Les tris

1) Evaluation.

L'objectif de la première manipulation est d'étudier l 'efficacité des différents tris
Vous disposez sur le site webgi d'un outil permettant de trier un tableau d'entiers et d'obtenir soit un temps de réalisation soit les nombres de comparaisons et d'affectations nécessaires.

Pour trier, vous devez définir la taille du tableau puis créer le tableau. Chaque tri exécuté travaille sur une copie du tableau original. Ainsi, des exécutions sur des données aléatoires ne peuvent pas pénaliser un tri. Enfin, vous pouvez afficher le tableau. Dans ce cas le tri travaille sur 30 éléments.

Vous devez réaliser les différents tris sur des jeux de données de tailles croissantes (triés ou non). Les résultats obtenus seront exploités pour établir des tables et des graphiques permettant de comparer les tris.

Note : L'outil est actuellement développé sous le système window, vous devrez donc passer momentanément sur les machines windows.

2) Préparation.

Vous pouvez reprendre la classe Tri du premier TP ou, pour la mise au point des tris, modifier votre classe pour utiliser la classe « donnees_visu ».
Cette dernière permet de visualiser le comportement de votre algorithme sur un tableau d'entiers limité à 40 éléments.
Cette classe fournit les services suivants :

public donnees_visu(int delai) : initialisation de l'objet avec précision du délai minimum entre deux affichages en ms
public void definition_taille(int nbrElements) : création du tableau de taille « nbrElements »
public void setValue(int index, int val) : affectation de la valeur val de l'élément de position index
public int getValue(int index) : récupération de la valeur de l'élément de position index
public int comparerIndex(int index1, int index2) : comparaison de deux éléments désignés par leurs positions. Le résultat est <0, =0 ou >0 selon que le second est > , = ou < au premier .
public int comparer(int index, int val) : comparaison d'un élément avec une valeur
public boolean infIndex(int index1, int index2) : test d'infériorité entre deux éléments
public boolean inf(int index, int val2) : test d'infériorité entre un élément et une valeur
public boolean supIndex(int index1, int index2) : test de supériorité entre deux éléments
public boolean sup(int index, int val2) : test d'infériorité entre un élément et une valeur
public void echange(int index1,int index2) : échange de deux éléments désignés par leurs positions
public int taille() : taille du tableau
public int nbComparaison : nombre de comparaisons réalisées dans l'objet
public int nbAffectation : nombre d'affectations réalisées par l'objet (un échange correspond à trois affectations).

L'objet affiche le tableau et vous indique si vous réalisez des débordements de l'intervalle des indices.

Pour utiliser cette classe vous devez récupérer le fichier JAR "visu_tri.jar" contenant la classe "données-visu".
Pour votre projet vous devez préciser que vous utilisé une bibliothèque supplémentaire qui sera le fichier JAR.
Pour ce faire si le projet est créé. Un clic-droit sur le projet dans la vue package vous permet de demander les propriétés. Une fois la fenêtre ouverte, choisissez "chemin de compilation", puis l'onglet bibliothèques.
Enfin le bouton "ajouter jar externe" vous permet de choisir un fichier JAR contenant des classes.

3) le tri shell.

3.1 implantation du tri

Le tri par insertion doit sa lenteur au fait que les échanges ne portent que sur des éléments adjacents. Par exemple, si le plus petit élément est situé en fin de tableau, N étapes sont nécessaires pour le ramener à sa place définitive. Le tri Shell est une généralisation simple et plus efficace du tri par insertion, car permettant l'échange d 'éléments éventuellement très éloignés.
L'idée est de réorganiser la suite d'éléments de manière à obtenir une sous suite ordonnée en sélectionnant tous les $h^{ièmes}$ éléments. Une telle suite est dite h-ordonnée et elle est constituée de h sous-suites ordonnées indépendantes, entrelacées.

En h-ordonnant la suite pour une grande valeur de h, on échange des éléments très distants dans le tableau, afin de faciliter la même opération pour les valeurs inférieures de h. L'utilisation d'une telle méthode avec une série décroissante de h terminée par 1 résulte en un tableau trié. C'est le principe du tri Shell.

La suite de valeurs possibles pour h finit par 1, les pas précédents étant de 4, 13, 40, 121, 364, 1093... ($P_i=3*P_{i-1}+1$). D'autres suites sont évidemment possibles, à condition de prendre des valeurs qui ne soient pas multiples entre elles (pour ne pas toujours traiter les mêmes éléments et laisser de côté les autres, par exemple les puissances successives de 2 ne traiteraient que les positions paires, sauf au dernier passage.

L'intérêt de ce tri, bien qu'il ait une boucle autour du tri par insertion, est qu'il crée rapidement un tableau presque trié, le dernier tri par insertion sera donc beaucoup plus rapide.

Note : Une manière d'implanter le tri Shell serait d'utiliser pour chaque valeur de h, un tri par insertion de manière indépendante, sur chacun des h suites. On peut faire beaucoup mieux que cela : si on remplace chaque apparition de 1 par h (et de 2 par h+1) dans le tri par insertion.

Développez une application java du tri shell.

3.2 Complexité du tri

Modifiez les tris de façon à afficher en fin de traitement le nombre total de déplacements d'éléments du tableau et le nombre total de comparaisons.

```
> java TriInsertion 100
Nombre d'affectations : 300
Nombre de comparaisons : 4950
>
```

Comparer le tri par insertion et le tri shell.
Que ce passe t'il lorsque l'on augmente le nombre d'éléments ? (La suppression de l'affichage des tableaux triés et non triés pourra être utile.)

3.3 Temps d'exécution

Vous pouvez chercher à vérifier le comportement de vos algorithmes en mesurant les temps de traitement.
Voir : http://java.sun.com/j2se/1.4/docs/api/java/util/Date.html

Ou voir les possibilités de l'option –Xprof de la commande java
http://java.sun.com/j2se/1.4/docs/tooldocs/solaris/java.html

applet du tri shell permettant de visualiser le rôle du choix des valeurs successives de h : test tri shell

4) Le tri fusion.

Ecrivez le tri fusion qui consiste à trier un tableau en commençant par trier les deux moitiés puis en réalisant une fusion de ces dernières.

5) Le tri bulle et récursivité.

Ecrivez le tri bulle sans aucune boucle itérative. On procèdera en développant une procédure bulle puis en éliminant les boucles dans cette procédure et dans l'algorithme de tri en utilisant la récursivité.

6) Le tri rapide et élimination de la récursivité.

Ecrivez le tri rapide sous sa forme récursive et sous sa forme itérative.
Le choix du pivot et la méthode de répartition basée sur ce pivot sont à votre convenance.