**M2 TIDE**

# Algorithmes de tri et expérimentation de la complexité via l'efficacité des fonctions

L'objectif du TP est d'implémenter trois algorithmes de tri en Python, en comparant leur temps de calcul (sur des listes aléatoires d'entier).

Ici il faut comprendre le « tri » comme « tri d'entiers par ordre croissant ».

## Algo 1 : Tri bulle

https://fr.wikipedia.org/wiki/Tri_%C3%A0_bulles

C'est l'algorithme présenté la semaine dernière. Implémentez-le en Python sous la forme d'une fonction qui prend une liste en paramètre et la trie. Testez votre fonction en vérifiant avec print le résultat obtenu en triant les listes suivantes :

1,2,3
3,2,1
3,1,2
1,3,2,4,5
5,4,3,2,1
1,3,4,2,5,
0,0,1,0,0

## Expérimentation 1 :

1) Écrivez une fonction qui prend un paramètre n et retourne une liste aléatoire d'entiers de longueur n, chaque entier étant compris entre 0 et 10*n.
Vous pouvez utiliser la librairie random.
Testez votre fonction visuellement.
2) Cherchez comment calculer le temps d'exécution d'une fonction en Python.
https://docs.python.org/fr/3.9/library/timeit.html
Expérimentez sur votre première fonction de tri.
3) En utilisant les questions précédentes et matplotlib, tracez le temps d'exécution de votre programme en fonction de la taille du tableau aléatoire à trier. À vous de voir quel intervalle de taille donne des résultats intéressants.

## Expérimentation 2 : sort

Cherchez sur internet quel est l'algorithme implémenté dans la méthode sort des listes de Python.
Ajoutez à votre dessin de la question précédente une courbe représentant le temps d'exécution de sort en fonction de la taille de la liste aléatoire.

## Algo 3 : Tri fusion

https://fr.wikipedia.org/wiki/Tri_fusion
Comprenez puis implémentez le tri fusion, testez votre fonction, puis comparez son efficacité aux algos des questions précédentes.

## Conclusion : que conclure ?

Nous corrigerons cela ensemble la semaine prochaine.