
Introduction à l'algorithmique et à la programmation

Antoine FRABOULET

antoine.fraboulet@insa-lyon.fr

INSA de Lyon, Département Télécommunications, Services
et Usages

Introduction à l'algorithmique et à la programmation – p. 1

Algorithmique

- **Algorithmique** : Science qui étudie l'application des algorithmes à l'informatique
- **Algorithme** : Suite finie, séquentielle de règles que l'on applique à un nombre fini de données, permettant de résoudre des classes de problèmes semblables.
L'algorithme d'Euclide permet de trouver le P.G.C.D de deux nombre – Calcul, enchaînement des actions nécessaires à l'accomplissement d'une tâche.

Le Petit Robert

Introduction à l'algorithmique et à la programmation – p. 2

Présentation

- Introduction
- Bases de l'algorithmique
 - Structure des données
 - Structure des opérations
- Quelques méthodes de tri
- Gestion des listes

Introduction

- Un algorithme est motivé par la résolution d'une tâche
 - méthode indépendante de la machine
 - méthode indépendante du langage de programmation
 - résolution structurée
- algorithme = description des étapes de la méthode utilisée

Algorithmique et programmation

1. Analyse du problème
2. Conception d'une solution : algorithmique
choix de la représentation des données
choix de la méthode utilisée
3. Développement : programmation
choix du langage de programmation
choix de la machine utilisée
4. Tests

On ne s'intéressera donc ici qu'à la partie **algorithmique**, ou plutôt aux **structures de données** ainsi qu'aux **opérations de calcul** utilisées en algorithmique.

Quelques thèmes

- Tri :
permet de réarranger et de classer des données. De nombreuses méthodes existent pour trier un ensemble, elles se différencient par la suite des étapes effectuées.
- Recherche :
localiser des données dans un fichier. Les méthodes sont très variées et dépendent de l'organisation des données dans la mémoire.
- Traitement de chaînes :
manipulation de (longues) chaînes de caractères.
recherche de motifs dans des chaînes (*pattern matching*),
compression de fichiers, cryptographie.

Quelques thèmes (2)

- Algorithmes sur graphes :
résolution de problèmes complexes pouvant être représentés par une structure de données particulière (par exemple le problème du voyageur de commerce).
- Algorithmes mathématiques :
méthodes provenant de l'analyse numérique et de l'arithmétique. Problèmes traitant de l'arithmétique des entiers, des polynômes, des matrices.
- ...

Bases de l'algorithmique

Données

- variables
- structures
- tableaux
- pointeurs
- modèles dynamiques

Calcul

- instructions
- conditions
- boucles
- fonctions
- récursion

l'algorithmique c'est :
le choix d'un algorithme
le choix d'une structure de données
les deux sont indissociables

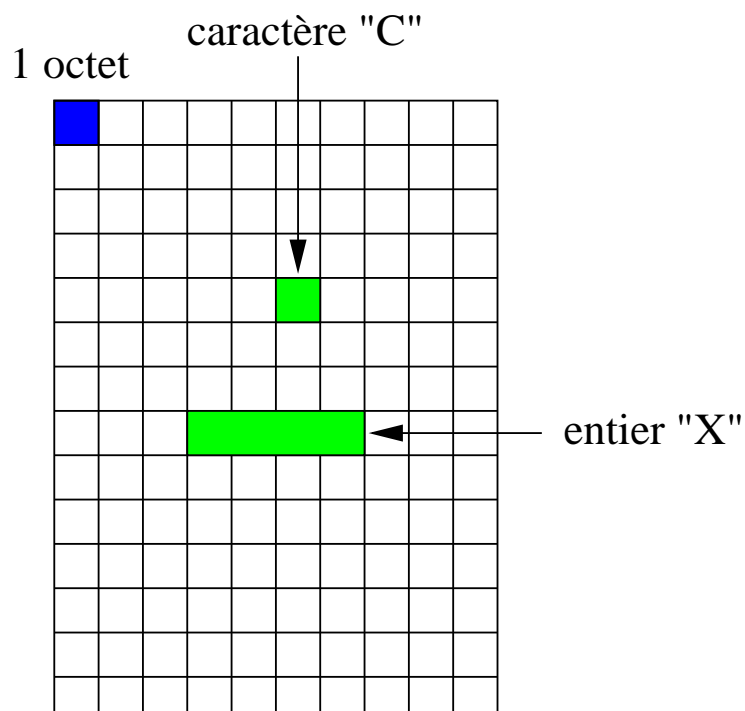
Structures de données

Un algorithme manipule des données :

- **Constante** : nombres, chaînes de caractères
 - 1, 2, 3, 4 ...
 - 3,1415
 - "bonjour"
- **Variable** : nom d'une valeur en mémoire
 - caractère
 - entier
 - nombre à virgule
 - ...

Structures de données

- Variable = nom d'un espace en mémoire



Structures de données : types de base

Les données sont *finies* : elles ont donc une taille maximale. Par exemple, les tailles usuelles des variables en C sont les suivantes:

| nom | taille (octets) | min | max | "unsigned" |
|-----------|-----------------|--------------------|--------------------|--------------|
| caractère | 1 (8 bits) | -128 | 127 | 255 |
| short | 2 (16 bits) | -32768 | 32767 | 65535 |
| int | 4 (32 bits) | -2^{31} | $2^{31} - 1$ | $2^{32} - 1$ |
| long | 4 (32 bits) | idem | idem | idem |
| float | 4 (32 bits) | $1.17 * 10^{-38}$ | $3.40 * 10^{+38}$ | |
| double | 8 (64 bits) | $2.22 * 10^{-308}$ | $1.79 * 10^{+308}$ | |

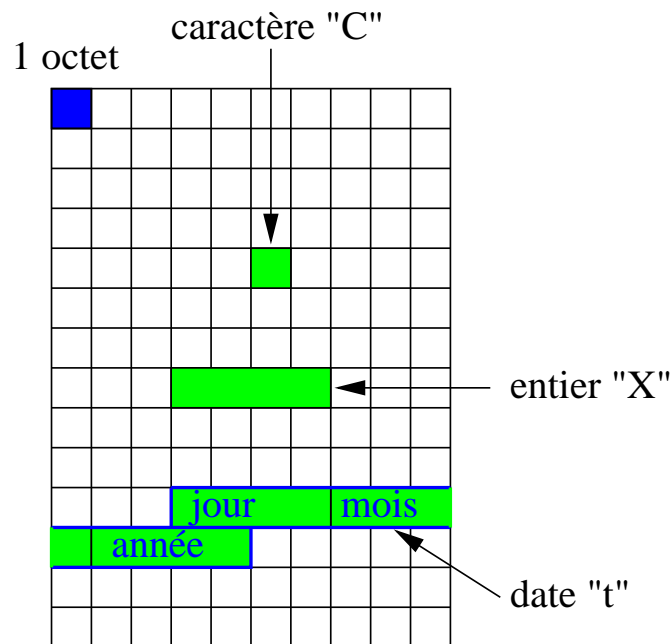
Structures de données

- **Type construit** : structure regroupant plusieurs variables.

```
structure date = (  
    entier jour  
    entier mois  
    entier annee  
)  
...  
structure rendez_vous = (  
    structure date date  
    structure personne nom  
    structure adresse lieu  
)
```

Structures de données

- Type structuré = création d'un nouveau type de variable = nom d'un espace en mémoire



Types de base

- La construction de types peut être faite en combinant tous les types de base et les types déjà définis.
- L'utilisation se fait en utilisant le point “.”

```
structure date (  
    entier jour  
    entier mois  
    entier annee  
)
```

```
structure date t1,t2  
t1.jour = 10  
t1.mois = 5  
t1.annee = 2008
```

```
...  
t2 = t1 (copie de toute la structure = 3  
variables)
```

Structures de données : tableaux

Collections de variables :

- **Tableaux :**

structure permettant d'accéder de façon indépendante à un ensemble de variables *de même type*.

entier t[12] tableau t contenant 12 entiers

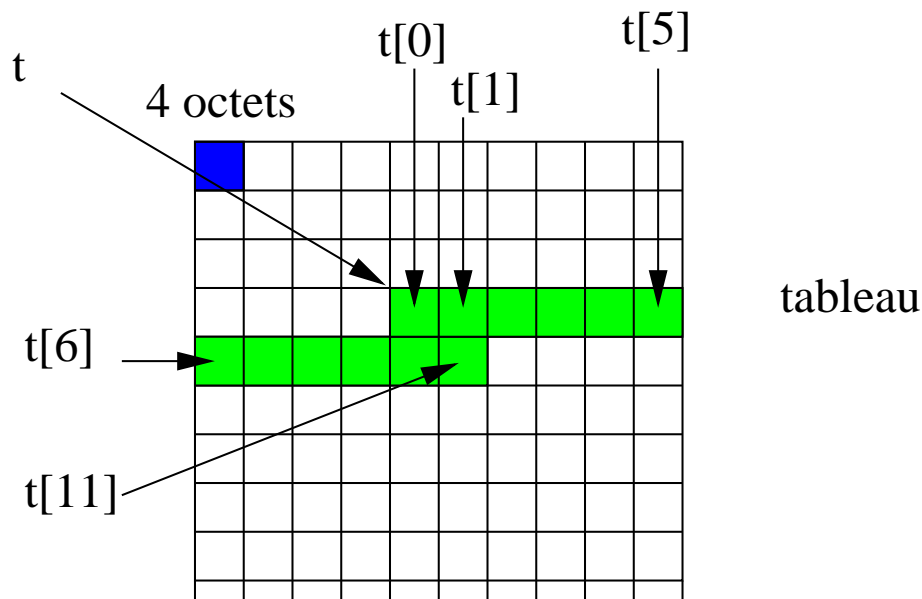
entier x

x = t[5] copie du 6^{ème} élément de t dans la variable x

t[6] = 42 modification du 7^{ème} élément

- Un tableau est de taille fixe et ses données sont contiguës en mémoire.
- Attention : le premier élément d'un tableau est souvent numéroté 0 en informatique.

Structures de données : tableaux (2)



Bases de l'algorithmique

- Structures de données
 - variables
 - structures
 - tableaux
- Structures de contrôle
 - ...

Structures de contrôle

Organisation des opérations :

- Opérations : $+$ $-$ $*$ $/$ $=$ \neq ...
 - $+$ $-$ $*$ $/$: opérations usuelles
 - comparaisons sur les types simples
 - opérateurs logiques (et, ou, négation)
 - la priorité, l'associativité et la commutativité des opérations sont respectées
- Instructions simples et expressions
 - $x = 123$
 - $y = x+w*3$

Exécution conditionnelle

- Exécution conditionnelle :

```
si [test est vrai] alors
    instructions
sinon
    instructions
fin si
```

instruction

- Exemple :

```
si a < b alors
    a = b
sinon
    b = a
finsi
```

Exécution répétitive

- Boucle de répétition fixe:

```
pour [ensemble de valeurs]
    faire
        instructions
    fin pour
```

instruction

- Exemple :

```
pour i=0 jusqu'à i<12 faire
    afficher t[i]
fin pour
```

Exécution répétitive

- Boucle de répétition “tant que”:

```
tant que [test vrai] faire
    instructions
fin tant que
```

} *instruction*

- Exemple :

```
i = 0
tant que i < 12 faire
    afficher t[i]
    i = i + 1
fin tant que
```

Structure de contrôle : fonctions

Découpage d'un programme en blocs ou *fonctions*.

- Permet de regrouper et nommer un ensemble d'instructions.
- Permet d'utiliser des paramètres.
- Permet de réutiliser une même portion de programme depuis plusieurs endroits.
- Interface = paramètres et valeur renvoyée.
- La valeur renvoyée peut être ignorée si on ne l'utilise pas.
- passage des paramètres
 - passage par valeur : seule la valeur de la variable est importante
 - passage par adresse : la variable peut être modifiée par la fonction

Exemple de fonction

```
entier TrouveMaximum(entier a, entier b)
{
    entier m                ( variable locale m )
    si a < b alors
        m = b
    sinon
        m = a
    fin si
    renvoyer m
}
```

```
...
i = TrouveMaximum(1,100)
j = TrouveMaximum(i,120)
...
```

Structure d'une fonction

```
entier TrouverMaximum( entier a, entier b )
{
    entier m
    si a < b alors
        m = b
    sinon
        m = a
    fin si
    renvoyer m
}
```

Fonctions récursives

- itérative : exécution répétitive sur des ensembles de données
- récursive : une fonction s'appelle elle même avec un ensemble de données à traiter plus petit

```
entier f(entier n)
{
    entier r
    si n = 0 alors
        r = 1
    sinon
        r = n * f(n-1)
    fin si
    renvoyer r
}
```

Structure d'un programme

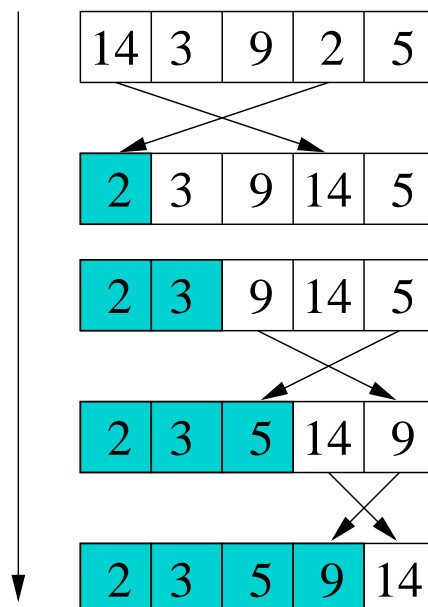
- On peut utiliser une variable ou une fonction uniquement si elle est déjà définie dans le programme
- Visibilité des variables
 - variables locales : internes à un bloc
 - variables globales : visibles dans tout le programme
- Le déroulement d'un programme commence
 - à la première instruction isolée
 - en appelant une fonction dont le nom est fixé à l'avance

Bases de l'algorithmique

- Structures de données
 - variables
 - structures
 - tableaux
 - ...
- Structures de contrôle
 - instructions
 - conditions
 - boucles
 - fonctions
 - récursion
- **Quelques méthodes de tri**

Tri par sélection

- on cherche l'élément de plus petite valeur dans le tableau
- le placer en tête du tableau
- recommencer avec le tableau moins la première case



Tri par sélection

```
TriSelection(entier t[], entier taille)
{
    entier i, j, min, tmp
    pour i=0 jusqu'à j < taille-2 faire
        min = i
        pour j = i+1 jusqu'à j < taille faire
            si t[j] < t[min] alors
                min = j
            fin si
        fin pour
        tmp = t[min]
        t[min] = t[i]
        t[i] = tmp
    fin pour
}
```

Autres tris

- insertion : méthode utilisée pour trier une main de jeu de carte. On prend les cartes une par une de la gauche vers la droite. Pour chaque carte on la place au bon ordre parmi les précédentes.
- bulle : les éléments les plus petits se «déplacent» vers le début du tableau. Les éléments contiguës sont échangés 2 à 2.
- par segmentation (ou tris rapide) : faire 2 ensembles séparés par une valeur *pivot*. Tous les éléments inférieurs au pivot sont rangés à gauche, tous les éléments supérieurs à droite, puis trier les deux nouveaux tableaux de la même manière (on s'arrête lorsque les tableaux ont un seul élément).

Tri par insertion

```
TriInsertion(entier t[], entier taille)
{
    entier i,j,tmp
    pour i=0 jusqu'à i<taille faire
        tmp = t[i]
        j = i
        tant que j>0 et t[j-1] > tmp faire
            t[j] = t[j-1]
            j = j-1
        fin tant que
        t[j] = tmp
    fin pour
}
```

Tri bulle

```
TriBulle(entier t[], entier taille)
{
    entier i,j,tmp
    pour i = taille jusqu'à i>1 avec i = i-1
    faire
        pour j = 1 jusqu'à j<i faire
            si t[j-1] > t[j] alors
                tmp = t[j-1]
                t[j-1] = t[j]
                t[j] = tmp
            fin si
        fin pour
    fin pour
}
```


Coût des algorithmes

Plusieurs algorithmes peuvent faire la même chose mais en effectuant un nombre d'opérations différent.

→ Différencier leurs coûts :

- coût en temps d'exécution
- coût en place mémoire
- nombre de transferts mémoire

Le nombre d'étapes et la place mémoire dépendent le plus souvent de la taille des entrées de l'algorithme (la taille des tableaux par exemple).

Complexité des algorithmes

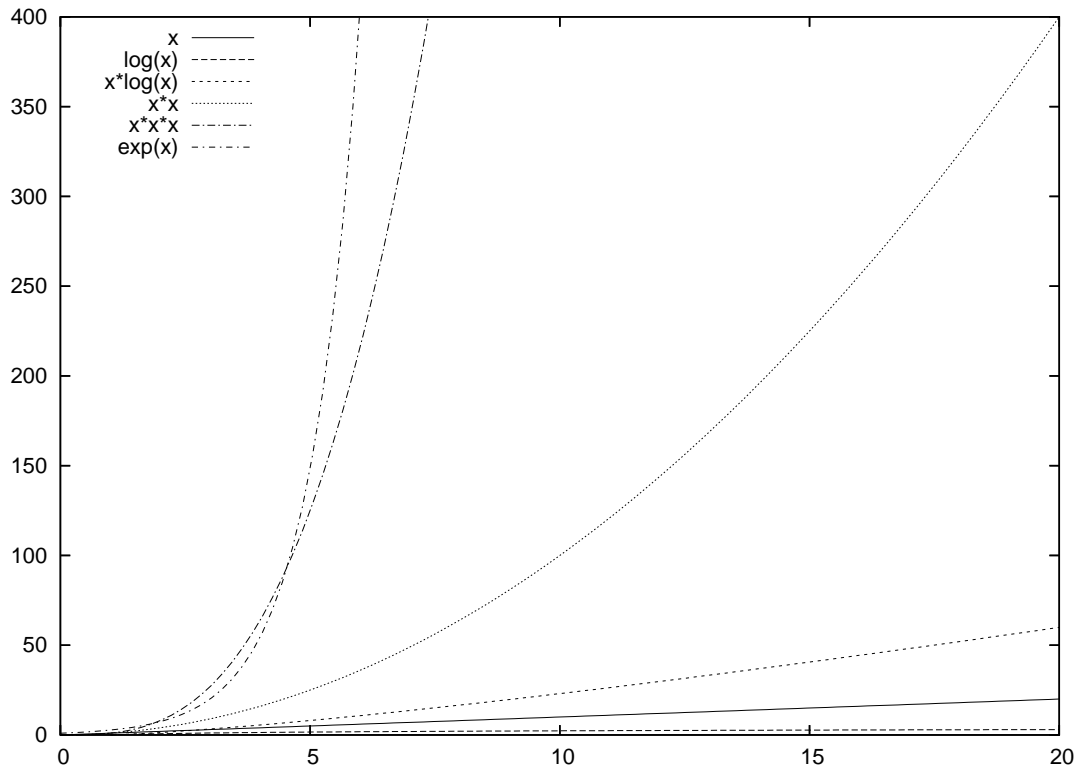
Tris par sélection : il y a 2 boucles imbriquées et le nombre d'opérations effectuées est proportionnel à n^2 .

On peut différencier l'analyse pour

- meilleur cas
- cas moyen
- pire cas

Le tri par insertion a la même complexité dans les 3 cas mais pas le tri bulle dont la complexité peut varier.

Différentes complexité



Comparaison des ordres de grandeur

- taille du problème / temps d'exécution

| | taille du problème : n | | | |
|-------------|--------------------------|----------|----------|-----------|
| | 2 | 2^4 | 2^6 | 2^8 |
| $\ln \ln n$ | 0 | 2 | 2.58 | 3 |
| $\ln n$ | 1 | 4 | 6 | 8 |
| n | 2 | 16 | 64 | 256 |
| $n \ln n$ | 2 | 64 | 384 | 2 048 |
| n^2 | 4 | 256 | 4096 | 65 536 |
| 2^n | 4 | 65 536 | 1.84e+19 | 1.15e+77 |
| $n!$ | 2 | 2.09e+13 | 1.26e+89 | 8.57e+506 |

Algorithmique pratique

- L'algorithme reste le même quel que soit la nature des données à traiter.
- Les modifications sont à faire dans des points très précis : ceux où l'on regarde le *contenu* des données.
- Pour changer le type des tableaux que nous savons pour l'instant trier et avoir, par exemple, des tableaux de carte de visite il suffit de changer les endroits où l'on **compare** les éléments des tableaux.

Les tris externes

- On peut vouloir avoir simultanément plusieurs classements sur un même ensemble de données (par nom, par ville, par date pour des rendez vous par exemple)
- On peut également vouloir avoir l'ordre des éléments d'un ensemble *sans les changer de place*
- Utilisation d'index pour avoir des classements sans modifier les données.

Bases de l'algorithmique

- **Structures de données**
 - variables
 - structures
 - tableaux
 - **pointeurs**

Pointeurs

- Un pointeur est une variable contenant une adresse
- Les pointeurs permettent de manipuler des objets sans avoir à les nommer (comme les tableaux !). Mais on peut allouer dynamiquement les de nouvelles variables.
- Les pointeurs peuvent être construits à partir de n'importe quel type

```
entier *pi : pi contient l'adresse d'un  
entier
```

```
structure date *t : t contient l'adresse  
d'une date
```

- On peut obtenir l'adresse d'un objet avec l'opérateur &

```
entier i
```

```
entier *pi = &i
```

```
i = 3
```

```
(*pi) = 4 : i est égal à 4
```

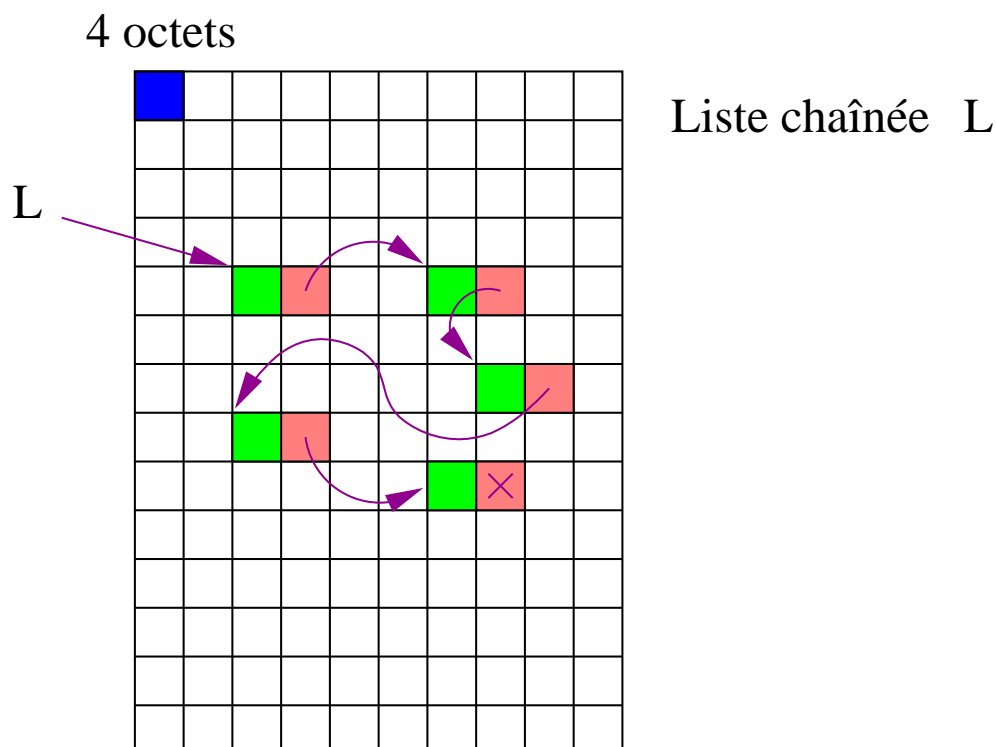
Modèles de données dynamiques

- **Structures dynamiques** : Arbres, listes chaînées, piles, files ...
structures dynamiques dont la taille peut varier au cours de l'exécution du programme. L'accès aux informations stockées dépend de la structure.

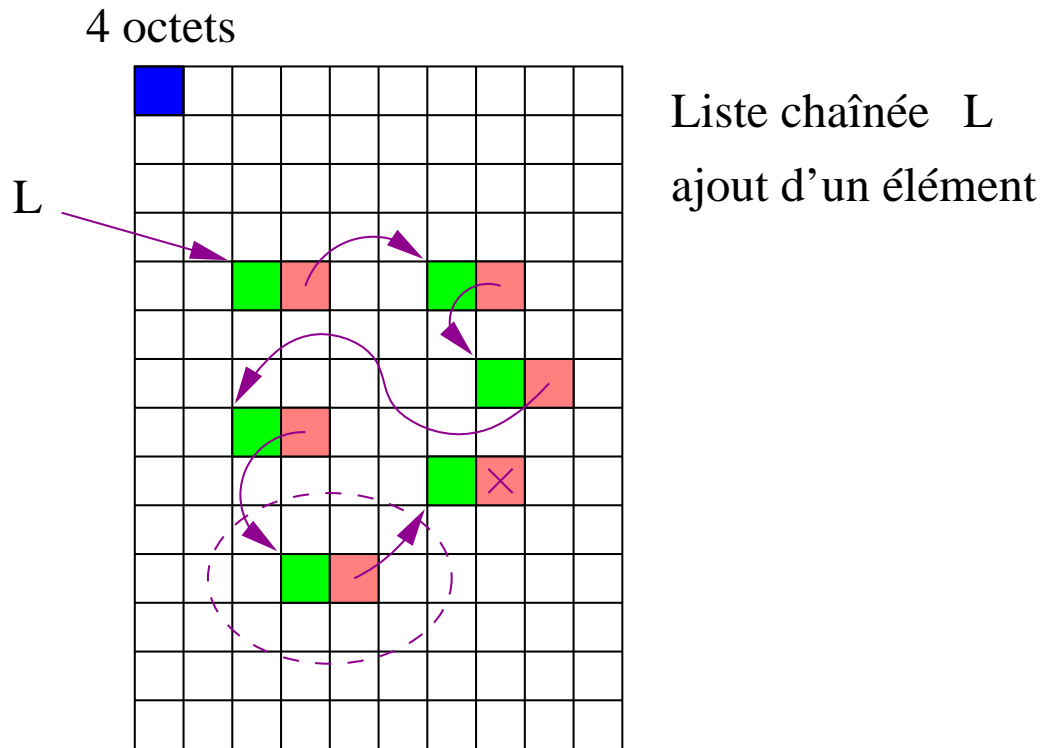
Ce type de structure utilise des **pointeurs**

```
structure elmt_liste {  
    int valeur;  
    structure elmt_liste* suivant;  
}
```

Structures de données : liste (2)



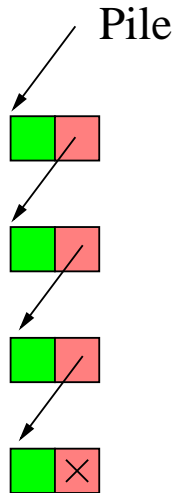
Structures de données: liste (3)



Structures de données

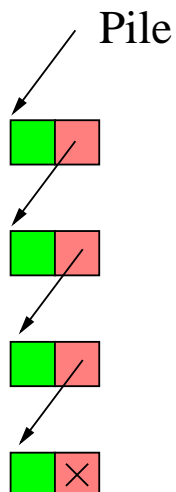
- Choix de manipulation des structures
 - Listes
 - Piles (Last In First Out)
 - Files (First In First Out)

Structures de données : Piles



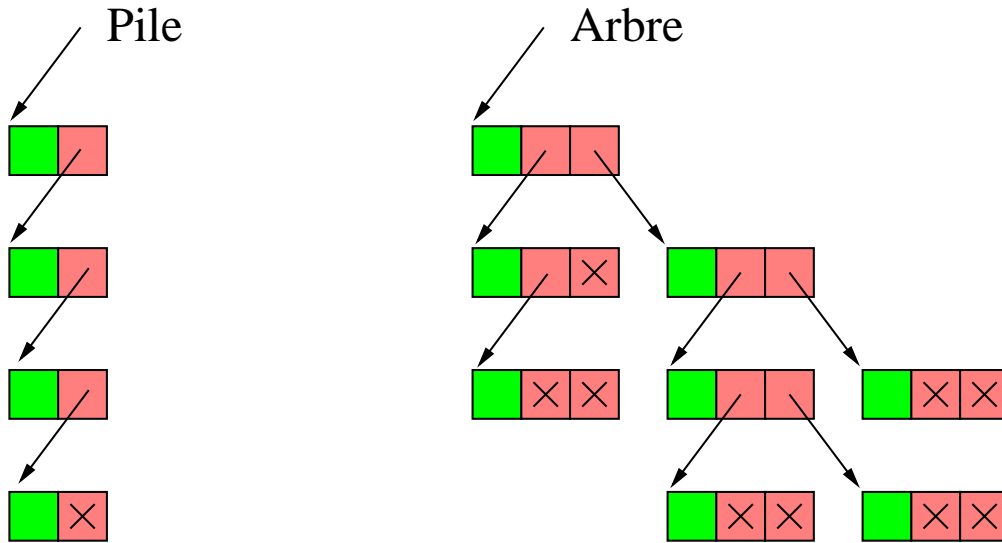
- ajouter toujours au sommet
- seul le sommet est visible
- retirer toujours au sommet

Structures de données : Files



- ajouter toujours au sommet
- seul le dernier est visible
- retirer toujours à la fin
- on peut garder un pointeur sur la fin pour être plus efficace

Structures de données



Conclusion

- données
 - variables
 - structures
 - tableaux
 - pointeurs
 - modèles dynamiques
- calcul
 - instructions
 - conditions
 - boucles
 - fonctions
 - récursion

l'algorithmique c'est :
le choix d'un algorithme
le choix d'une structure de données
les deux sont indissociables

Conclusion

- Dans un programme de grande taille le temps d'exécution est dominé par un poignée d'algorithmes
- Beaucoup de programmes sont sur-optimisés. Il n'est pas indispensable de trouver tout le temps le «meilleur» algorithme partout, sauf si ces derniers sont des tâches conséquentes ou très répétitives.
- Un mauvais choix d'algorithme pour une partie critique peut rendre le programme entier inutilisable.

Quelques livres

- Initiation à l'algorithmique et aux structures de données, Courtin & Kowarski, Dunod, 1989
- Introduction à l'algorithmique, Cormen, Leiserson & Rivest, Dunod, 1994
- Algorithmes en langage C, Sedgewick, InterEditions, 1991