# Retour sur le tri des listes

**Exercice 1 (𝄞).** On se propose de voir un nouvel algorithme de tri que l'on appelle tri rapide.

1. Créer une fonction `Concatene(L,M,N)` qui, à trois listes `L`, `M` et `N` renvoie la liste concaténée des éléments de `L`, `M` et `N`. Ainsi, `Concatene([1,2,8],[2],[10,-11])` renvoie `[1,2,8,2,10,-11])`.
2. Créer une fonction `Pivot(L)`, où `L` est une liste, qui renvoie un triplet **trois listes** `(G,[L[0]],D)` où `G` est est une liste contenant les éléments de `L` strictement plus petit que `L[0]` et `D` est une liste contenant les éléments de `L` d'indice est strictement positif qui sont supérieurs ou égales à `L[0]` (mais sans y ajouter `L[0]` donc) de sorte que `len(G)+1+len(D)=len(L)`.
   Par exemple si `L=[3,8,3,2,6,1]`, alors la fonction renvoie `([2,1],[3],[8,3,6])`.
3. Enfin créer la fonction récursive `TriRapide(L)` qui renvoie la version triée de `L`. Pour cela :
   - Si `L` a un élément ou moins, quelle liste faut-il renvoyer ?
   - Sinon, diviser la liste `L` grâce à la fonction `Pivot`, trier les listes `G` et `D` de façon récursive, et combiner le tout, grâce à la fonction `Concatene`.
4. Testez votre fonction sur des listes à zéro et à un élément, des listes triées et des listes non tries.
5. Tester ensuite cet algorithme sur une liste créée au hasard.

**Exercice 2 (𝄞).**
1. Écrire une fonction récursive `Fusion(G,D)`, où `G` et `D` sont deux listes triées. Cette fonction renvoie la liste contenant les éléments de `G` et `D` de façon triée. Ainsi, `Fusion([1,3,8],[2,3,4,6,7,11,13])` vaudra `[1,2,3,3,4,6,7,8,11,13]`. Pour cela :
   - Que faut-il renvoyer si `G` ou `D` est vide ?
   - Dans le cas contraire, on compare le plus grand élément de `G` à celui de `D`, si c'est celui de `G`, le plus grand, on le retire de `G` et on applique la récursivité à `G` privé de cet élément et à `D`, puis on rajoute au résultat obtenu cet élément, on procède de manière similaire si c'est celui de `D` le plus grand.
2. Tester avec `L=[1,3,8]`, `M=[2,4,6,7,11,13]`, `N=Fusion(L,M)` et `print(L,M,N)`
3. Créer une fonction **récursive** `TriFusion(L)` qui trie la liste `L`. Pour cela, si la liste a un élément ou moins, on a rien à faire pour la trier. Sinon, on coupe la liste au milieu, on trie chacune de ces deux sous-listes de façon récursive. Puis on les fusionne les deux sous-listes triées grâce à la fonction `Fusion` créée à la question précédente.

**Exercice 3.** À l'aide d'un des deux tris, écrire une fonction `Mediane(L)` qui à une liste `L` de nombres et non vide renvoie sa médiane.

**Exercice 4.** Considérer l'une des deux méthodes de tri vues aujourd'hui : tri fusion ou tri rapide et une des deux méthodes déjà vues (tri à bulles, insertion ou sélection) : afficher sur un même graphe en fonction de $n$ le temps de trier une liste aléatoire de longueur $n$ pour les deux méthodes. Que peut-on en conclure ?

**Exercice 5.** Soit `L` une liste quelconque d'éléments (pouvant être des nombres ou pas) et $h$ une fonction définie sur l'ensemble des éléments de `L` à valeurs dans $\mathbb{R}$. On dit qu'on trie la liste `L` suivant la clé $h$, si on a un algorithme qui renvoie `M` une liste contenant les mêmes éléments que `L` à permutation triée, telle que

$$\forall i \in [\![\, 0\,;n-2 \,]\!] \qquad h(M[i]) \leqslant h(M[i+1])$$

Où $n$ est la longueur de la liste `L`. Adapter le tri rapide ou le tri fusion, en une fonction `TriÀclef(L,h)` qui trie suivant la clé $h$.

# Les classes d'alcools dans le vin

Ruben est un élève peu sérieux en chimie. Monsieur F, son professeur de phyique-chimie lui a enseigné qu'il y avait trois classes (notées 0, 1 et 2) d'alcools dans le vin. Monsieur F a aussi expliqué comment, à l'aide des différentes caractéristiques chimiques, on pouvait déterminer la classe du vin.

Seulement, Ruben n'a pas écouté la méthode pour déterminer la classe du vin. Ainsi, il a une liste de 178 vins avec chacune les différentes caractéristiques chimiques. Mais il a seulement la classe des vins pour ceux d'indice pairs (en effet, Ruben n'a décidé de prendre seulement la moitié du cours). Il va falloir aider le pauvre Ruben à reconstituer, du mieux possible, les classes manquantes. Pour cela, l'algorithme des $k$-plus proches voisins semble tout approprié : si un vin donné a parmi les 3 plus proches vins, 2 vins de classe 1 et un vin de classe 0, on pourra en déduire que ce vin est probablement de classe 1.

On va charger les données de Ruben dans Python grâce aux commandes suivantes :

```
from sklearn import datasets # À mettre en début de fichier

W = datasets.load_wine()
Li = W.data.tolist()#Li[i] caractéristique du vin numéro i
N = W.feature_names#liste des caractéristiques
T = W.target#T[i] la classe du vin numéro i
donnees = list(range(0,len(Li),2))#liste des indices des vins
#où Ruben a l'étiquette
test = list(range(1,len(Li)-1,2))#liste des indices des vins
#où Ruben n'a pas l'étiquette
print(Li[0])
print(N)
print(T[0])
```

Ainsi `Li[0]` est la liste des caractéristiques du vin numéro 0. `Caractéristiques` est la liste des caractéristiques :

- comme `Caractéristiques[0]` est égal à `'alcohol'` et que `Li[0][0]=14.23`, on peut dire que le degré d'alcool du vin numéro 0 est de 14.23.
- comme `Caractéristiques[4]` est égal à `'magnesium'` et que `Li[0][4]=127`, on peut dire que la quantité de magnésium du vin numéro 0 est de 127 [1].
- Par contre, Ruben a bien noté que `T[0]` vaut 0, donc que le vin numéro 0 a pour classe 0.

**Exercice 6.** 1. 𝄞Créer une fonction `Dist(M,N)`, où `M` et `N` sont deux listes de même longueur $n$, renvoie la distance entre ces deux listes à savoir :

$$\sqrt{\sum_{k=0}^{n-1}(M[k]-N[k])^2}$$

2. Créer une fonction `DicoOccurrences(L)` qui, à une liste `L`, renvoie le dictionnaires du nombre d'occurrences, c'est-à-dire que les clés de ce dictionnaire sont les éléments de la liste et la valeur associée à une clé est le nombre d'occurrences de cet élément dans `L`.
3. En déduire une fonction `Majoritaire(L)` qui renvoie un élément dont le nombre d'occurrences est majoritaire.
4. Créer la fonction `PlusProchesVoisins(i,k)` qui renvoie la classe majoritaire parmi les $k$ plus proches voisins de `Li[i]`.
5. Monsieur F a enfin pitié du pauvre Ruben et décide de lui donner les classes manquantes, grâce à cela en déduire la matrice de confusion avec $k=3$. Quel pourcentage de réussite avait Ruben ? La matrice de confusion est ici une matrice $M\in\mathscr{M}_3(\mathbb{R})$ tel que $m_{i,j}$ est le nombre de vin de classe $i$ qui ont été pris pour des vins de classe $j$.
6. En faisant varier $k$, trouver la valeur de $k$ optimal.
7. On peut décider de modifier la fonction distance et de calculer : $\sum_{k\in I}(M[k]-N[k])^2$ pour $I\subset [\![0\,;n-1]\!]$ avec $I$ non vide. Déterminer $I$ de façon optimale (c'est-à-dire à maximiser le nombre de bonne déduction de l'algorithme).

1. Dans une certaine unité que Ruben n'a pas notée.