SAE 1.02 - Comparaison Algorithmique

Projet n°2 Équipe n°21
Rendu n°2

Justine BONDU

Jules CHUZEVILLE
Alizéa LEBARON

Nous avons testé les tris, sélection, insertion et a bulle, sur toutes nos machines. Nous avons utilisé le même code à chaque fois. Le seul logiciel d'ouvert sur nos machines était Visual Studio Code c'est le logiciel que l'on a utilisé pour coder en java. Nous avons utilisé des tableaux de taille de 200 000 avec des valeurs entre 0 et 200 000.

---

tri par sélection

Temps en milliseconde pour trier un tableau désordonné :

Nom	Justine	Alizéa	Jules
Temps de tri	11061	15544	13763

Temps en milliseconde pour trier un tableau presque ordonné :

Nom	Justine	Alizéa	Jules
Temps de tri	11071	15562	13381

Temps en milliseconde pour trier un tableau ordonné :

Nom	Justine	Alizéa	Jules
Temps de tri	6352	9089	7943

---

tri à bulle

Temps en milliseconde pour trier un tableau désordonné :

Nom	Justine	Alizéa	Jules
Temps de tri	47726	61678	52910

Temps en milliseconde pour trier un tableau presque ordonné :

Nom	Justine	Alizéa	Jules
Temps de tri	13502	19339	16946

Temps en milliseconde pour trier un tableau ordonné :

Nom	Justine	Alizéa	Jules
Temps de tri	4839	9506	8379

---

tri par insertion

Temps en milliseconde pour trier un tableau désordonné :

Nom	Justine	Alizéa	Jules
Temps de tri	11607	12072	10417

Temps en milliseconde pour trier un tableau presque ordonné :

Nom	Justine	Alizéa	Jules
Temps de tri	32	1	2

Temps en milliseconde pour trier un tableau ordonné :

Nom	Justine	Alizéa	Jules
Temps de tri	0	2	1

Description des méthodes

Méthodes de tri

Pour trier les tableaux selon les différentes méthodes, nous avons programmé les méthodes de tri par insertion, par sélection et à bulle.
La méthode de tri par insertion compare un élément n et tous les éléments avant n et le place dès que cet élément est plus que l'élément de comparaison
La méthode de tri par sélection cherche la valeur la plus grande et la met au dernier indice et réduit la taille de la boucle de tri.
La méthode de tri à bulle compare l'élément n à l'élément n+1 pour toutes les cases du tableau et échange de place avec cet élément si il est plus grand, cette méthode s'effectue jusqu'à avoir un tableau complètement trié.

Méthode CopierTableau

La méthode copier tableau est assez simple, on prend une simple boucle et on copie un à 1 les valeurs dans un autre tableau.
De cette façon, on renvoie le tableau copié qui n'a pas la même adresse que le premier tableau. (On a vérifié que changer l'un n'impactent pas l'autre).

Méthode estTrier

La méthode estTrier permet de renvoyer un booléen en fonction de si le tableau est bien trié ou non. Elle consiste simplement à l'aide d'une boucle de vérifier que tab[cpt] est plus grand ou égal à table de cpt-1.

Méthode toString

La méthode toString renvoie les 4 premières cases et les 4 dernières cases du tableau. Pour cela on utilise deux boucles. Une première affichant les 4 premiers nombres avec un cpt allant de 0 à 4, et une deuxième (en dehors de la première) partant de tab.length et avec un cpt qui diminue pour afficher les 4 derniers.

Méthode genererTableau

La méthode de générer tableau renvoie un tableau de taille 200 000 non trié avec des valeurs aléatoires sur l'intervalle [0;199 999] inclus. La méthode commence par générer un tableau vide de 200 000 en taille, ensuite on parcourt les indice du tableau, a chaque indice on attribue un valeur prise au hasard sur l'intervalle [0;199 999] inclus à l'indice sélectionner.

Méthode permuter

La méthode permuter permet d’échanger deux valeur d’un même tableau à l'aide des indices, cette méthode ne renvoie rien. Pour cela on commence par vérifier que les deux valeurs à échanger ne sont pas identiques, ensuite on met la valeur du deuxième indice dans une variable temporaire puis on remplace la valeur du deuxième indice par la valeur du premier indice on termine par affecter au premier indice la valeur qui était dans la variable temporaire.

Le tri par insertion est environ 5 fois plus rapide que le tri à bulle et est environ 1.5 fois plus rapide que le tri par sélection sur un tableau désordonné d’après les graphiques ci-dessous
Il est également plus rapide sur un tableau presque ordonnée où il est quasiment instantané alors que les autres méthodes de tri sont largement plus longues.
Idem pour les tableaux ordonnés là où les autres méthodes de tris sont longues, le tri par insertion est quasiment instantané.

Tableau moyen des résultats

Temps par tri			
	Sélection	Bulle	Insertion
Non ordonné	13456,0	40578,5	11365,3
Ordonné	7794,7	7574,7	1,0
Presque Ordonné	13338	16595,7	11,7

Classement de rapidité

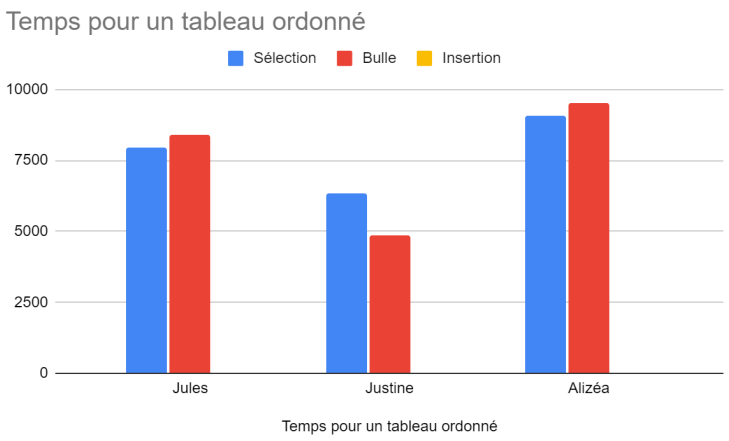
1. Insertion
2. Sélection
3. Bulle

× 1.5 plus rapide

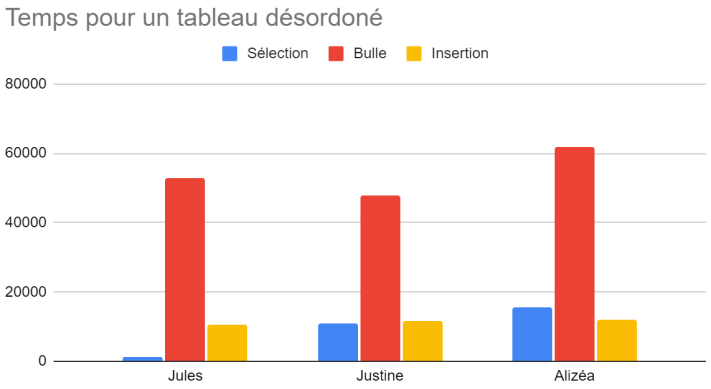
× 4 à 5 plus rapide

Quelques graphiques

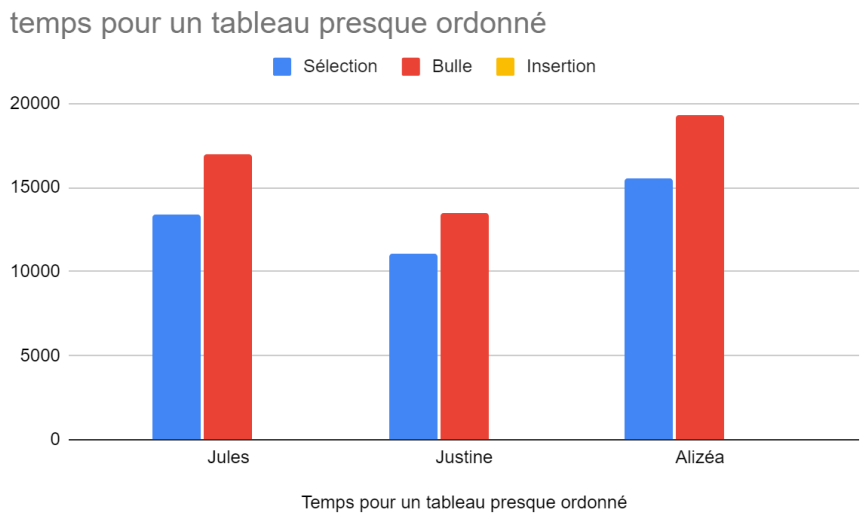
Quelques fois l'on ne voit pas l'insertion, c'est parce qu'il y a trop de différence entre son résultat et celui des autres.



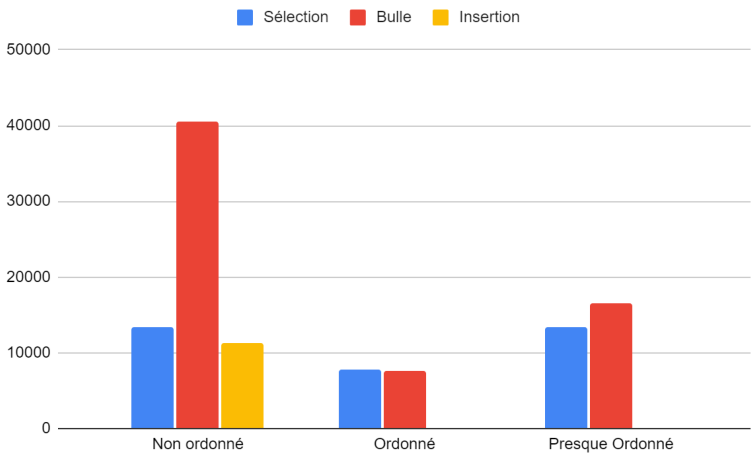
Temps pour un tableau ordonné



Temps pour un tableau non ordonné



Graphique des temps moyens de tous les tris



Avec le fait que le tri par insertion ne soit presque pas visible sur les graphiques, on peut en déduire que le tri par insertion est le plus rapide.

Le code de test pour un type de tri

```
public class test
{

    public static void main(String[] argv)
    {
        /*------------------*/
        /* Données          */
```

```java
        /*-----------------*/

        /* Variables       */
        /*-----------------*/

        int[]   tabEntier;
        String  res;

        long     tempsDeb;
        long     tempsFin;

        int     cpt;

        /*-----------------*/
        /* Instructions    */
        /*-----------------*/

        /*-------------------------*/
        /* Tableau non ordonné     */

        //Initialisation du tableau de 50000
        tabEntier = TriUtil.genererTableau();

        // Tri du tableau

        tempsDeb = System.currentTimeMillis();

        TriUtil.triBulle(tabEntier);

        tempsFin = System.currentTimeMillis();
        //Le tableau est-il bien trié ?

        System.out.println(TriUtil.estTrie(tabEntier));

        // Tableau en string afin d'avoir un apperçu
        res = TriUtil.toString(tabEntier);
        System.out.println(res);

        //Affichage du temps de tri

        System.out.println("Temps pour le tableau non ordonné : " +
(tempsFin-tempsDeb));
```

```
        /*-----------------------------*/
        /* Tableau presque ordonné    */

        //Initialisation du tableau de 100000

        cpt = 0;
        while (cpt < 10000)
        {
            tabEntier[cpt] += (int) (Math.random() * 100);

            cpt++;
        }

        // Tri du tableau

        tempsDeb = System.currentTimeMillis();

        TriUtil.triBulle(tabEntier);

        tempsFin = System.currentTimeMillis();

        //Le tableau est-il bien trié ?

        System.out.println(TriUtil.estTrie(tabEntier));

        // Tableau en string afin d'avoir un apperçu
        res = TriUtil.toString(tabEntier);
        System.out.println(res);

        //Affichage du temps de tri

        System.out.println("Temps pour le tableau presque ordonné : " +
(tempsFin-tempsDeb));

        /*--------------------*/
        /* Tableau ordonné    */

        // Tri du tableau

        tempsDeb = System.currentTimeMillis();

        TriUtil.triBulle(tabEntier);
```

```
        tempsFin = System.currentTimeMillis();

        //Le tableau est-il bien trié ?

        System.out.println(TriUtil.estTrie(tabEntier));

        // Tableau en string afin d'avoir un apperçu
        res = TriUtil.toString(tabEntier);
        System.out.println(res);

        //Affichage du temps de tri

        System.out.println("Temps pour le tableau ordonné : " +
(tempsFin-tempsDeb));
    }
}
```