Dans ce chapitre, on désire trier éléments d'un tableau donné dans l'ordre croissant.

1 Tri par sélection

16 fin

C'est la méthode de tri la plus intuitive.

Le tableau à trier est « divisé » en deux parties : la $1^{\grave{e}re}$ constituée des éléments triés (initialisée avec seulement le 1^{er} élément) et la seconde constituée des éléments non triés (initialisée du $2^{\grave{e}me}$ au dernier élément)

- Le premier élément constitue, à lui tout seul, un tableau trié de longueur 1.
- On recherche le plus petit élément dans la partie non triée du tableau et on l'échange avec le dernier élément de la partie triée du tableau. À la première étape, le plus petit élément du tableau est donc mis au début. On obtient alors un sous-tableau trié de longueur 1.
- On augmente de 1 la taille du sous-tableau trié, en y incluant le deuxième élément du tableau. On recherche le plus petit élément dans la partie non triée du tableau, en commençant par le troisième élément, et on l'échange avec le dernier élément de la partie triée du tableau. On obtient alors un sous-tableau trié de longueur 2. Et ainsi de suite...
- Le principe du tri par sélection est donc d'échanger à la n^{ieme} itération le dernier élément de la partie triée du tableau avec le plus petit élément de la partie non triée du tableau.

```
→ ALGORITHME DU TRI PAR SÉLECTION
 1 TriSelection (S : Tab)
    Entrée :
        S: tableau non trié d'entiers
    Sorties:
        S : tableau trié
    Variables locales:
        i : entier - compteur pour boucle
       i: entier - compteur pour boucle
        indice : entier - indice de l'élément le plus petit
 2 début
        pour i=1 à S.Fin-1 faire
 3
            - initialisation de l'indice de l'élément le plus petit avec l'indice du dernier élément de la
             partie du tableau trié
            indice \leftarrow i;
  5
            - recherche de l'élément le plus petit de la partie du tableau non triée
            pour j=i+1 à S.Fin faire
                si S[j] <= S[indice] alors
  8
                    - Mise à jour de l'indice de l'élément le plus petit de la partie du tableau non trié
                    indice \leftarrow j;
 10
                fin
 11
            fin
 12
            - permutation de cet élément le plus petit avec le 1 er élément de la partie du tableau non triée
 13
             qui devient le dernier élément de la partie du tableau trié.
            S[i] \leftrightarrow S[indice]
 14
        fin
```

\hookrightarrow Un exemple pour illustrer

Tableau non trié : 10 16 3 21 6 9	12	2
Pointage de l'élément courant et Recherche de l'élément le plus petit : 10 16 3 21 6 9	12	2
Permutation de l'élément courant et de l'élément le plus petit : 16 3 21 6 9	12	10
Pointage de l'élément courant et Recherche de l'élément le plus petit : 2 16 3 21 6 9	12	10
Permutation de l'élément courant et de l'élément le plus petit : 16 21 6 9	12	10
Pointage de l'élément courant et Recherche de l'élément le plus petit : 16 21 6 9	12	10
Permutation de l'élément courant et de l'élément le plus petit :	12	10
Pointage de l'élément courant et Recherche de l'élément le plus petit : 2 2 1 16 9	12	10
Permutation de l'élément courant et de l'élément le plus petit :	12	10
On renouvelle les deux étapes précédentes :	12	16
On renouvelle les deux étapes précédentes :	21	16
On renouvelle les deux étapes précédentes :	16	21
Aucune étape supplémentaire - le tableau est trié :	16	21
Légende : élement courant : 1er élément du tableau non trié élément le plus petit tableau trié		

 $\hookrightarrow Un \ lien \ pour \ voir \ fonctionner \ le \ processus : \ \texttt{http://lwh.free.fr/pages/algo/tri/tri_selection.html}$

\hookrightarrow La complexité de l'algorithme

- 1. Combien de comparaisons et d'échanges vont être effectués dans tous les cas?
- 2. Si le tableau contient 20 millions de valeurs, combien y aura-t-il de comparaisons et d'échanges dans tous les cas?

ADMIS

La complexité de la méthode de tri par sélection d'un tableau contenant n éléments est $O(n^2)$.

2 Tri par insertion

Le tri par insertion le tri « naturel » du joueur de cartes.

Le tableau à trier est « divisé » en deux parties : la $1^{\grave{e}re}$ constituée des éléments triés (initialisée avec seulement le 1^{er} élément du tableau) et la seconde partie constituée des éléments non triés du tableau (initialisée du $2^{\grave{e}me}$ au dernier élément). On procède comme si les éléments d'un tableau à trier étaient donnés un par un.

- Le premier élément constitue, à lui tout seul, un tableau trié de longueur 1.
- On range ensuite le second élément « à sa place » pour constituer un tableau trié de longueur 2, puis on range le troisième élément pour avoir une tableau trié de longueur 3 et ainsi de suite...
- Le principe du tri par insertion est donc d'insérer à la n^{ieme} itération le n^{ieme} élément à la « bonne » place.

\hookrightarrow Algorithme du tri par insertion

```
1 TriInsertion (S : Tab)
   Entrée:
       S: tableau non trié d'entiers
  Sorties:
       S: tableau trié
   Variables locales:
       i: entier - compteur pour boucle
       j: entier - compteur pour boucle
       valeur : entier - valeur de l'élément à déplacer par insertion
       indice : entier - indice futur de l'élément à déplacer par insertion
2 début
       - le tableau est constitué de deux parties : la 1ère constituée des éléments triés
        (initialisée avec seulement le 1^{er} élément du tableau) et la seconde partie constituée
        des éléments non triés du tableau (initialisée du 2ème au dernier élément)
4
       pour i=1 à S.Fin-1 faire
           - mémorisation du 1^{er} élément de la partie du tableau non trié que nous allons
5
            déplacer
6
           valeur \leftarrow S[i+1]
           - recherche de l'indice que doit prendre ce 1er élément dans la partie du tableau trié
7
           indice \leftarrow 1
           tant que S/indice]<valeur faire
               indice \leftarrow indice + 1
10
           fin
11
           - décalage des éléments compris entre le dernier élément de la partie du tableau trié
12
            et l'emplacement trouvé précédemment (parcours décroissant)
           pour j=i à indice en décroissant faire
13
              S[j+1] \leftarrow S[j]
14
           fin
15
           - Déplacement par insertion du 1^{er} élément de la partie du tableau non trié à l'indice
16
            trouvé ... qui devient un élément trié
           S[indice] \leftarrow valeur
17
       fin
18
19 fin
```

\hookrightarrow Un exemple pour illustrer

Tableau non trié : 10 16 3 21 6 9 12 2
Intialisation du tableau au seul 1er élément et Pointage de l'élément courant : 16 3 21 6 9 12 2
Insertion de l'élément au bon endroit courant dans la partie triée : 16 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Pointage de l'élément courant : 15 15 16 9 12 2
Insertion de l'élément au bon endroit courant dans la partie triée :
Pointage de l'élément courant : 10 10 16 9 12 2
Insertion de l'élément au bon endroit courant dans la partie triée :
Pointage de l'élément courant : 3 18 18 18 6 9 12 2
Insertion de l'élément au bon endroit courant dans la partie triée :
On renouvelle les deux étapes précédentes : 3 5 9 10 16 21 12 2
On renouvelle les deux étapes précédentes : 18 18 18 18 18 18 20 2
On renouvelle les deux étapes précédentes : 12 / 32 / 16 / 34 / 16 / 23
le tableau est trié : 12 / 13 / 16 / 14 / 16 / 12 / 12

Légende : élément courant : 1er élément du tableau non trié tableau trié

 \hookrightarrow Un lien pour voir fonctionner le processus : http://lwh.free.fr/pages/algo/tri/tri_insertion.html

\hookrightarrow La complexité de l'algorithme

- 1. Combien de comparaisons et d'échanges vont être effectués dans le meilleur cas?
- 2. Combien de comparaisons et d'échanges vont être effectués dans le pire cas?
- 3. Si le tableau contient 20 millions de valeurs, combien y aura-t-il de comparaisons et d'échanges dans le pire cas ?

ADMIS

La complexité de la méthode de tri par insertion d'un tableau contenant n éléments est $O(n^2)$.