;
    return k;
}
```

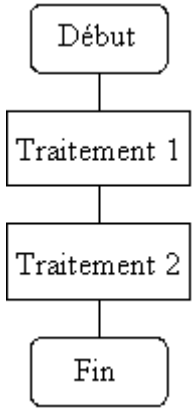
```
static int rechercheDichotomique(int[] t,
int x){
    int m, g, d, cmp;

    g = 0; d = N-1;
    do{
        m = (g+d)/2;
        if(t[m] == x)
            return m;
        if(t[m] < x)
            d = m-1;
        else
            g = m+1;
    } while(g <= d);
    return -1;
}
```

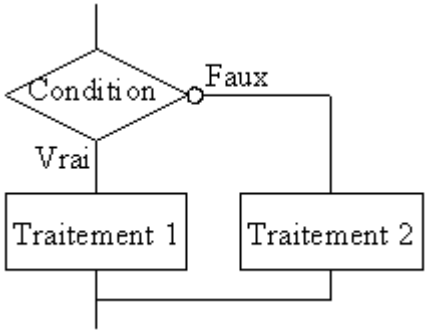
Convention d’écriture d’un algorithme

- **Les organigrammes**

  - Une représentation graphique, avec des carrés, des losanges, etc.

  - Cette représentation est quasiment abandonnée,





Convention d’écriture d’un algorithme

- Pseudo-code :

-Il ressemble à un langage de programmation authentique dont on aurait évacué la plupart des problèmes de syntaxe. Ce pseudo-code est susceptible de varier légèrement d’un livre (ou d’un enseignant) à un autre.

-Il est purement conventionnel ; aucune machine n’est censée le reconnaître (indépendant du langage qui sera utilisé pour l’implémentation)

Entrées : Un entier positif n.
Sorties : Le n-ième nombre de Fibonacci.

```
1 FIBO-PG-IT(n)
2 Créer un tableau T de longueur n ; T[0] :=0 ; T[1] :=1;
3 pour i allant de 2 à n faire
4   T[i] := T[i − 1] + T[i − 2]
5 retourner T[n]
```

```
Variable Temp en Entier
Début
Ecrire "Entrez la température de l’eau :"
Lire Temp
Si Temp =< 0 Alors
  Ecrire "C’est de la glace"
SinonSi Temp < 100 Alors
  Ecrire "C’est du liquide"
Sinon
  Ecrire "C’est de la vapeur"
Finsi
Fin
```

Programmation

La programmation dans le domaine informatique est l'ensemble des activités qui permettent l'écriture des programmes informatiques :

Cinq grandes étapes :

- Spécification
- Conception
- Codage
- Test
- Documentation

Programmation

Phase de spécifications : Une spécification est un ensemble explicite d'exigences à satisfaire

- Spécification fonctionnelle
  - ◆ Les spécifications fonctionnelles générales :

    - décrivent les différentes procédures d'un même processus métier (ce que doit faire le programme, les cas qui peuvent se présenter, gestion des cas particuliers, des erreurs, impératifs de qualité de service : Performance, sécurité, robustesse et autres objectifs et contraintes que le système doit rencontrer)

    - elles sont généralement élaborées par la maîtrise d'ouvrage ;

  - ◆ Les spécifications fonctionnelles détaillées :

    - décrivent dans le détail les opérations et les tâches à exécuter par les utilisateurs, et les composants à développer pour accomplir ces tâches;

    - elles peuvent être élaborées par la maîtrise d'œuvre.

- Spécification d'architecture : description du système informatique dans lequel le produit sera implanté

Programmation

Phase de conception

- Détermination des entrées et formats :
  - claviers, fichiers, base de donnée, réseaux (tcp/udp...) : flux entrée
  - format binaire, texte, xml, json...

- Méthodes de traitement :
  - choix de algorithme (importance de la boite a outils)
  - choix de implémentation (procédurale, objet?)

- Détermination des sorties : flux de sortie

Programmation

Phase de codage :

- Choix du langage (C, C++, JAVA, C#, PHP…), Déterminé par:
  - L'existant (mélange de 2 langages ?, plate-forme ?)
  - Les compétences (ressources humaines)
  - Le projet (ex : web vs application desktop) et l'adéquation du langage
  - Les contraintes techniques (temps réel , mémoire disponible…)
- Choix des librairies , bibliothèques, framework

Programmation

Phase de tests

- Test unitaires : vérification d'une partie d'un programme, module, librairie...

- Test d'intégration : vérification de l'interaction entre module, programme...

- Tests fonctionnels : vérification de la conformité de l'application développée avec le cahier des charges initial (spécifications fonctionnelles et techniques)

- Tests IHM

- Test de performances/monté en charge

- Tests d'installation

=> les tests unitaires et d’intégrations sont le plus souvent menés en parallèle

Programmation

Phase de documentation

- Dans le code (description des classes, des méthodes, des procédures...) puis extraction ou utilisation en ligne (commentaires et annotations dans le code...)

- Dans un document externe : paramètres d'entrée, format des fichiers, paramètres de configuration...

Langages de programmation

Définition : un langage de programmation est une notation conventionnelle destinée à formuler des algorithmes et produire des programmes informatiques qui les appliquent. D'une manière similaire à une langue naturelle, un langage de programmation est composé d'un alphabet, d'un vocabulaire, de règles de grammaire et de significations (wikipedia)

Il permet de définir :

-La structure des données

-Les manipulation à effectuer sur ces données

.

Langages de programmation

Il existe de nombreux critères de classification :

- Langage machine (assembleur) – langage évolué (Java,...) - langage de liaison / inter-application (javascript)
- Langage bas niveau – Langage haut niveau
- A typage fort/typage faible,
- Langage compilé – interprété – Pseudo Compilé (code intermédiaire)
  - Notion de code intermédiaire

    PHP : code 'tokenisé', l'analyseur syntaxique convertit chaque instruction texte en un lien direct vers la procédure exécutable. Ce code est conservé en mémoire pendant un certain temps

    Java : byte code,

    C# : common intermediate language.
  - Compilation JIT

Langages de programmation

Exemple de code intermédiaire java :

Code source :

```
1 public static void main(String[] args) {
2     int a = 1;
3     int b = 2;
4     int c = a + b;
5 }
```

Langages de programmation

Byte code

```
1  public static void main(java.lang.String[]);
2  descriptor: ([Ljava/lang/String;)V
3  flags: (0x0009) ACC_PUBLIC, ACC_STATIC
4  Code:
5  stack=2, locals=4, args_size=1
6  0: iconst_1
7  1: istore_1
8  2: iconst_2
9  3: istore_2
10 4: iload_1
11 5: iload_2
12 6: iadd
13 7: istore_3
14 8: return
15 ...
```

Byte code généré par la commande
javap -v

De-compilateur java en ligne : http://www.javadecompilers.com/

Langages de programmation

- En fonction des types de paradigmes pris en charge
  - Langages impératifs ou procéduraux (C, pascal, fortran),
  - Fonctionnel (Lisp, expression Lambda)
  - Logique (Système expert, PROLOG)
  - Déclaratif (HTML)
  - Orienté Objets (Java, C++)
  - Distribué (Erlang, E)

Cf : https://fr.wikipedia.org/wiki/Comparaison_des_langages_de_programmation_multi-paradigmes

Langages de programmation

Les langages peuvent utiliser plusieurs paradigme de programmation

Exemple pour java :

- programmation structurée,
- impérative,
- objet,
- fonctionnelle.
- événementielle, récursive, etc