Projet : « Tri (très) sélectif »

L'étude de l'efficacité d'un algorithme porte souvent sur deux principaux facteurs : le temps d'exécution et l'espace mémoire nécessaire pour résoudre un problème. Bien que les performances des ordinateurs ne cessent de croître de manière exponentielle, il est toujours intéressant d'utiliser des algorithmes performants ne serait-ce que parce que la quantité des données que ces algorithmes doivent traiter est aussi en constante augmentation.

Dans ce projet, nous allons nous intéresser aux performances de quelques algorithmes de tri (simples) et essayer de déterminer leur complexité temporelle (qui s'exprime en fonction de **n**, taille des données).

Outils généraux

- Ecrire tout d'abord la fonction **T = Genere_Tableau(n,valmax)** qui génère un tableau de **n** données entières aléatoires entre **0** et **valmax**.
- Comme vu dans le TP 5/6, *reproduisez* une fonction Octave qui réalise l'insertion d'un élément à partir d'un tableau de nombre entiers, d'un indice de position initiale et d'un indice de position finale.

Algorithmes de tri

Coder les cinq algorithmes de tri suivants :

- **Tri_Insertion** : comme vu dans le TP 5/6, programmez une fonction Octave qui réalise le tri par insertion d'un tableau.
- **Tri_Selection** : comme vu dans le TP 5/6, programmez une fonction Octave qui réalise le tri par sélection d'un tableau.
- **Tri_Bulle (BubbleSort)** : comme vu dans le TP 5/6, programmez une fonction Octave qui réalise le tri à bulles d'un tableau.
- **Tri_Rapide** (QuickSort) : comme vu dans le TP 5/6, programmez une fonction Octave qui réalise le tri rapide d'un tableau
- **Tri_Shell** : Ecrivez enfin la fonction **tri_Shell** définie par :

```
fonction T = tri Shell(tableau T)
     pas = 0
     longueur = taille(T)
     tant que pas < longueur faire
          pas=3*pas+1 ;
     fintantque
     tant que pas !=0 faire
          pas = pas/3
          pour i=pas+1 à longueur faire
               valeur = T[i]
               j = i
               tant que (j>pas) et (T[j-pas]>valeur)
                    T[j] = T[j-pas]
                    j=j-pas
               fintantque
               T[j]=valeur
          finpour
     fintantque
fin
```

Comparaison des cinq méthodes

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

Il vous est **demandé** d'évaluer ces cinq méthodes de tri pour des tableaux a minima de taille de **10** à **5 010** éléments en augmentant à chaque fois la taille du tableau de **100 éléments**. Pour chaque taille, les tris seront évalués au moins 10 fois chacun afin de pondérer le temps d'exécution de chaque méthode. (vous pouvez bien évidemment tester pour des valeurs plus grandes)

A chaque opération, vous devrez créer un tableau de la taille demandée et remplir chaque cellule avec une valeur générée aléatoirement (pensez aux fonctions codées dans la section « outils »). Ce tableau devra alors être trié successivement avec les conq méthodes de tri.

Les résultats d'évaluation seront compilés :

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

Taille Tableau	Temps moyen ***tri 1***	Temps moyen ***tri 2***	…	Temps moyen ***tri n***

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

3. Sous la forme de graphiques représentant les temps nécessaires 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 avec un texte associé.

De la même manière, vous devez estimer le temps minimum et maximum pour trier un tableau pour chacun des algorithmes de tri pour chaque taille requise.

Essayez enfin pour chacune des méthodes de déterminer la complexité temporelle pour chacune des méthodes de tri : l'évolution du temps de tri évolue-t-il de manière linéaire, suivant une fonction en x^2, suivant une loi logarithmique, autre ?

Concluez enfin sur la « ***meilleure*** » méthode de tri d'éléments parmi celles proposées en justifiant vos ou votre choix.

Travail à rendre

Vous devrez rendre pour le dimanche **12 janvier 2020 23h55 GMT dernier délai** par voie électronique (à l'adresse : **Philippe.Truillet@irit.fr**) :

- Les fichiers **.m** que vous aurez créés. Il vous est conseillé de faire un fichier pour chaque fonction de calcul un fichier pour l'évaluation et la comparaison des méthodes de calcul ;
- Un **rapport** décrivant les algorithmes que vous aurez programmés (notamment pour l'évaluation) en expliquant votre démarche de travail.

Une présentation orale de 10 minutes (avec documents pour la vidéo-projection) aura enfin lieu après la fin du projet **entre le 13 et 17 janvier 2020**.