Algorithmique avancée
TD n° 7

Tris et complexité

1ere partie

Le temps d'exécution des algorithmes est un élément important de la qualité du logiciel.
Nous allons comparer les temps d'exécution de quelques algorithmes servant à trier des tableaux. Nous disposons pour cela de la classe abstraite **Tri**, dont le code figure ci dessous.

Quand vous aurez lu ce code, vous constaterez que pour étudier un algorithme de tri particulier, il suffit de créer une sous-classe et d'y implémenter l'algorithme dans la méthode de nom **triMethod()** en utilisant les méthodes lire() et echanger() pour accéder au tableau. Pour trier un tableau donné, on crée une instance de la classe en lui passant le tableau en question, et on lance le tri par **trier()**.

- Comment peut on ensuite récupérer le nombre d'opérations effectuées et le temps de calcul ?
- Quand on a implémenté un algorithme qui trie du plus petit au plus grand, comment implémenter le même algorithme pour qu'il trie du plus grand au plus petit ?

```
abstract class Tri {
   private int table[] ;
   private long compteurSwap,
compteurComp ;
   private long time ;
   protected int taille ; // héritable par les
classes qui implémentent un Tri

   Tri(int tab[]) {
     table = tab ;
     taille  = table.length ;
     compteurSwap = 0 ;
     compteurRead = 0;
   }// fin du constructeur

   public long getNumberSwap() {
     return(compteurSwap) ;
   }
     public long getNumberComp() {
     return(compteurComp) ;
   }
     public long getTime() {
     return(time) ;
   }

   public void affichertemps() {
     System.out.println("Duree d'execution " +
time + " millisecondes. ");
 }
boolean read(int i) {
                return table[i] ; }// fin de read()
void swap(int pos1, int pos2) {
  int tmp ;
  tmp = table[pos1] ;
  table[pos1] = table[pos2] ;
  table[pos2] = tmp ;
  compteurSwap++;
}//fin de swap

boolean after(int pos1, int pos2) {
  compteurComp++ ;
        return isAfter(pos1,pos2) ;
}// fin de after

/** Cette méthode peut etre masquée dans une
sous-classe  pour changer l'ordre de tri */
   boolean isAfter(int pos1, int pos2)   {
     return table[pos1]>table[pos2];
   } // fin de isAfter

   abstract void triMethod() ;

  public void trier() {
    long time0 = System.currentTimeMillis() ;
    triMethod();
    long time1 = System.currentTimeMillis() ;
    time =(time1 - time0) ;
  }// fin de trier
} //fin de la classe Tri

void afficheDonnées() {/* non recopiée */ }
String toString() {/*non recopiée */}
```

exemple du tri à bulle :

algorithme :

Paramètre : entier n /* la taille du tableau */, tableau[0…n-1],
variables : entier i ; booléen changé ;
début :

```
Répeter
    i = 0;
    changé = faux ;
    Tant que (i < n-1 ) faire
        si (tableau[i] > tableau[i+1])
            échanger(tableau[i] , tableau[i+1])
            changé = vrai
        finsi
        i = i + 1 ;
    fin Tant que
Tant que (changé == vrai)
```

fin

classe java correspondante :

```
public class TriBulle extends Tri
 { public TriBulle(int tab[])
    { super(tab);}

   public  void triMethod()
    { boolean changed ;
      do
       { changed = false ;
         for(int i=0;i<taille-1;i++)
          { if(after(i,i+1)
            { changed = true ;
              swap(i,i+1);
            }// fin du if
          }// fin du for
       }while (changed);
    }//fin de triMethod
  }// fin de la classe Tribulle
```

utilisation :

```
import java.util.Random ;
public class TestBulle
 { public static void main(String args[])
   {  Tri tri;
      int nb = 15;
      int tabBulle[] = new int[nb] ;

      //initialisations
      Random rand = new Random();
      for(int i=0; i< nb ; i++)
        { tabBulle[i] = rand.nextInt(1000) ;
        }//fin du for
      tri = new TriBulle(tabBulle) ;
      tri.afficherDonnées(); // pour la mise au point et le debugage
      tri.trier() ;
      tri.afficherDonnées(); // pour la mise au point et le debugage
    }//fin du main
}
```

Sur ce modèle vous devez rédiger deux classes **TriMaxSelect** et **TrisTas.** Vous utiliserez aussi pour les tests de temps de calcul la classe **TriBullePasVariable** qui vous sera fournie.
Vous trouverez ci-dessous les algorithmes du tri par sélection du maximum ainsi que celui du tri à bulle à pas variable. Pour le tri par tas, reportez-vous au cours et au TD 6 (plus particulièrement à sa question 7). L'implémentation du tas ayant été donnée en corrigé, celle du tri par tas demande seulement quelques lignes de code supplémentaire.

```
/* Tri par sélection du maximum */
Paramètre : entier n /* la taille du tableau */, tableau[1…n] ; /* attention aux indices */
variables : entier lastind, courind, maxind,; //ind pour indice
initialisation :lastind =n ;
début :
tant que(lastind>1 ) faire
    maxind = 1 ;
    courind=2;
  tant que(courind<=lastind;) faire
      si(tableau[courind]> tableau[maxind])
         maxind=courind;
      finsi
      courind = courind + 1 ;
  fin tant que
    échanger(tableau[maxind],tableau[lastind]);
    lastind = lastind - 1
fin tant que
fin
```

```
/* Tri à bulle à pas variable */
Paramètre : entier n /* la taille du tableau */, tableau[1…n] ;
variables : entier i , inc , start, stop ; booléen changé ;
initialisation :  inc = la plus grande puissance de 2 inférieure à n
début :
   Faire
      i = 1; start = 1; stop = n ; changé = faux ;
      Tant que (i ≤ stop - inc) faire
          si (tableau[i] > tableau[i+inc])
             échanger(tableau[i] , tableau[i+inc])
             changé = vrai
          finsi
          i = i + 1 ;
      fin Tant que
      si (changé == vrai)
           start = start + inc ;
           stop = stop – inc ;
      sinon
            inc = inc / 2 ;
      finsi
   Tant que (inc > 0 OU changé == vrai)
fin
```

2eme partie

Faire des tests comparatifs des performances des 4 algorithmes sur le même jeu de données.
1. Par des essais successifs, déterminer pour chaque algorithme combien environ il peut trier de données en 10 secondes (on utilisera la classe de test dont le code est sur les disques)

2. Comparer les performances des algorithmes en essayant de déterminer pour différentes tailles de tableaux, le gain de temps apporté par un meilleur algorithme, et pour chaque algorithme, le plus grand tableau avec lequel on peut l'utiliser. On utilisera pour cela la classe Test ci-dessous, qui contient une solution pour le tirage au sort des tableaux (avec un **import java.util.Random**), et les outils fournis dans ~levy/ pour étudier les résultats : le main() de cette classe produit deux fichiers de résultats. resultat.txt est fait pour être lu avec Afficheur, et moyenne.txt avec Visualiseur.jar

```
public class Test  {
    public static void main(String args[])     {
        /* initialisations */
        // la liste des noms doit correspondre avec "instanciation des differents tris"
        String nom[] = {"selection","bulle","bulle pas variable"," par tas"};
        int nbTris = nom.length ;
        Tri tri[] = new Tri[nbTris];
        // La liste des tailles de données à tester est passée en ligne de commande
        // avec une liste par défaut si args[] est vide
        int taille[]= {1000,2000,3000,4000,6000,8000,10000} ;
        int nbTailles = args.lenth;
        if (nbTailles == 0) {
            nbTailles = taille.length ;
        } else {
            taille = new int[nbTailles] ;
            for(int i = 0 ; i < nbTailles ; i++) {
                taille[i] = Integer.parseInt(args[i]) ;
            } // fin du for
        }// fin du if
         // autres initialisations
        int nbEssais = 6;
        long tempsMoyen [][] = new long [nbTris][nbTailles];
        for (int i=0;i<nbTris;i++)
            for (int j=0;j<nbTailles;j++)
                tempsMoyen [i][j] = 0;
        Random rand = new Random();

        try {
            // ouverture fichier texte pour Afficheur
            FileWriter fOut= new FileWriter("resultat.txt");
            BufferedWriter bOut=new BufferedWriter(fOut);
            PrintWriter  pOut = new PrintWriter(bOut);
            // ecriture des noms des tris dans l'ordre
            for(int i = 0 ; i < nom.length ; i++)
                System.out.print(nom[i] + ";") ;
            System.out.println("") ;

/* execution de nbEssais tests pour chaque taille et chaque tri */
            for (int nb =0 ; nb  < nbTailles ; nb++)  {
                for (int nbe=0 ; nbe < nbEssais ; nbe++)    {
                    // réservation d'espace pour ce jeu de tests
                    int tabSel[] = new int[taille[nb]] ;
                    int tabBull[] = new int[taille[nb]] ;
                    int tabPas[] = new int[taille[nb]] ;
                    int tabTas[] = new int[taille[nb]] ;
                    // tirage au sort d'un tableau dont chaque tri a une copie
```

```
                    for(int i=0; i< taille[nb] ; i++)   {
                        tabSel[i] = rand.nextInt() ;
                        tabBull[i] = tabSel[i];
                        tabPas[i] = tabSel[i];
                        tabTas[i] = tabSel[i];
                    }//fin du for( int i=0;..
                     // instanciation des differents tris
                    tri[0] = new TriMaxSelect(tabSel) ;
                    tri[1] = new TriBulle(tabBull) ;
                    tri[2] = new TriBullePasVar(tabPas) ;
                    tri[3] = new TriTas(tabTas) ;
                    //execution du jeu de test
                    for (int i=0;i<nbTris;i++)   {
                        tri[i].trier() ;
                        // calcul des temps cumulés
                        tempsMoyen [i][nb]= tempsMoyen [i][nb] + tri[i].getTime();
                        // enregistrement des resultats pour Afficheur ou lecture
                        pOut.println(tri[i]);
                    }
                    pOut.println("");
                } //fin du for nbe
                // calcul des temps moyens par taille et par tri
                for (int i=0;i<nbTris;i++)
                    tempsMoyen [i][nb]= tempsMoyen [i][nb] /nbEssais;
            }// fin du for nb

            pOut.close();
            bOut.close();
            fOut.close();

/* enregistrement des resultats pour Visualiseur */
            fOut= new FileWriter("moyenne.txt");
            bOut=new BufferedWriter(fOut);
            pOut = new PrintWriter(bOut);

            for (int i=0;i<nbTris;i++)   {
                pOut.println(nom[i]); // le nom du graphe
                pOut.println("taille ; temps"); // les noms des axes
                pOut.println("0 ; 0"); // la courbe part de l'origine
                for (int j=0;j<nbTailles;j++)
                    pOut.println(taille[j] + " ; " + tempsMoyen [i][j]); // les points
                pOut.println("");
            }

                        // fin d'enregistrement
            pOut.close();
            bOut.close();
            fOut.close();
        }
        catch(IOException e){System.err.println(e);}

    }//fin du main
}
```