Projet N°2

Exercice N°2

Equipe 13 composé de :

- HAUTOT Sarah
- CHALANSONNET Adrian
- GIROMELLA Hugo

Sommaire:

A)Intro

Il existe plusieurs types de tri différent, le tri à bulle, par sélection et par insertion.

Dans cette partie nous allons regarder quel tri est le plus optimisé en regardant le temps pris par chaque tri pour trier trois type de tableau :

- Un tableau généré aléatoirement
- Un tableau totalement inversé
- Un tableau déjà partiellement trier

Vous trouverez donc dans ce pdf les explications de chaque tri et de comment on a fait pour que les tri aient le même tableau.

B)Les tries et s'assurer qu'il marchents

1/triSelection

Dans le programme java du tri par sélection on cherche la case contenant la plus grande valeur (valeur maximale):
On le met ensuite à la fin du tableau donc dans un tableau de 0 à x on le met à la position x

Puis on répète l'opération dans le tableau de 0 à x-1, puis x-2, x-3 etc…
La longueur du tableau moins la variable tour moins 1, correspondant à la case avec laquelle le max doit être permuté

Et ce jusqu'à ce qu'il ne reste qu'une valeur qui est alors triée d'office grâce au tri des autres cases.

```
public static void triSelection ( int[] tab )
{
	int   max;
	int   tempo;

	for(int tour = 0; tour < tab.length;tour++)
	{

		max = 0;
		for (int cpt = 0; cpt < tab.length - tour; cpt++)
			if (tab[cpt] > tab[max])
max = cpt;

		permuter(tab, max, tab.length - tour - 1);
	}

}
```

2/triBulle

Dans le programme Java du tri à bulle on compare la case sélectionnée à la case suivante et on incrémente de 1 à chaque fois, lorsque la valeur prise est plus grande que la valeur suivante on échange les deux et on continue notre "chemin" ce ainsi de suite jusqu'à finir le tableau

```
public static void triBulle ( int[] tab )
{
        int valCaseA, valCaseB;
        int swap;


        swap=1;
        while (swap >=1)
        {
                swap=0;
                for(int cpt = 0; cpt+1 < tab.length; cpt++)
                {
                        if (tab[cpt] > tab[cpt+1])
                        {
                                swap ++;
                                permuter(tab, cpt, cpt+1);
                        }
                }
        }
}
```

Puis on répète l'opération jusqu'à ce que tout le tableau soit trié et que le programme passe d'un coup sans échanger 2 cases une fois (la valeur swap sera à 0 et la boucle sera finie

3/triInsertion

Le programme du tri par insertion "sépare" le tableau en deux entre la partie triée (qui est croissante) et la partie non triée (qui est mélangée comme son nom le fait penser)
On commence par sélectionner la première valeur de la partie non triée(donc tab[1] pas tab [0]) puis on la compare avec la dernière valeur de la partie triée, si la valeur précédente est supérieure on inverse les deux pui on teste entre la nouvelle position et la position qui précède celle ci (si il y a une telle valeur) ce jusqu'à ce qu'il n'y ai pas de valeur

précédente supérieure ou que l'on ai atteint l'indice 0, puis on continue à tester comme cela la partie mélangée jusqu'à avoir un tableau trié.

4/estTri

Ce programme Java permet de savoir si le tableau est dans l'ordre croissant.
Tout d'abord il regarde si la valeur précédente est supérieure au chiffre de la case suivante:

```
public static boolean estTrie ( int[] tab )
{
	boolean estTrie = true;

	int   valPrec = tab[0];
	for (int cpt = 1; cpt < tab.length; cpt++)
	{
		if ( valPrec > tab[cpt] )
			estTrie = false;

		valPrec = tab[cpt];
	}

	return estTrie;
}
```

Si aucune des valeurs est comptée comme inférieure à sa précédente, le programme renvoie vrai, sinon il renvoie faux.

C)S'assurer que les tris soient testés sur les mêmes bases

1/Tableau original (tabOG)

Nous créons un tableau au tout début avec générer un tableau :

Ce programme Java permet de construire un tableau avec des chiffres aléatoires.

```
public static int[] genererTableau ( int nbCases, int valMin, int valMax )
{
	int[] tab;
	tab = new int [nbCases];

	for(int cpt = 0; cpt<nbCases; cpt++)
		tab[cpt] = (int) (Math.random() * (valMax-valMin-1)) + valMin;

	return tab;
}
```

Nous ne toucherons pas à ce tableau entre chaque test. Nous copierons le contenu de ce tableau dans un autre tableau : tabUse.

2/copierTableau

Ce programme permet de faire une copie du tableau, nous permettons donc de garder le tableau original.

```
public static int[] copierTableau ( int[] tab )                    /* Cette méthode retourne un
nouveau tableau à l'identique du tableau passé en paramètre. */
	{
		int[] tabCopie;

		for(int cpt = 0; cpt <tab.length; cpt++)
		{
			tabCopie[cpt] = tab[cpt];
		}
		return tab;
	}
```

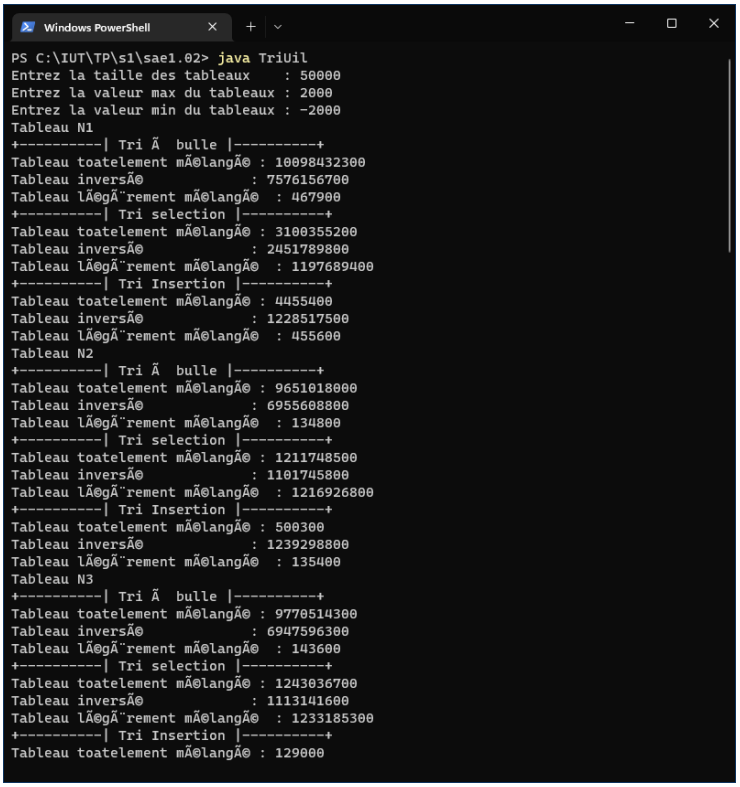
D)Exécution

Avant de faire quoi que ce soit, nous allons fermer tous les logiciels actuellement actifs et se mettre en mode avion. Cela nous permettra d’avoir des résultats plus fiables.
Nous avons aussi fait en sorte que notre programme fasse 10 fois le test, pour pouvoir faire la moyenne de ces résultats et donc de se rapprocher de la vraie valeur.

Ici, nous ferons le test sur un tableau de 50 000 case sur un intervalle de [-2000;2000]

Il est important de noter que ce que nous avons ici est le temps en nanoseconde que prend chaque trie à faire. Le tableau n'est volontairement pas affiché, car cela gêne la lecture de nos résultats.

Dans le programme original, il sera affiché. De plus, le programme s’arrête automatiquement s' il rencontre un problème dans les tries.



Suite à l'exécution de notre programme, et à la rentrée des différentes données, nous nous retrouverons avec plusieurs valeurs que nous avons entrez dans différents tableaux excel.

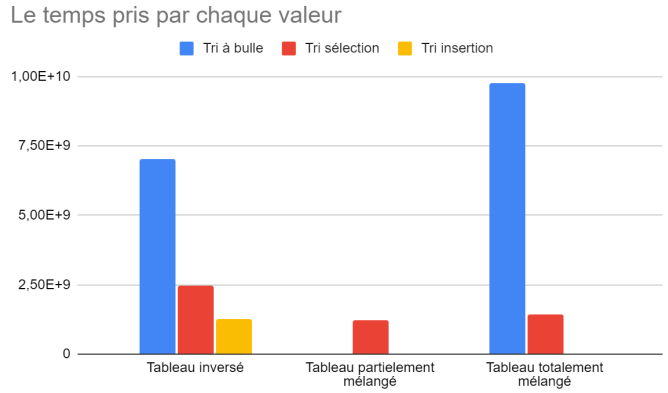
Voici les tableaux que nous avons :

Tableau totalement mélangé		
Tri à bulle	**Tri sélection**	**Tri insertion**
10098432300	3100355200	4455400
9651018000	1211748500	500300
9770514300	1243036700	129000
9774393600	1247710500	91200
9709860500	1221422700	90900
9700274600	1223361000	91500
9727091900	1239815800	127700
9669798500	1200925700	91200
9639508900	1239525500	127500
9695325500	1219954200	127400
Moyenne		
9743621810	1414785580	583210

Tableau partielement mélangé		
Tri à bulle	**Tri sélection**	**Tri insertion**
467900	1197689400	455600
134800	1216926800	135400
143600	1233185300	171700
141500	1214188200	127200
202000	1219787600	91400
100500	1217874900	127100
134900	1228753800	119700
135100	1211014400	91100
135000	1254547400	127200
96700	1230222500	134900
Moyenne		
169200	1222419030	158130

Tableau inversé		
Tri à bulle	**Tri sélection**	**Tri insertion**
7576156700	2451789800	1228517500
6955608800	2451789800	1239298800
6947596300	2451789800	1286166500
6986265100	2451789800	1258070900
6955965600	2451789800	1259056500
6921019700	2451789800	1261541100
6934179100	2451789800	1268637000
6936146600	2451789800	1271665500
6922694000	2451789800	1256738100
6947434700	2451789800	1268241600
Moyenne		
7008306660	2451789800	1259793350

En regardant les moyennes, nous nous rendons rapidement compte que le tri par insertion est le plus rapide dans tous les cas.



Le tri par insertion serait donc le plus optimisé.