Algorithme du tri par sélection d'un tableau

```
Pour i allant de 1 à N-1 faire
   indice_min ← i;
        Pour j ← i+1 à N faire      //Chercher la plus petite valeur dans le sous tableau de droite
             Si (T[j] < T[indice_min]) alors
                  indice_min ← j;
             Finsi
        Finpour j
             Si ( i <> indice_min) alors  //on a trouvé une valeur minimale ayant indice_min
                  Aide ← T[indice_min];
                  T[indice_min] ← T[i];
                  T[i] ← Aide;
             Finsi
Finpour i
```

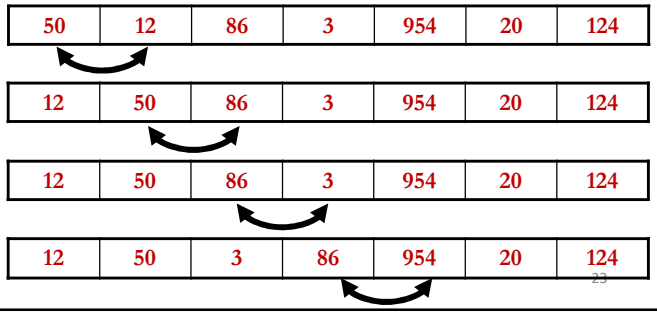

Tri à bulle d'un tableau

1. Parcourir tout le tableau en comparant successivement les éléments du tableau **deux à deux**.
2. Permuter les deux éléments comparés s'ils ne sont pas dans l'ordre.
3. Cette opération est répétée plusieurs fois jusqu'à ce que le tableau soit entièrement parcouru sans réaliser aucune permutation.

Soit le **tableau suivant** :

Premier passage :

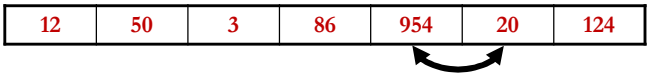
1. On se positionne dans la 1ere case :
   50>12 → On permute
2. On se positionne dans la 2ième case :
   50<86 → Pas de permutation
3. On se positionne dans la 3ième case :
   86>3 → On permute
3. On se positionne dans la 4ième case :
   86<954 → Pas de permutation



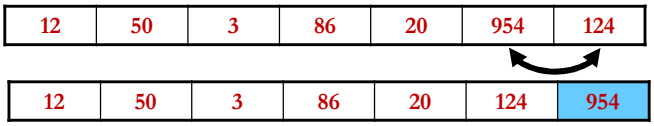
Tri à bulle d'un tableau

4. On se positionne dans la 4ième case :
   86<954 → Pas de permutation

5. On se positionne dans la 5ième case :
   954>20 → On permute



6. On se positionne dans la 6ième case :
   954>124 → On permute



Tri à bulle d'un tableau

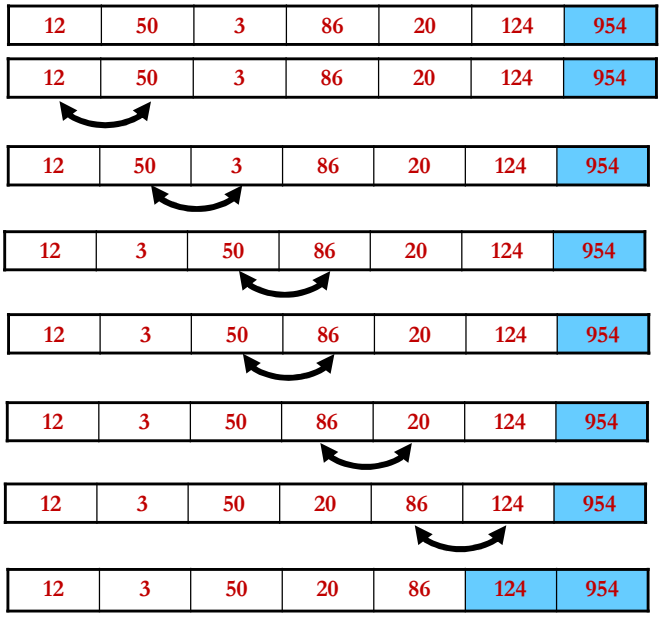
Remarques :

12	50	3	86	20	124	954

1. A l'issue de ce premier passage, on remarque que le **tableau n'est pas entièrement** trié, mais la **plus grande valeur** a été placée dans **la dernière case** du tableau (*colorée en bleu*).
2. Il faut **effectuer plusieurs passages** en vérifiant à chaque passage **si des permutations ont eu lieu.**
3. Quand une **permutation au moins a eu lieu** lors d'au passage, **il faut en relancer un autre**
4. Il faut **mettre en place un drapeau** indiquant si une permutation a eu lieu ou non

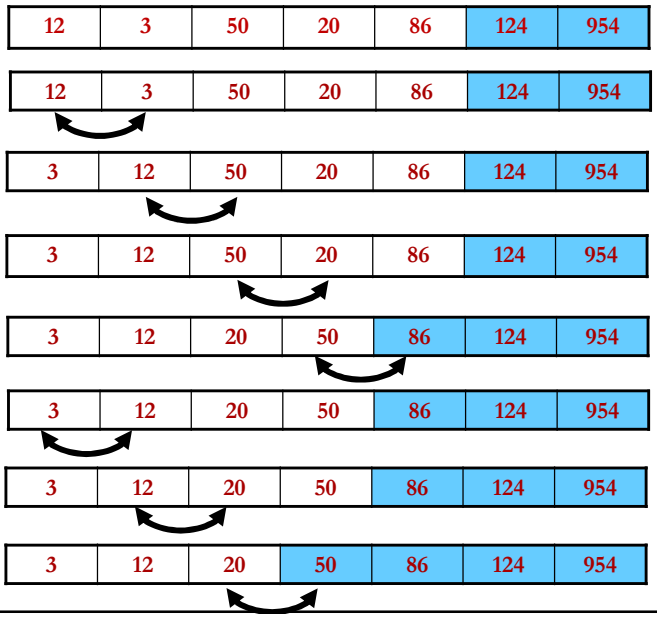
Tri à bulle d'un tableau

Deuxième passage :



Tri à bulle d'un tableau

Troisième passage :



On doit effectuer plusieurs passages … Jusqu'à ce que le tableau soit dans l'ordre (trié) : En parcourant tout le tableau sans faire aucune permutation.

Tri à bulle d'un tableau

```
terme_tri ← N-1;
Répéter
        ordre ← vrai;
        Pour i allant de 1 à terme_tri faire
                Si (T[i] > T[i+1]) alors
                        Aide ← T[i];
                        T[i] ← T[i+1];
                        T[i+1]← Aide;
                        ordre ← faux;
                Finsi
        Finpour
        Si (ordre = faux) alors
                terme_tri ← terme_tri -1;
        Finsi
Jusqu'à (ordre = vrai)
```


Tri par extraction

Cette méthode utilise en plus du tableau à trier un deuxième tableau dans lequel on place les éléments triés :

- On cherche le plus **petit élément min** dans le premier tableau T1 et on le place au début du deuxième tableau T2.
- Ensuite on cherche le **plus petit élément** parmi ceux non encore sélectionnée du T1 et on le place dans T2 jusqu'à ce que tous les éléments soient recopiés dans T2.
- A chaque fois qu'un élément est sélectionné, il est **remplacé par une valeur spéciale** pour ne pas être sélectionné une deuxième fois.

Exemple:

Soit un tableau T1 suivant contient **6 entiers (**notes d'étudiants, **valeur spéciale = 21)** :

T1	8	4	15	6	3	11
Tableau résultat T2						

Itération 1 :

T1	8	4	15	6	3	11
Tableau résultat T2	3					
T1 (après iteration 1)	8	4	15	6	21	11

Itération 2 :

T1	8	4	15	6	21	11
Tableau résultat T2	3	4				
T1 (après iteration 2)	8	21	15	6	21	11

Itération 3 :

T1	8	21	15	6	21	11
Tableau résultat T2	3	4	6			
T1 (après iteration 3)	8	21	15	21	21	11

Itération 4 :

T1	8	21	15	21	21	11
Tableau résultat T2	3	4	6	8		
T1 (après iteration 4)	21	21	15	21	21	11

Itération 5 :

T1	21	21	15	21	21	11
Tableau résultat T2	3	4	6	8	11	
T1 (après iteration 5)	21	21	15	21	21	21

Itération 6 :

T1	21	21	15	21	21	21
Tableau résultat T2	3	4	6	8	11	15
T1 (après iteration 6)	21	21	21	21	21	21

- Tous les éléments de T1 sont remplacés par une valeur spéciale (21).
- Tous les éléments sont triés et recopiés dans le tableau résultat T2.

Tri par extraction

```
Algorithme triParExtraction;
  Var
      T1, T2 : tableau[1..100] de réel;
      i, j, max, N, ind : entier;
  Début
        Répéter
              Ecrire(''saisir la dimension N du tableau'');
              Lire(N);
        Jusqu'à (N>=0 ET N<=100);
        Si (N=0) alors
              Ecrire(''le tableau est vide'');
        Sinon
              // Saisie des éléments du tableau
              Pour i ← 1 à N faire
                    Ecrire("entrer l'élément T1 ", i); Lire(T1[i]);
              FinPour i
              // Saisie des éléments du tableau avant le tri
              Pour i ← 1 à N faire
                    Ecrire(T1[i]);
              FinPour i
```

```
        max ← 1000;
        Pour i ← 1 à N faire
              ind ← 1;
              Pour j ← 2 à N faire
                    Si(T1[ind]>T1[j]) alors
                          ind ← j;
                    Finsi
              FinPour j
                    T2[i] ← T1[ind];
                    T1[ind] ← max;
        FinPour i
        // affichage des éléments du tableau triés
        Pour i ← 1 à N faire
                Ecrire(T2[i]);
        FinPour i
  Finsi
  Fin
```

Recherche dichotomique dans un tableau

- **Recherche dichotomique :**

  Dans le cas où le tableau est ordonné, on peut améliorer l'efficacité de la recherche séquentielle en utilisant la méthode de recherche dichotomique

- **Principe :** diviser par 2 le nombre d'éléments dans lesquels on cherche la valeur x à chaque étape de la recherche. Pour cela on compare x avec T[milieu] :

  – Si x < T[milieu], il suffit de chercher x dans la 1ère moitié du tableau entre (T[0] et T[milieu-1])

  – Si x > T[milieu], il suffit de chercher x dans la 2ème moitié du tableau entre (T[milieu+1] et T[N-1])

Recherche dichotomique dans un tableau

```
inf ← 0 , sup ← N-1, Trouvé ← Faux
TantQue ((inf <=sup)  ET (Trouvé=Faux)) faire
    milieu←(inf+sup)/2
        Si (x=T[milieu]) alors
                Trouvé ← Vrai
        Sinon Si (x>T[milieu]) alors
                inf ← milieu + 1
                Sinon
                        sup ← milieu - 1
                FinSi
        FinSi
FinTantQue
Si (Trouvé) alors
    écrire ("x appartient au tableau")
Sinon
    écrire ("x n'appartient pas au tableau")
FinSi
```