# TP 6 - Tris

Le but de ce TP est de découvrir et de manipuler des algorithmes de tris classiques.

Trier une liste signifie remettre ses différents éléments dans l'ordre (croissant à priori, on peut aussi trier par ordre décroissant), cela implique bien sûr que l'on puisse ordonner les différents éléments de la liste, on utilisera ici des éléments qui sont des nombres entiers.

**Important :** Pour tester les tris, il sera utile de pouvoir générer des listes désordonnées d'entiers de manière aléatoire.

- On commence par importer la fonction `randint` du module `random` avec la commande `from random import randint`. L'appel `randint(a,b)` retourne un entier aléatoire compris entre `a` et `b` (inclus).
- On construit une liste aléatoire d'entiers compris entre `a` et `b` de taille `n` fixée par compréhension avec la commande `L = [randint(a,b) for i in range(n)]`.

## I. Un premier tri

Dans le fichier *TP-Tris - script.py*, on dispose du code de la fonction `triBulle` ci-dessous implémentant le tri Bulle d'une liste.

```
1  def triBulle(L) :
2      n=len(L)
3      Done=False
4      while not Done and n>=2 :
5          Done=True
6          for i in range(n-1) :
7              if L[i]>L[i+1] :
8                  L[i],L[i+1]=L[i+1],L[i]
9                  Done=False
10         n-=1
```

Les questions 1 à 3 sont à traiter sur feuille.

**Question 1.** Dans cette question, on ne considère que la boucle inconditionnelle `for` (lignes 6 à 9). En prenant, comme exemple la liste `L=[2,1,4,3]` avec `n=4`, donner le contenu de la liste `L` à la fin de la première itération de la boucle `for` puis à la fin de la deuxième.

**Question 2.**

1. Que se passe-t-il au niveau de la variable `Done` lorsqu'une permutation de deux éléments, au moins, est effectuée ?
2. Que peut-on affirmer si aucune permutation n'a été effectuée dans la boucle `for` ?
3. En déduire le rôle de cette variable et le fonctionnement global de l'algorithme en commentant le code. Utiliser une liste simple pour vous aider.

**Question 3.** Pourquoi n'est-il pas nécessaire d'utiliser `return` ?

**Question 4.** En produisant une liste aléatoire de 30 éléments avec des valeurs comprises entre 0 et 50, tester l'algorithme `triBulle` dans pyzo.

Nous allons mesurer le temps que met cet algorithme à trier une liste pour différentes tailles de liste. Pour ce faire, nous utilisons le code suivant (disponible dans *TP-Tris - script*).

```
1  from matplotlib import pyplot as plt
2  from time import perf_counter
3  tailles=[50*i for i in range(20)]
4  def mesures() :
5      temps=[]
6      for longueur in tailles :
7          L=[randint(0,1000) for i in range(longueur)]
8          temps_initial=perf_counter()
9          triBulle(L)
10         temps.append(perf_counter()-temps_initial)
11     plt.plot(tailles,temps) #Tracer la courbe des temps en fonction des tailles
12     plt.show() #Afficher le graphe
```

**Question 5.** Sélectionner les deux première lignes de ce code puis les exécuter en appuyant sur `<Alt + Entrée>`. Tester ensuite la fonction `perf_counter()` dans la console en l'appelant plusieurs fois. En déduire le rôle de cette fonction.

**Question 6.** Analyser ce que réalise le script proposé en le commentant.

**Question 7.** Exécuter le script.

**Question 8.** Modifier la liste `tailles` avec `tailles=[500*i for i in range(10)]` puis ré-exécuter le script. Que peut-on dire du temps d'exécution du tri Bulle par rapport à la taille de la liste ?

# II. Tri par insertion

Dans cette deuxième partie, on s'intéresse à l'algorithme du tri par insertion. Le principe du tri par insertion consiste à considérer seulement le premier élément qui est, de fait, bien placé. On considère ensuite le deuxième élément qu'il faut bien placer pour que les deux premiers éléments soient triés. De manière générale, à partir de l'élément d'indice $i$, on considère que les éléments dont les indices sont compris entre 0 et $i-1$ inclus sont triés et on cherche à bien placer l'élément d'indice $i$ pour que le nouvel ensemble, constitué des éléments d'indice compris entre 0 et $i$ inclus, soit trié.

**Question 9.** On considère une liste d'entiers naturels de taille $n$ dont les $n-1$ premiers éléments sont triés dans l'ordre croissant, le dernier élément ayant une valeur quelconque, par exemple `[2,3,6,8,4]`. Ecrire le code d'une fonction `placer` prenant en paramètres une liste telle que celle décrite ci-dessus et modifiant en place la liste pour qu'elle soit totalement triée dans l'ordre croissant. Le principe de cette fonction est de placer le dernier élément au bon endroit dans la liste (puisque les $n-1$ premiers éléments sont triés), en utilisant des permutations d'éléments. Par exemple, pour `L=[2,3,6,8,4]` la fonction `placer` modifie `L` telle que `L=[2,3,4,6,8]`.

**Question 10.** Modifier très légèrement la fonction `placer` pour qu'elle prenne comme paramètre supplémentaire l'indice `i` de l'élément à bien placer entre les indices `0` et `i-1`. On suppose, dans ce cas, que les éléments de la liste dont les indices sont inférieurs à `i` sont triés dans l'ordre croissant. Par exemple, l'appel `placer(L,4)` avec `L=[2,3,5,7,4,1,3]` modifie en place la liste `L` telle que `L=[2,3,4,5,7,1,3]`.

**Question 11.** En déduire le code de la fonction `triInsertion()` prenant en paramètre une liste d'entiers et modifiant en place la liste pour qu'elle soit triée selon la stratégie du tri par insertion.

**Question 12.** Comme pour la partie précédente, en réutilisant le script fourni, tracer l'allure du temps mis pas l'algorithme `triInsertion` pour trier une liste en fonction de la taille de cette liste. Que peut-on conclure par rapport au tri bulle ?

# III. Tri fusion

Pour finir, on s'intéresse au tri Fusion utilisant la fusion de listes ainsi que le principe du diviser pour régner.

**Question 13.** Ecrire le code de la fonction `fusionner` prenant en paramètres deux listes triées et retournant une nouvelle liste triée contenant les mêmes éléments que les listes d'entrée. Par exemple, avec les listes `L1=[2,4,6]` et `L2=[2,3,4,7]`, l'appel `fusionner(L1,L2)` renvoie la liste `[2,2,3,4,4,6,7]`.

Il est conseillé de commencer par travailler avec un exemple simple sur une feuille. (Penser à enregistrer votre TP avant !)

On donne ci-dessous une implémentation du tri Fusion.

```
1 def triFusion(L) :
2     if len(L)<2 :
3         return L
4     else :
5         m = len(L)//2
6         return fusionner(triFusion(L[:m]), triFusion(L[m:]))
```

**Question 14.** Quelle propriété particulière présente cette fonction ?

**Question 15.** Décrire le comportement de cette fonction à l'aide de la dernière ligne de code.

**Question 16.** Adapter le code du script permettant de tracer le temps mis par la fonction pour trier des listes de tailles différentes. Que remarque-t-on pour cette solution par rapport aux deux tris précédents ?