

## Chap 6 : Les traitements avancés

### Leçon 1 : Méthodes de tri

#### I-Introduction:

→ **Définition:** Un algorithme de tri est une suite finie d'instructions servant à réordonner une séquence d'éléments suivant un critère fixé a priori. Ce critère est en effet une relation d'ordre total sur les éléments à trier.

La conception d'un algorithme de tri dépend du support matériel de la séquence de valeurs à trier (en mémoire centrale ou sur une mémoire secondaire).

**Mémoires centrales:** rapides (en nanosecondes) mais de taille limitée(en Mo)

**Mémoires secondaires:** lentes(en microsecondes) mais de grande taille (en Go).

Les algorithmes de tri vont en devoir tenir compte.

Les algorithmes de tri que nous allons définir traitent des tableaux situés dans la mémoire centrale.

#### Remarques:

- Il faut faire la distinction entre tri d'un grand nombre d'éléments et le tri de quelques éléments.
- On peut trier autres types que les entiers. Il suffit de disposer d'un domaine de valeurs muni d'une relation d'ordre total. On peut donc trier des caractères, des mots en ordre alphabétique...

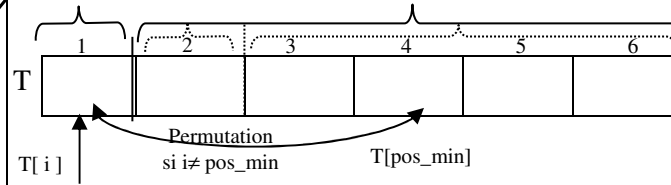
#### II-Les méthodes de tri:

On a choisi de trier les séquences par ordre croissant (de plus petit au plus grand)relativement à un ordre total noté  $\leq$

##### → Le tri par sélection:

#### 1.1. Principe:

- Commencer par  $i=1$  et on cherche la position de l'élément le plus petit du tableau ( $pos\_min$ ).
- Une fois cet emplacement trouvé, on compare son contenu avec  $T[1]$  et s'ils sont différents( $T[1] \neq T[pos\_min]$ ), on permute l'élément de l'emplacement trouvé par l'élément de la première position  $T[1]$  sinon  $T[1]$  reste à sa place → Après ce parcours le premier élément est bien placé.
- On recommence le même procédé pour le reste du tableau ( $T[2], \dots, T[n]$ ), ainsi on recherche le plus petit élément de cette nouvelle partie du tableau et on l'échange éventuellement avec  $T[2]$ .
- Ainsi de suite jusqu'à la dernière partie du tableau formée par les deux derniers éléments( $T[n-1], \dots, T[n]$ ).



**N.B:** La recherche du plus petit élément d'un tableau sera assuré par une fonction renvoyant la position du minimum. Ce minimum est éventuellement répété plusieurs fois dans T; on décidera de choisir le premier.

#### Remarque:

nombre d'opérations nécessaires pour trier tout le tableau T:

à chaque itération on démarre à l'élément  $T[i]$  et on le compare successivement à  $T[i+1]$ ,  $T[i+2]$ , ...,  $T[n]$  on fait donc  $n-i$  comparaison.

On commence avec  $i=1$  et on finit avec  $i=n-1$

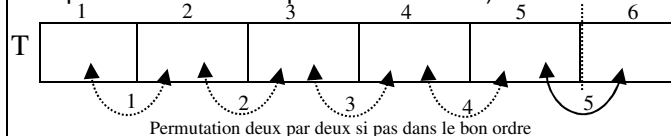
Donc, on fait  $(n-1)+(n-2)+\dots+2+1 = n(n-1)/2$  comparaisons et au maximum  $n-1$  permutations.

##### → Le tri à bulle:

#### 2.1. Principe:

Faire remonter le plus grand élément du tableau en comparant les éléments successifs.

- On commence par  $i=1$ , on compare le 1er( $T[1]$ ) et le 2ème élément( $T[2]$ ) du tableau, s'il ne sont pas dans le bon ordre, on les permute, on passe ensuite au 2ème ( $T[2]$ )et 3ème ( $T[3]$ ), puis 3ème et 4ème et ainsi de suite jusqu'au  $(n-1)$ ème ( $T[n-1]$ )et  $n$ ème éléments ( $T[n]$ ).
- A la fin du premier parcours, on aura poussé le plus grand élément du tableau vers sa place finale qui est le  $n$ ème élément du tableau.
- On recommence cette opération en parcourant de 1 à  $n-1$  puis de 1 à  $n-2$  et ainsi de suite.
- On arrête quand la partie à trier est réduite à un seul élément ou que le tableau est devenu trié (c.à.d aucune permutation n'a été faite lors du dernier parcours à vérifier par un indicateur)



**Remarque:** Dans le pire des cas le tri à bulles fait  $(n-1)+(n-2)+\dots+2+1$  comparaisons et autant de permutations.

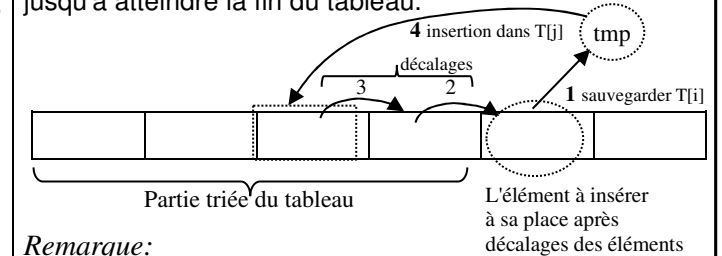
##### → Le tri par insertion:

#### 3.1. Principe:

on suppose que les  $i-1$  premiers éléments sont triés

- On cherche la position de l' $i$ ème élément dans la partie du tableau commençant de 1 à  $i$ .
- Si cette position est  $i$ , l'élément est donc à sa bonne place.
- Sinon, supposant que cette position est  $j$ ; ce  $j$  est forcément entre 1 et  $i-1$ . → (1) on affecte  $T[i]$  à une variable auxiliaire  $tmp$ , (2) On décale d'un pas vers l'avant (à droite) tous les éléments de  $j$  à  $i-1$  (3) puis on insère l'élément d'indice  $i$  (précédemment sauvegardé dans  $tmp$ ) à la position  $j$ .

On commence ce procédé à partir du 2ème éléments  $i=2$  jusqu'à atteindre la fin du tableau.



#### Remarque:

Dans le pire des cas le tri par insertion fait  $1+2+\dots+n-1$  comparaisons et autant de décalages. → Il fait  $n(n-1)$  opérations (comparaisons et décalages confondus)

### Leçon 2 : Algorithmes de recherche d'un élément dans un tableau

**I-Introduction:** Recherche d'un élément dans un tableau par deux méthodes:

#### II-La recherche séquentielle:

→ **Définition:** La méthode de recherche séquentielle d'un élément dans un tableau consiste à parcourir le tableau élément par élément progressivement de début vers la fin en les comparant avec l'élément à chercher jusqu'à trouver ce dernier ou achever le tableau.

#### III-La recherche dichotomique:

→ **Définition:** La méthode de recherche dichotomique consiste à chercher un élément dans un tableau **trié**.

On compare l'élément à chercher à l'élément central du tableau, s'ils sont différents, un test permet de trouver dans quelle moitié du tableau on peut trouver l'élément.

On continue ce processus jusqu'à trouver l'élément ou bien arrive à un sous-tableau de taille 1.