Introduction

Les exercices ci-dessous ont pour objectif d'approfondir vos connaissances sur différents algorithmes de tri en Python, ainsi que sur la notion de stabilité dans les tris. Un tri est dit **stable** s'il conserve l'ordre relatif des éléments égaux dans la liste d'origine.

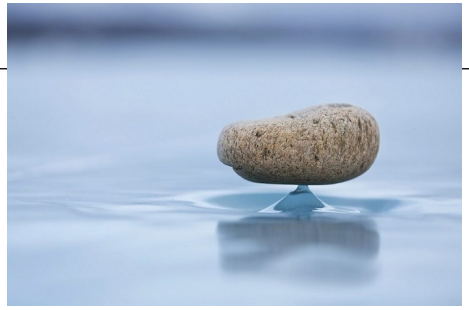

FIGURE 1 – Pierre au dessus d'un pilier de glace, lac Baïkal

.

I. Tri à bulles

Définition :

Soit une liste L de n éléments. Le tri à bulles consiste à comparer les deux premiers termes de la liste. S'ils sont désordonnés, on les échange. On recommence ensuite avec le deuxième et le troisième terme du tableau et ainsi de suite. Ainsi, au bout d'un premier passage, l'élément le plus grand est bien placé : il est remonté comme une bulle. On recommence l'opération jusqu'à ce que le tableau soit trié.

1 - Prise en main

On peut proposer l'algorithme du tri à bulles selon le pseudo code suivant issu de *The Art of Computer Programming, Knuth, 1997* :

```
1 tri_bulle(T)
2     pour i allant de  (taille de T) a 1
3         pour j allant de 0 a i-1
4             si T[j+1] < T[j]
5                 T[j+1], T[j] = T[j], T[j+1]
```

Exemple 1 :

Q 1 - Ecrire l'algorithme du tri à bulles en code Python.
Q 2 - Déterminer sa complexité dans le meilleur et le pire des cas.
Q 3 - Proposer une adaptation du code pour trier la liste à plusieurs éléments suivantes : `L = [ (2,'a'), (2,'b'), (0,'a'), (4,'a')` uniquement sur la valeur numérique.
Q 4 - En effectuant quelques tests, le tri bulle est il stable ?

II. Tri rapide

1 - Principe

Le tri rapide - aussi appelé "tri de Hoare" (du nom de son inventeur Tony Hoare) ou "tri par segmentation" ou "tri des bijoutiers" ou, en anglais "quicksort" - est certainement l'algorithme de tri interne le plus efficace.

Le principe de ce tri est d'ordonner le tableau en cherchant dans celui-ci une clé pivot autour de laquelle réorganiser ses éléments. Il est souhaitable que le pivot soit aussi proche que possible de la clé relative à l'enregistrement central du vecteur, afin qu'il y ait à peu près autant d'éléments le précédant que le suivant, soit environ la moitié des éléments du tableau. On applique ensuite le tri récursivement à, sur la partie dont les éléments sont inférieurs au pivot et sur la partie dont les éléments sont supérieurs au pivot.

2 - Implémentation

Exemple 2 :

Q 5 - Proposer une fonction `separe(L,pivot)` qui renvoie deux sous-listes de `L`, la première contenant toutes les valeurs strictement inférieures `pivot` et la seconde toutes les valeurs supérieures ou égales à pivot.
Q 6 - En utilisant la fonction `separe`, proposer une fonction récursive `tri_rapide` qui renvoie une liste triée. On s'appuyera sur le pseudo-code fourni ci dessous.

Le test de votre fonction devra donner :

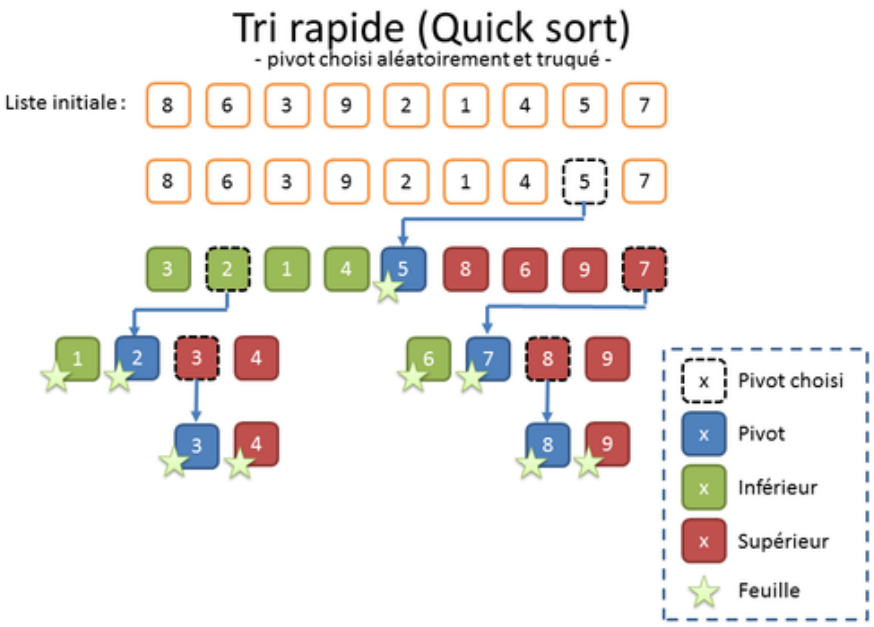

FIGURE 2 – Illustration du tri rapide

```
>>>separe([3,9,8,4,2],4)
[2,3] , [9,8,4]
>>>tri_rapide([3,9,8,4,2])
[2,3,4,8,9]
```

```
triRapide (L) :
    si la liste contient moins d un element
        renvoyer liste

    sinon
        choisir comme pivot le premier element
        separer la liste sans le pivot
            en deux sous liste selon le pivot
        Concatener le resultat de tri rapide des deux sous-listes et du pivot
```

3 - Tri sur critère

a) Extraction de données

On dispose d'un fichier texte `Notes.csv` contenant une liste de notes d'un devoir commun entre deux classes. On désire obtenir un classement général. La liste se présente sous la structure suivante :

```
L = [ (eleve1,note,classe),(eleve2,note,classe),... ]
```

La méthode `.split()` permet de séparer une chaîne de caractère en liste. Ainsi :

```
>>> mot = "Paracelse;12"
>>> mot.split(';')
["Paracelse","12"]
```

La fonction `extraction(nomFichier)` renvoie une liste `L` contenant les informations du fichier. On vérifiera que les notes sont bien des flottants :

```
# Création de la liste des notes
fichier = open("Notes.csv",'r')
L = []
for l in fichier:
    v = l.split(";")
    v[1] = float(v[1])
    v[2] =v[2].strip()
    L.append(v)
```

b) Adpatation du tri rapide

Ex. 1 :

Q 1 - Proposer une fonction `separe(L,pivot)` qui renvoie deux sous-listes de `L`, la première contenant toutes les éléments dont la note est strictement inférieure à `pivot` et la seconde toutes les notes sont supérieures ou égales à pivot.

Q 2 - Proposer une fonction `tri_rapide_note(L)` qui renvoie la liste des élèves classés par moyenne croissante.

Q 3 - Le tri est il stable ?

s plus efficaces (par exemple le tri rapide qui consiste à introduire un élément pivot autour duquel on réorganise les éléments)

III. Tri Fusion

1 - Présentation

Le tri fusion (”Merge Sort” en anglais) est un algorithme de tri efficace basé sur le paradigme ”diviser pour régner”. Il divise la liste en sous-listes plus petites, les trie récursivement, puis fusionne ces sous-listes pour obtenir la liste triée finale.

Le tri fusion fonctionne en trois étapes principales :

1. Diviser la liste en deux sous-listes de taille approximativement égale.
2. Trier récursivement chaque sous-liste.
3. Fusionner les deux sous-listes triées en une seule liste triée.

L'algorithme est naturellement décrit de façon récursive.

— Si le tableau n'a qu'un élément, il est déjà trié.

— Sinon, on sépare le tableau en deux parties (à peu près égales).

— On trie récursivement les deux parties avec l'algorithme du tri fusion.

— On fusionne les deux tableaux triés en un tableau trié.

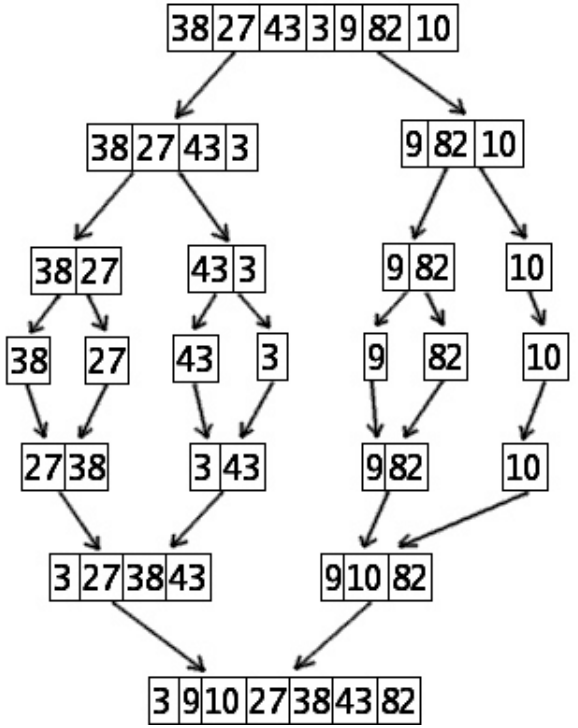

FIGURE 3 – Tri fusion pour 7 éléments

2 - code Python

a) Fusion de deux listes triées

Exemple 3 :

Q 4 - Implémenter une fonction $fusion(L1,L2)$ de paramètres $L1$ et $L2$, deux listes triés non vide, qui renvoie une liste $L3$ triée à partir des éléments de $L1$ et $L2$.

Q 5 - À l'aide de la fonction précédente, écrire une fonction `Trifusion(L)`, qui effectue le tri de la liste L selon la méthode citée plus haut.

Exemple d'attendu :

```
>>>fusion([0,2,4],[1,3,5])
[0,1,2,3,4,5]
>>>data = [(4, 'X'), (2, 'Y'), (4, 'Z'), (3, 'W'), (2, 'V')]
>>>tri_fusion(data)
[(2, 'Y'), (2, 'V'), (3, 'W'), (4, 'X'), (4, 'Z')]
```