1ère NSI - Algorithmes de tri : le cours

Dans ce chapitre, on désire trier dans l'ordre croissant les n éléments d'un tableau S. Par convention, les indices des éléments sont numérotés de 0 à $S.Fin$ avec $S.fin = n - 1$.

1 Tri par sélection

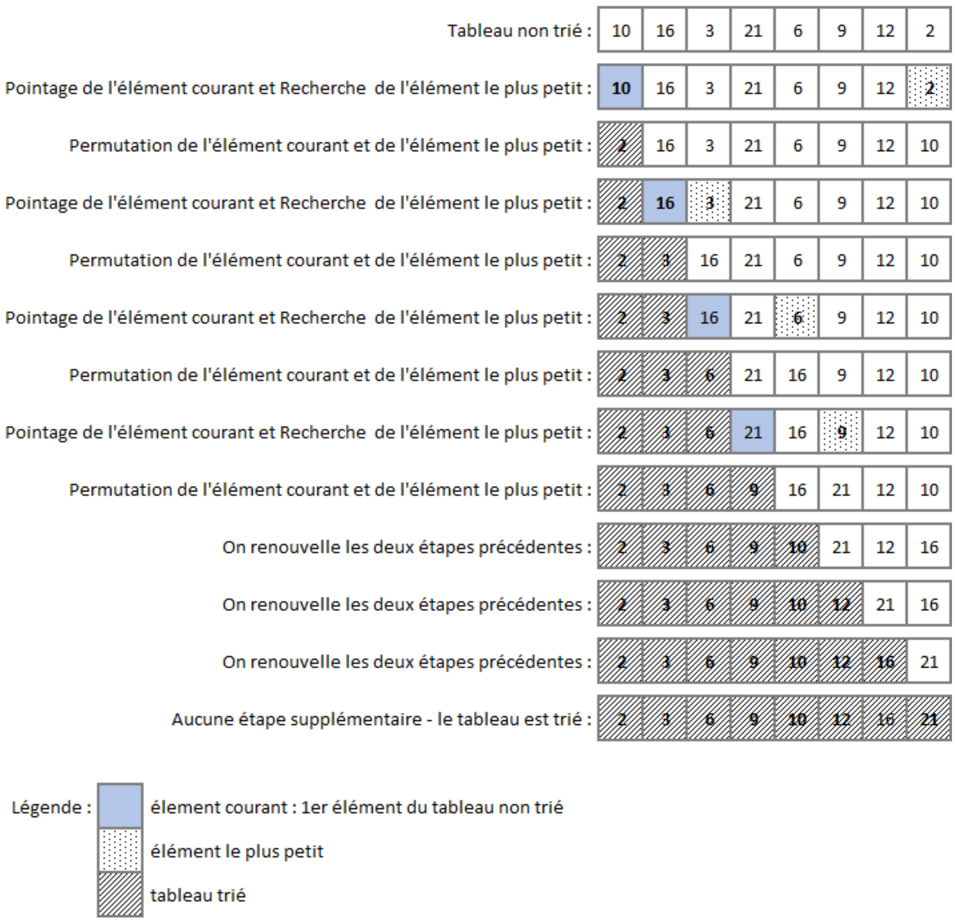
C'est la méthode de tri la plus intuitive.
Le tableau à trier est « divisé » en deux parties : la 1ère constituée des éléments triés (initialisée vide) et la seconde constituée des éléments non triés (initialisée du 1er au dernier élément)

— 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 (indice 0). 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

```
TriSelection (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
      indice : entier - indice de l'élément le plus petit
début
    // le tableau est constitué de deux parties : la 1ère
       partie à gauche constituée des éléments triés
       (initialisée vide) et la seconde partie à droite
       constituée des éléments non triés (initialisée du 1er au
       dernier élément)
    pour i=0 à S.Fin-1 faire
        // initialisation de l'indice de l'élément le plus petit
           avec l'indice du premier élément de la partie du
           tableau non trié
        indice ← i;
        // 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
                // Mise à jour de l'indice de l'élément le plus
                   petit de la partie du tableau non trié
                indice ← j;
            fin
        fin
        // permutation de cet élément le plus petit avec le 1er
           élément de la partie du tableau non triée qui devient
           le dernier élément de la partie du tableau trié
        S[i] ↔ S[indice];
    fin
fin
```

↪ Un exemple pour illustrer



↪ Un lien pour voir fonctionner le processus : `http://lwh.free.fr/pages/algo/tri/tri_selection.html`

↪ La complexité de l'algorithme

La complexité de l'algorithme de tri par sélection se mesure en comptant le nombre de comparaisons et d'échanges. Elle n'a réellement de signification que pour des tableaux de très grande taille.

— Dans tous les cas, pour trier n éléments d'un tableau, le tri par sélection effectue $\dfrac{n(n-1)}{2}$ comparaisons : $(n-1)$ comparaisons pour rechercher le plus plus élément à la première étape, puis $(n-2)$ comparaisons à la deuxième, ..., puis 1 à la dernière étape.
— Dans le pire cas (tableau rangé dans l'ordre décroissant), le nombre d'échanges est $(n-1)$. Dans le meilleur cas (tableau déjà trié), aucun échange n'est effectué. On dit que le nombre d'échanges est *linéaire*.

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 $1^{ère}$ 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 $2^{ème}$ 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.

↪ ALGORITHME DU TRI PAR INSERTION

```
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
début
    // le tableau est constitué de deux parties : la 1ère
       partie constituée des éléments triés (initialisée avec
       seulement le 1er élément) et la seconde partie
       constituée des éléments non triés (initialisée du 2ème
       au dernier élément)
    pour i=0 à S.Fin-1 faire
        // mémorisation du 1er élément de la partie du tableau
           non trié que nous allons déplacer
        valeur ← S[i + 1];
        // recherche de l'indice de la place que doit prendre ce
           1er élément dans la partie du tableau trié
        indice ← 0;
        tant que S[indice]<valeur faire
            indice ← indice + 1;
        fin
        // décalage des éléments compris entre le dernier
           élément de la partie du tableau trié et l'emplacement
           trouvé précédemment (parcours décroissant)
        pour j=i à indice en décroissant faire
            S[j + 1] ← S[j];
        fin
        // Déplacement (insertion) du 1er élément de la partie du
           tableau non trié à l'indice trouvé ... qui devient un
           élément trié
        S[indice] ← valeur;
    fin
fin
```

Une version améliorée de l'algorithme est donnée ci-dessous :

```
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
début
    pour i=1 à S.Fin faire
        // mémorisation du 1^er élément de la partie du tableau
           non trié que nous allons déplacer
        valeur ← S[i];
        // initialisation de l'indice futur de l'élément à
           déplacer par insertion
        indice ← i;
        // décalage des éléments compris entre le dernier
           élément de la partie du tableau triée et
           l'emplacement trouvé précédemment (parcours
           décroissant)
        tant que indice>=1 et valeur<=S[indice-1] faire
            S[indice] ← S[indice − 1];
            indice ← indice − 1;
        fin
        // Déplacement (insertion) du 1^er élément de la partie du
           tableau non triée à l'indice trouvé ... qui devient
           un élément trié
        S[indice] ← valeur;
    fin
fin
```

↪ 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 :	10	16	3	21	6	9	12	2
Insertion de l'élément au bon endroit courant dans la partie triée :	10	16	3	21	6	9	12	2
Pointage de l'élément courant :	10	16	3	21	6	9	12	2
Insertion de l'élément au bon endroit courant dans la partie triée :	3	10	16	21	6	9	12	2
Pointage de l'élément courant :	3	10	16	21	6	9	12	2
Insertion de l'élément au bon endroit courant dans la partie triée :	3	10	16	21	6	9	12	2
Pointage de l'élément courant :	3	10	16	21	6	9	12	2
Insertion de l'élément au bon endroit courant dans la partie triée :	3	6	10	16	21	9	12	2
On renouvelle les deux étapes précédentes :	3	6	9	10	16	21	12	2
On renouvelle les deux étapes précédentes :	3	6	9	10	12	16	21	2
On renouvelle les deux étapes précédentes :	2	3	6	9	10	12	16	21
le tableau est trié :	2	3	6	9	10	12	16	21

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

tableau trié

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

↪ La complexité de l'algorithme

La complexité de l'algorithme de tri par insertion se mesure en comptant le nombre de comparaisons et d'affectations. Elle n'a réellement de signification que pour des tableaux de très grande taille.

— Dans le pire cas (tableau trié dans l'ordre décroissant), on va effectuer $\frac{n^2}{2}$ comparaisons et affectations : 1 comparaison et 2 affectations à la première étape, puis 2 et 3, puis 3 et 4, ..., puis $(n-1)$ et n pour finir.

— Dans le meilleur cas (tableau trié), on effectue n comparaisons et n affectations.

Admis

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