# TP - Exercices sur les tris

---

## I Tri sur un ordre particulier

On considère des tableaux de couples dont la première coordonnée est un entier, et la seconde une chaîne de caractère.
On dira que le couple `(n,ch1)` est plus petit que le couple `(p,ch2)` si

`n<p ou (n=p et ch1<ch2).`

On admet que cela définit un ordre total sur `int`×`str`.

1. Écrire une fonction `compare(T,i,j)` qui renvoie `True` si `T[i]` est plus petit que `T[j]` avec l'ordre défini ci-dessus, et `False` sinon.
2. Adapter une fonction de tri du cours pour trier un tel tableau.

## II Estimation des temps des différents tris

On revient sur la notion de tri de tableaux de flottants, d'entiers ou de chaînes de caractères où le test `<` de pyhon est suffisant. On pourra tester les fonctions sur des tableaux aléatoires, par exemple construits à l'aide de

```
from random import randint

def tab_alea (n):
    t = [0]*n
    for k in range (n):
        t[k] = randint (0,100)
    return t
```

On utilise le module `time` pour mesurer le temps que prend un programme.
On peut s'inpirer du programme suivant pour mesurer le temps qu'il faut à votre ordinateur pour effectuer `n` affectations.

```
import time

def test(n):
    temps1=time.clock()
    for i in range(n):
        p=i
    temps2=time.clock()-temps1
    print(temps2)
```

**Exercice 1.** Proposer une méthode pour estimer le temps moyen utilisé par un algorithme de tri de tableau.
On demande une représentation graphique.
On rappelle le code minimum suivant pour représenter des lignes brisées reliant les points de coordonnées `(i,L[i])`.

```
import matplotlib.pyplot as plt

plt.hlines(y=moyenne2, xmin=0, xmax=n, color='g', linestyle='-', label
    ='Tri') # Affiche une
ligne entre xmin et xmax
plt.legend(bbox_to_anchor=(0,1), loc="center left") # Ajoute une
    legende
plt.plot(L, color='red') # Affiche la courbe des valeurs de la liste L
plt.show()
```

**Exercice 2** (Comparaison entre tri par insertion et tri rapide)**.** Reprendre les algorithmes du cours de tri par insertion et tri rapide.

1. Les modifier de telle sorte qu'ils renvoient également le temps utilisé par l'ordinateur pour effectuer ces tris.
2. Comparer leur efficacité « en moyenne ».

**Exercice 3** (Tri par comptage)**.** On se donne deux entiers $n$ et $m$ et $T$ un tableau de longueur $n$ d'entiers compris entre 0 et $m$.

1. Expliquer ce que fait la fonction python suivante :

```
def comptage(T,n,m):
    C=[0]*(m+1)
    for k in range(n):
        C[T[k]] += 1
    return C
```

2. En déduire une fonction `tri_comptage(T,n,m)` qui permet, pour $m$ fixé, de trier le tableau $T$ en $O(n)$.
3. Comparer cette complexité avec celles des tris vus en cours.