On programme, modifie et teste les algorithmes de tri vus dans le chapitre 5. On en déduit une fonction de calcul de la médiane, et l'on voit un autre exemple de tri par balayage : le tri par sélection.

1 Le tri à bulle

1.1 Principe

C'est l'algorithme de tri le plus simple :

- Parcourir les éléments du tableau de gauche à droite.
  - Dès que l'on rencontre deux éléments consécutifs qui ne sont pas dans le bon ordre, on échange leur position. C'est à dire :
    `SI tableau[i] > tableau[i+1]`
    `ALORS : échanger tableau[i] et tableau[i+1]`
- Recommencer tant que l'on a changé quelque chose.

Exemple :

Tableau :	3	2	1	5	4
Parcours 1	**3**	**2**	1	5	4
	2	**3**	1	5	4
	2	**3**	**1**	5	4
	2	**1**	**3**	5	4
	2	1	3	**5**	**4**
	2	1	3	**4**	**5**
Parcours 2	**2**	**1**	3	4	5
	1	**2**	3	4	5
Parcours 3	1	2	3	4	5
	1	2	3	4	5
Tableau trié :	1	2	3	4	5

1.2 Algorithme

Le principe illustré ci-dessus conduit à l'algorithme général suivant pour le tri d'un tableau unidimensionnel (une liste en `python`) :

Algorithme du Tri à bulle (Paramètre : un tableau T)

```
    N = longueur(T)
    Changement = VRAI
    TANT QUE Changement est VRAI :
        Changement = FAUX
        POUR  i variant de 0 à N-2
            SI T[i] > T[i+1]
            ALORS  échanger T[i] et T[i+1]
                   Changement = VRAI
        FIN POUR
        N = N-1
    FIN TANT QUE
```

Exercice 1.

1. Écrire en Python, une fonction `tri_Bulle(T)` prenant en paramètre une liste `T`, et qui trie `T` dans le sens croissant par un tri à bulle.
2. Modifier la fonction pour que le tri se fasse dans l'ordre décroissant.

2 Tri par insertion

2.1 Principe

On constitue (imaginairement) 2 tas :

- l'un dans la main droite, contenant toutes les cartes avant tri,
- l'autre dans la main gauche, contenant les cartes déjà triées,

Initialement la main gauche est vide.

Chaque étape consiste à :

- prendre la première carte du tas non trié
- L'insérer progressivement à sa bonne place dans le tas trié, en la faisant descendre d'une position tant que sa valeur reste inférieure à celle de la carte située en dessous-d'elle.

Après chaque étape le tas non trié contient une carte de moins, le tas trié une carte de plus.
A la fin du tri la main droite est vide. La main gauche contient toutes les cartes, triées.

- **Exemple.** :
  - En noir : Partie du tableau, non triée,
  - En **gras** : Partie du tableau triée,
  - En surligné : Elément à insérer dans la partie triée.

3	2	1	5	6	4	Tableau à trier
3	2	1	5	6	4	Premier élément
3	2	1	5	6	4	Deuxième élément
2	**3**	1	5	6	4	Inséré dans le tableau trié
2	**3**	1	5	6	4	Troisième élément
2	1	**3**	5	6	4	Inséré dans le tableau trié
1	**2**	**3**	5	6	4	⋮
1	**2**	**3**	5	6	4	Quatrième élément
1	**2**	**3**	**5**	6	4	Inséré dans le tableau trié
1	**2**	**3**	**5**	6	4	Cinquième élément
1	**2**	**3**	**5**	**6**	4	Inséré dans le tableau trié
1	**2**	**3**	**5**	**6**	4	Dernier élément
1	**2**	**3**	**5**	4	**6**	Inséré dans le tableau trié
1	**2**	**3**	4	**5**	**6**	⋮
1	**2**	**3**	**4**	**5**	**6**	Tableau trié.

2.2 Algorithme

En pseudo-code, l'algorithme de Tri par insertion s'écrit :

Algorithme de Tri par insertion (Paramètre : un tableau T)

```
    N = longueur(T)    # N = longueur du tableau
    # Balayage du tableau du 2ème jusqu'au dernier élément :
    POUR i variant de 1 à N-1:
        x = T[i]      # C'est l'élément à insérer
        k = i         # k est son indice
        # Insertion dans la partie triée :
        TANT QUE k > 0 et T[k-1] > x:
            # Décalage d'un cran sur la gauche :
            T[k] = T[k-1]
            k = k-1
        T[k] = x
```

Exercice 2.

1. Écrire en Python, une fonction `tri_Insertion(T)` prenant en paramètre une liste `T`, et qui trie `T` dans le sens croissant par un tri par insertion.
2. Modifier l'algorithme pour que le tri se fasse dans l'ordre décroissant.

3 Tri par sélection

Exercice 3.

La recherche de la valeur maximale dans une liste numérique est une méthode très utile qui est à maitriser. Le principe n'est pas difficiile :
ON déclare une variable `m` à laquelle on affecte la première valeur dans la liste. On parcourt au sein d'une boucle for les éléments suivants de la liste, en les comparant à la valeur stockée dans `m`. Dès que l'élement est plus grand que `m`, on l'affecte à la variable `m`. A la fin de la boucle, la variable `m` contiendra le plus grand élément de la liste.

- Écrire une fonction `maximum(L)` prenant en argument une liste numérique `L`, et qui renvoie le plus grand élément dans la liste `L`.
- Écrire une fonction `Indicemaximum(L)` prenant en argument une liste numérique `L`, et qui renvoie l'indice où se trouve le plus grand élément dans la liste `L`.

Exercice 4.

Il existe un autre algorithme célèbre de tri d'un tableau par balayage, le *tri par sélection*. Pour un tableau de N éléments, il consiste à :

- Sélectionner la valeur maximale dans le tableau
- Le déplacer en dernière position du tableau.
- Recommencer avec le tableau des $N-1$ premiers éléments, et ainsi de suite.

1. Ecrire une fonction `indiceMax(T,i)` prenant en paramètre un tableau numérique `T` et un entier `i` et qui retourne l'indice du plus grand élément de `T` d'indice au plus `i`.
2. L'utiliser pour écrire une fonction `tri_selection()` qui applique l'algorithme de tri par sélection.

4 Tri comptage

Lorsque on est en présence d'une liste, où la valeur des éléments est connue a priori, et en petit nombre, l'ordonner peut s'exécuter plus rapidement par un tri comptage. Le principe est le suivant :
Considérons une liste ne contenant que des 0 et des 1. On la parcourt à l'aide d'une boucle for, pour stocker dans deux variables `nb0` et `nb1`, le nombre de 0 et de 1 de la liste. On crée ensuite une nouvelle liste constituée de `nb0` zéros suivies de `nb1` uns ; que l'on renvoie.

Exercice 5.

Ecrire une fonction `triComptage(L)` prenant en argument une liste `L` constituée de zéros et de uns, et qui renvoie la liste ordonnée contenant les mêmes éléments.
On pourra l'essayer avec la liste créée de la façon suivante :

```
from random import randint
n = 20
L = []
for k in range(n):
    L.append(randint(0,1))
```

5 Application : calcul de médiane

La médiane d'une liste numérique est une valeur numérique à laquelle la moitié des éléments du tableau sont supérieurs ou égaux et l'autre moitié inférieurs ou égaux. Elle sépare le tableau en deux parties de même taille : les éléments inférieurs et ceux supérieurs à la médiane.

Le calcul d'une médiane est plus compliqué que celui de la moyenne. Pour l'obtenir on peut trier le tableau et dans le tableau trié :

- Si le tableau contient un nombre impair d'éléments, prendre l'élément au milieu du tableau trié.
- Si le tableau contient un nombre pair d'éléments, par exemple $2p$, prendre n'importe quelle valeur entre le p-ème et le $(p+1)$-ème élément du tableau trié. En pratique on prendra la moyenne de ces deux valeurs.

Exercice 6.

Écrire une fonction `mediane(T)` qui renvoie la médiane du tableau numérique `T`.
Elle utilisera l'une des fonctions de tri écrites précédemment.
Le tableau `T` ne devra pas être modifié.