TPinfo : Tri par sélection, tri à bulles

Objectif pédagogique du TP :
- Tri à bulles
- Lecture d'un fichier de données simples
- Tri par sélection

Le tri par ordre croissant d'une liste est l'une des premières problématiques de l'algorithmique moderne et a fait naître de nombreuses familles d'algorithmes, ainsi que des recherches mathématiques dédiées. Il s'agit d'un outil omniprésent en informatique, que ce soit dans les tableurs, en métrologie…Nous allons ici présenter deux types de tris.

Remarque : Il existe sur Python une commande pour trier une liste : on utilise la commande *sort*.

```
>>> a=[4,2,8,6,3,-2]
>>> a.sort()
>>> a
[-2, 2, 3, 4, 6, 8]
```

Le but de notre TP est d'obtenir la même chose, mais sans utiliser la commande *sort*.

I) Le tri à bulles

Le tri à bulles consiste à comparer répétitivement les éléments consécutifs d'une liste, et à les permuter lorsqu'ils sont mal triés. Il doit son nom au fait qu'il déplace rapidement les plus grands éléments en fin de tableau, comme des bulles d'air qui remonteraient rapidement à la surface d'un liquide.

L'image suivante est pris sur le site internet : https://fr.wikipedia.org/wiki/Tri_%C3%A0_bulles

Application du tri à bulles au tableau de nombres « 5 1 4 2 8 » ; pour chaque étape, les éléments comparés sont en gras.

Étape 1.

1.1. (**5 1** 4 2 8) → (**1 5** 4 2 8). Les nombres 5 et 1 sont comparés, et comme 5 > 1, l'algorithme les échange.
1.2. (1 **5 4** 2 8) → (1 **4 5** 2 8). Échange, car 5 > 4.
1.3. (1 4 **5 2** 8) → (1 4 **2 5** 8). Échange, car 5 > 2.
1.4. (1 4 2 **5 8**) → (1 4 2 **5 8**). Pas d'échange, car 5 < 8.
À la fin de cette étape, un nombre est à sa place définitive, le plus grand : 8.

Étape 2.

2.1. (**1 4** 2 5 8) → (**1 4** 2 5 8). Pas d'échange.
2.2. (1 **4 2** 5 8) → (1 **2 4** 5 8). Échange.
2.3. (1 2 **4 5** 8) → (1 2 **4 5** 8). Pas d'échange.
5 et 8 ne sont pas comparés puisqu'on sait que le 8 est déjà à sa place définitive.
Par hasard, tous les nombres sont déjà triés, mais cela n'est pas encore détecté par l'algorithme.

Étape 3.

3.1. (**1 2** 4 5 8) → (**1 2** 4 5 8). Pas d'échange.
3.2. (1 **2 4** 5 8) → (1 **2 4** 5 8). Pas d'échange.
Les deux derniers nombres sont exclus des comparaisons, puisqu'on sait qu'ils sont déjà à leur place définitive.
Puisqu'il n'y a eu aucun échange durant cette étape 3, le tri optimisé se termine.

Étape 4.

4.1. (**1 2** 4 5 8) → (**1 2** 4 5 8). Pas d'échange.
Le tri est terminé, car on sait que les 4 plus grands nombres, et donc aussi le 5e, sont à leur place définitive.

1. Ecrire une fonction Python **triabulles(liste)** qui prend une liste de nombres (de type integer ou float) en entrée et qui renvoie la même liste triée dans l'ordre croissant, par la méthode du tri à bulles.

Remarque : On remarque, comme dans l'exemple précédent, que le tri à bulles continue à fonctionner, même si la liste est triée. On va donc optimiser notre algorithme.

2. Créer une fonction **testri(liste)** qui prend une liste de nombres en entrée, et qui renvoie 'true' si la liste est triée, 'none' si elle n'est pas triée.

3. A l'aide de la fonction testri précédente, écrire une fonction **optimisetriabulles(liste)** qui prend une liste de nombres (de type integer ou float) en entrée et qui renvoie la même liste triée dans l'ordre croissant, par la méthode du tri à bulles optimisée par notre fonction testri.

II) Le tri par sélection

Le principe du tri à sélection d'une liste appelée $listenontriée$ est le suivant :

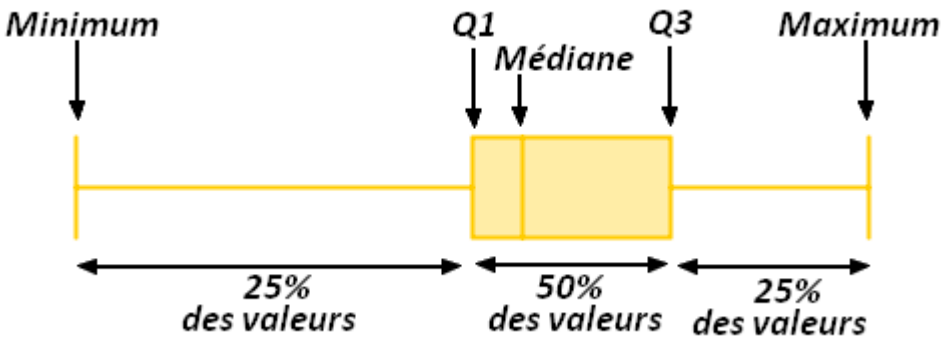
- Créer une liste vide $listetriée$
- Rechercher le plus petit élément de la liste $listenontriée$
- Ajouter à la $listetriée$ cet élément et le supprimer de $listenontriée$.
- On réitère le procédé jusqu'à ce que $listenontriée$ soit vide.

1. Ecrire un programme **triparselection(listenontriée)** qui prend une liste de nombres (de type integer ou float) en entrée et qui renvoie la même liste triée dans l'ordre croissant, par la méthode du tri par sélection.

Remarque : On pourra utiliser les commandes liste.remove(nombre) ainsi que min(liste). Mais on peut aussi s'en passer.

III) Applications

Pour représenter des données, on utilise parfois en statistique une boite à moustache, comme sur la figure ci-dessous.



On rappelle que la médiane d'une série statistique est la valeur du caractère qui partage en deux effectifs égaux les unités statistiques qui ont été rangées au préalable par valeur croissante.
On peut de même calculer le premier quartile Q_1 et le troisième quartile Q_3.

1. Ecrire une fonction **quartile(liste)** qui prend une liste de nombres (de type integer ou float) en entrée et qui renvoie le premier quartile Q_1 la médiane et le troisième quartile Q_3 de cette liste. (La liste n'est pas nécessairement triée au départ !)
2. Ecrire une fonction **boiteamoustache(liste)** qui prend une liste de nombres (de type integer ou float) en entrée et qui renvoie une représentation graphique de la boite à moustache.
3. En reprenant le document du TP traitement des données sur les données du covid, tracer la boite à moustache du nombre de comtaminer par jour.