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 à 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...

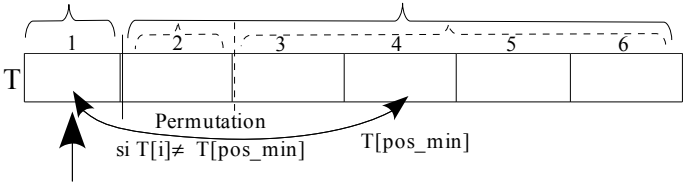
II-Les méthodes de tri:

On a choisit de trier les séquences par ordre croissant (de plus petit au plus grand) relativement à un ordre total noté ≤ .

➔ *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'il sont différents(T[1]≠ 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],..,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],..T[n]]).



N.B: La recherche du plus petit élément d'un tableau sera assurée 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 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

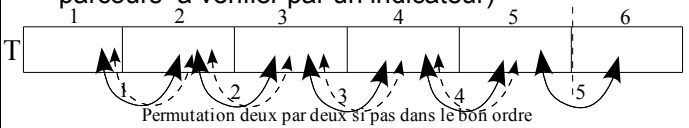
Donc, on fait (n-1)+n-2)+..+2+1= n(n-1)/2 comparaisons et au maximum n-1 permutations.

➔ *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 2éme 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)ième (T[n-1])et nième éléments (T[n]).
- À la fin du premier parcours, on aura pousser le plus grand élément du tableau vers sa place finale qui est le nième 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)+...+2+1 comparaisons et autant de permutations.

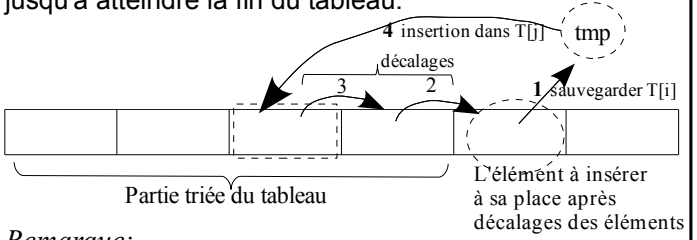
➔ *Le tri par insertion:*

3.1. Principe:

on suppose que les i-1 premiers éléments sont triés

- On chercher la position du ième 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 forcement 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+...+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'il son différents, un test permet de trouver dans quelle moitié du tableau on peut trouver l'élément.

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

Tri par sélection :

Analyse du P.P:

NOM : sélection
Résultat=PROC Afficher(T, n)
Proc Tri_Selction(T, n)
Proc Remplir(T,n)
FIN Sélection

Tableau de déclaration des nouveaux types :

Type
Tab= tableau de 9 entiers

Analyse de la procédure Remplir :

```
DEF Proc Remplir ( VAR T :TAB ; n :entier)
Résultat=  [ ] Pour i de 1 à n faire
                T[i]← hasard (101)
            FinPour
FIN Remplir
```

Analyse de la procédure Tri_Selection:

```
DEF Proc Tri_selection (VAR T :TAB ; n :entier)
Résultat=  [] Pour i de 1 à n-1 faire
          [pos_min←i] Pour j de i+1 à n faire
              [] Si ( T[j] < T[pos_min])Alors
                        pos_min ← j
                    Finsi
                FinPour
              [] Si ( pos_min < > i)Alors
                 Permute(T[i], T[Pos_min])
               Finsi
            FinPour
FIN Tri_Selection
```

Analyse de la procédure Permute :

```
DEF Proc Permute ( VAR A, B :entier)
Résultat= AUX ←A
          A ← B
          B←AUX
FIN Permute
```

Analyse de la procédure Afficher:

```
DEF Proc Affiche ( T :Tab ;n :entier)
Résultat= [] Pour i de 1 à n faire
       Ecrire(" l'élément n° ",i, "est  " ,T[i])
          FinPour
FIN Afficher
```

Tri à bulles :

Analyse du P.P:

NOM : Bulles
Résultat=PROC Afficher(T, n)
Proc Tri_Bulles(T, n)
Proc Remplir(T,n)
FIN Bulles

Tableau de déclaration des nouveaux types :

Type
Tab= tableau de 9 entiers

Analyse de la procédure Remplir :

```
DEF Proc Remplir ( VAR T :TAB ; n :entier)
Résultat= [ ]  Pour i de 1 à n faire
              T[i]← hasard (101)
           FinPour
FIN Remplir
```

Analyse de la procédure Tri_Bulles:

```
DEF Proc Tri_Bulles (VAR T :TAB ; n :entier)
Résultat=  [] Répéter
[ Echange← faux] Pour i de 1 à n-1 faire
                    [] Si ( T[i] > T[i+1])Alors
                          Permute(T[i], T[i+1])
                           Echange ← vrai
                       FinSi
                   FinPour
            n←n-1
        Jusqu'à (Echange = Faux) ou (n=1)
FIN Tri_Bulles
```

Analyse de la procédure Permute :

```
DEF Proc Permute ( VAR A, B :entier)
Résultat= AUX ←A
          A ← B
          B←AUX
FIN Permute
```

Analyse de la procédure Afficher:

```
DEF Proc Afficher ( T :Tab ;n :entier)
Résultat= Pour i de 1 à n faire
       Ecrire(" l'élément n° ",i, "est  " ,T[i])
          FinPour
FIN Afficher
```

Tri par insertion :

Analyse du P.P:

NOM : Insertion
Résultat=PROC Afficher(T, n)
Proc Tri_Insertion(T, n)
Proc Remplir(T,n)
FIN Insertion

Tableau de déclaration des nouveaux types :

Type
Tab= tableau de 9 entiers

T.D.O. Globaux :

Objet	Type/Nature
Affiche,tri_insertion, Remplir	Procédure
T	Tab
n	Constante=9

Analyse de la procédure Remplir :

```
DEF Proc Remplir ( VAR T :TAB ; n :entier)
Résultat= [ ]  Pour i de 1 à n faire
              T[i]← hasard (101)
           FinPour

FIN Remplir
```

Analyse de la procédure Tri_Insertion :

```
DEF Proc Tri_Insertion (VAR T :TAB ; n :entier)
Résultat= [ ] Pour i de 2 à n faire
  Tmp ← T[i]
      [ j ← i - 1]
     Tant que ( j >=1) et ( T[j] > Tmp ) faire
           T[j + 1]← T[j]
           j ← j - 1;
     FinTantQue
  T[j + 1] ←Tmp
            FinPour

FIN Tri_Insertion
```

Analyse de la procédure Afficher:

```
DEF Proc Afficher ( T :Tab ;n :entier)
Résultat= Pour i de 1 à n faire
       Ecrire(" l'élément n° ",i, "est  " ,T[i])
          FinPour

FIN Afficher
```

Tri par sélection :

```
program selection;
uses wincrt;
const n=9;
type tab=array[1..n]of integer;
var
t:tab;
procedure remplir(var T:tab;n:integer);
var i:integer;
begin
for i:=1 to n do
t[i]:=random(101);
end;
procedure permute (var A,B:integer);
var Aux:integer;
begin
aux:=A;
A:=B;
B:=aux;
end;

function premposmin(t:tab;i:integer;n:integer):integer;
var j,posmin:integer;
begin
posmin:=i;
for j:=i+1 to n do
if t[j]<t[posmin] then posmin:=j;
premposmin:=posmin;
end;
procedure tri_selction(var t:tab; n:integer);
var i,ppm:integer;
begin
for i:=1 to n-1 do
begin
ppm:=premposmin(t,i,n);
if i<>ppm then permute( t[i],t[ppm]);
end;
end;
procedure afficher(t:tab;n:integer);
var i:integer;
begin
for i:=1 to n do
writeln('l''element n° ',i,' est ',t[i]);
end;

begin
randomize;
remplir(t,n);
writeln('********avant le tri********');
afficher(t,n);
tri_selction(t,n);
writeln('********après le tri********');
afficher(t,n);
end.
```

Tri à bulles :

```
program bulles;
uses wincrt;
const n=9;
type tab=array[1..n]of integer;
var
t:tab;
procedure remplir(var T:tab;n:integer);
var i:integer;
begin
for i:=1 to n do
t[i]:=random(101);
end;

procedure permute (var A,B:integer);
var Aux:integer;
begin
aux:=A;
A:=B;
B:=aux;
end;
procedure tri_bulles(var t:tab; n:integer);
var i:integer;echange :boolean ;
begin

repeat
echange :=false ;
     for i :=1 to n-1 do
      begin
         if ( T[i] > T[i+1]) then
                              begin
                    Permute(T[i], T[i+1]) ;
                       Echange :=true;
                              End ;
      End ;   n :=n-1 ;
Until  (echange = false)or (n=1) ;
End ;
procedure afficher(t:tab;n:integer);
var i:integer;
begin
for i:=1 to n do
writeln('l''element n° ',i,' est ',t[i]);
end;

begin
randomize;
remplir(t,n);
writeln('********avant le tri********');
afficher(t,n);
tri_bulles(t,n);
writeln('********après le tri********');
afficher(t,n);
end.
```

Tri par insertion :

```
program insertion;
uses wincrt;
const n=9;
type
tab=array[1..n]of integer;
var
t:tab;

procedure remplir(var T:tab;n:integer);
var i:integer;
begin
for i:=1 to n do
t[i]:=random(101);
end;

Procedure Tri_Insertion(var t:tab;n: integer);
var i, j, tmp : integer;
begin
   for i:=2 to n do
   begin
       tmp := t[i];

       j := i - 1;
       while (j >= 1) and (t[j] > tmp) do
             begin
           t[j + 1] := t[j];
           j := j - 1;
             end;


       t[j + 1] := tmp;
   end;
end;

procedure afficher(t:tab;n:integer);
var i:integer;
begin
for i:=1 to n do
writeln('l''element n° ',i,' est ',t[i]);
end;

begin
randomize;
remplir(t,n);
writeln('********avant le tri********');
afficher(t,n);
tri_insertion(t,n);
writeln('********après le tri********');
afficher(t,n);

end.
```