TRAVAUX PRATIQUES 5

TRIS QUADRATIQUES

Voici les instructions éventuellement utiles dans ce TP :

`len(L)`	donne la longueur de `L`
`L=[]`	`L` est la liste vide
`L[i]`	donne la valeur de l'élément de la liste d'indice `i`
`L+M`	concatène (juxtapose) les listes `L` et `M`
`L.append(a)`	ajoute `a` à la fin de la liste
`L.insert(i,a)`	ajoute l'élément `a` à la i^e position
`del L[i]`	supprime et retourne le i^e élément
`for k in range(i,j)`	boucle **pour** `k` allant **de** `i` (inclus) **à** `j` (exclus)
`while`	boucle **tant que**
`randrange(a,b)`	renvoie un nombre entier `n` aléatoire tel que $a \leqslant n < b$
`randint(a,b)`	renvoie un nombre entier `n` aléatoire tel que $a \leqslant n \leqslant b$

1. **Tri par sélection**

   Pour trier un tableau `T` comportant n éléments par **sélection**, voici la méthode :

   En partant de la position `i` $= 0$, on recherche le plus petit élément parmi les éléments d'indice `i` $+1$ à `n` -1 et on l'échange avec `T[i]`.

   Par exemple, pour trier $[12,3,17,9,4,16]$, on obtient successivement :

   $$[12,3,17,9,4,16] \quad [12,|\boxed{3}|,17,9,4,16] \quad [3,12,|17,9,|\boxed{4}|,16] \quad [3,4,17,|\boxed{9}|,12,16]$$

   $$[3,4,9,17,|\boxed{12}|,16] \quad [3,4,9,12,|17,|\boxed{16}|] \quad [[3,4,9,12,17,|\boxed{16}|] \quad [3,4,9,12,16,17]$$

   Compléter la fonction suivante pour qu'elle retourne le tableau trié par sélection :

```
def tri_selection(T):
    for i in range(len(T)-1):
    ...
        for j in range (i+1,len(T)):
            # recherche du minimum dans le tableau restant
            # comparer ce minimum à T[i]
    return T
```

2. **Tri par insertion**

   Pour trier un tableau `T` par **insertion**, voici la méthode :

   On prend le premier élément et on le met à l'indice `i` $= 0$; puis on insère les autres éléments dans la partie déjà triée en plaçant chaque nouvel élément à la bonne place.

   Cela donne :

   $$[12,|3,17,9,4,16] \quad [3,12,|17,9,4,16] \quad [3,12,17,|9,4,16]$$

   $$[3,9,12,17,|4,16] \quad [3,4,9,12,17,|16] \quad [3,4,9,12,16,17]$$

   Ecrire une fonction python qui permette de trier le tableau par insertion.

   Evaluer la complexité temporelle des deux algorithmes de tri.

3. **Tri à bulles**

Soit T un tableau de valeurs, décrire l'évolution de la variable T au cours de l'algorithme suivant :

```
def tri_a_bulles(T):
    for i in range(len(T)-1):
        for j in range(len(T)-1,i,-1):
            if T[j]<T[j-1]:
                T[j],T[j-1]=T[j-1],T[j]
    return T
```