

TP°5 : algorithmes de tri

Exercice 1 : fonction de création manuelle de tableau

Écrire la fonction **T = Creer_Tableau()** qui à à partir des entrées clavier de l'utilisateur crée un tableau T des n éléments entrés.

L'utilisateur devra taper '\$' pour terminer sa saisie.

Exercice 2 : fonction de génération de tableaux

Écrire la fonction **T = Genere_Tableau(n)** qui génère un tableau de n valeurs entières aléatoires entre 0 et 100 000.

Exercice 3 : fonction d'insertion d'élément

Comme vu dans le TD 3, programmez la fonction Octave **T = Insérer_Element(T, p_init, p_fin)** qui réalise l'insertion d'un élément à partir d'un tableau de nombre entiers, d'un indice de position initiale (p_init) et d'un indice de position finale (p_fin).

Exercice 4 : tri par insertion

Comme vu dans le TD 3, programmez une fonction Octave qui réalise le tri par insertion d'un tableau.

Exercice 5 : tri par sélection

Comme vu dans le TD 3, programmez une fonction Octave qui réalise le tri par sélection d'un tableau.

Exercice 6 : tri à bulles (Bubble sort)

Écrivez la fonction **tri_a_bulles** définie par l'algorithme suivant

```
fonction t= tri_a_bulles(tableau t)
    taille = taille(t);
    pour i de taille à 1
        pour j de 2 à i
            si T[j-1]>T[j] alors
                échanger T[j-1] et T[j]
            finsi
        finpour
    finpour
fin
```

Exercice 7 : tri rapide (QuickSort)

Ecrivez les fonctions **partitionner**, **tri_rapide** et **tri_r** définies par les algorithmes suivants :

```
fonction [p, T] = partitionner(tableau T, premier, dernier)
    pivot = T[premier] ;
    p = premier ;
    pour i de premier+1 à dernier
        si T[i] <= pivot alors
            p = p+1 ;
            échanger T[i] et T[p]
        finsi
    finpour
    échanger T[premier] et T[p]
fin

fonction t = tri_rapide(tableau t, premier, dernier)
    début
        si premier < dernier alors
            [pivot, t] = partitionner(t,premier,dernier)
            t = tri_rapide(t,premier,pivot-1)
            t = tri_rapide(t,pivot+1,dernier)
        fin si
    fin

fonction t= tri_quicksort(tableau t)
    taille = taille(t);
    t = tri_rapide(t, 1, taille);
fin
```

Exercice 8 : comparaison des quatre méthodes de tri

Les trois méthodes de tri seront évaluées au moyen de deux variables : **le nombre d'éléments à trier** et **le temps nécessaire** pour réorganiser le tableau (utilisez par exemple les fonctions Octave **tic()** et **toc()**).

Il vous est demandé d'évaluer ces trois méthodes de tri pour des tableaux de taille de **50 à 1 050** éléments (aléatoires) en augmentant à chaque itération la taille du tableau de 100 éléments.

A chaque fois, vous devrez créer un tableau de la taille demandée avec des valeurs générées aléatoirement. Ce tableau devra alors être trié successivement avec les trois méthodes de tri. Pour chaque taille de tableau, les tris seront évalués 10 fois chacun afin de voir si le temps d'exécution varie selon les éléments.

Vous générerez aussi un tableau déjà trié et un tableau trié de manière inverse et effectuerez les mêmes opérations que précédemment.

Les résultats d'évaluation seront présentés :

- **sous la forme d'un tableau** pour les résultats concernant le temps d'exécution. Ce tableau aura la forme suivante :

Taille	Temps moyen insertion	Temps moyen sélection	Temps moyen tri à bulles	Temps moyen QuickSort
--------	--------------------------	--------------------------	-----------------------------	--------------------------

Les temps moyens seront calculés sur les 10 essais effectués pour chaque tri et chaque taille.

- **sous la forme d'un graphique 2D** représentant le temps nécessaire pour chaque méthode de tri. Le graphique aura comme abscisse le nombre d'éléments du tableau à trier, en ordonnée la variable étudiée (temps) et chaque méthode de tri sera représentée par une courbe de couleur différente.

Liens :

- **Les tris sous forme de dance folklorique** : <http://www.laboiteverte.fr/algorithmes-tri-visualises-danses-folkloriques>