SAE 1.2

Projet n°2 - Exercice n°2

Equipe n°18

DUNET	Tom	A1
ROCHE	Gabriel	A2
BUREAUX	Axel	A1
GOUREAU	Tom	A1

Conditions de test des tris

Les tests doivent être effectués sur la même machine afin d'obtenir des résultats homogènes (performances équivalentes).
De plus, il ne faut pas avoir de programmes autres que le terminal et activer le mode avion de son ordinateur pour ne pas recevoir de données extérieures pouvant fausser les résultats, afin que l'ordinateur ne soit occupé qu'à trier les valeurs (et donc qu'il soit à son niveau de performance optimal).

Données calculées et comparées

Afin de comparer les différents tris, on regardera le temps d'exécution.

On vérifie que le tableau sera bien trié en utilisant un sous-programme vérifiant la bonne croissance des éléments (celui-ci ne sera pas compté dans les calculs des données de comparaison).

Nous allons utiliser plusieurs tableaux…

- un tableau déjà trié
- un tableau pratiquement trié (100 valeurs non-triées)
- un tableau trié dans le sens inverse
- un tableau rempli aléatoirement (comme utilisé de base)

…afin d'obtenir des données supplémentaires dans des conditions différentes.

Afin d'obtenir des résultats plus cohérents et plus précis, on testera plusieurs fois les tris, tout en récupérant les données dans un tableau, puis nous ferons la moyenne de chaque tableau et tri.

Explication des fonctions

---

estTrie :

La fonction est Trié a pour but de détecter si un tableau entier (qu'elle prend en argument) est trié dans l'ordre croissant ou non. Pour se faire, elle compare chaque valeur du tableau (sauf la première) avec la précédente.
Si la valeur précédente est plus grande que celle sur laquelle on est, si c'est le cas, elle retourne FALSE. SI la condition n'est pas vérifiée, elle passe à la valeur suivante et si la condition n'est jamais vérifiée, elle retourne TRUE.

genererTableau :

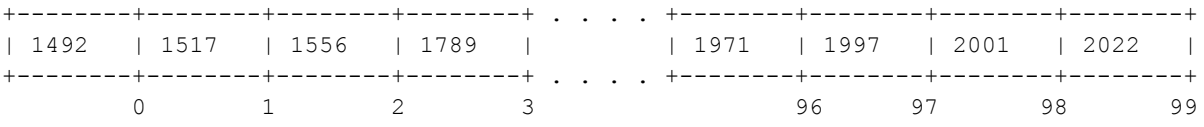
La fonction générer Tableau a pour but de créer un tableau de valeurs aléatoires entières. Elle prend en argument 3 entiers : un nombre de cases, la valeur minimale présente dans le tableau et la valeur maximale présente dans le tableau.
La fonction commence par instancier un tableau du nombre de case passer en argument et rempli ensuite chaque case avec une valeur aléatoire comprise entre les deux bornes définis en argument et finit par retourner le tableau.

copierTableau :

La fonction copier Tableau a pour but de copier un tableau d'entier. Elle prend en argument un tableau d'entier et retourne un autre tableau d'entier. Cette fonction copie chaque valeur dans un autre tableau instancier à la même taille.
On ne renvoie pas pas simplement le tableau donné en argument car sinon, on recopie l'adresse mémoire et ce qui nous intéresse, c'est d'avoir un tableau ayant les mêmes valeurs.

toString :

La fonction toString a pour but d'afficher notre tableau de façon lisible et non toutes les valeurs. Pour cela, on affiche uniquement les 4 premières et les 4 dernières valeurs sous la forme :



`permuter :`

La fonction permuter a pour but d'inverser deux valeurs dans un tableau d'entier, elle prend en argument un tableau d'entier et deux indices, qui représentent les deux valeurs a inverser. Pour inverser les deux valeurs, elle utilise une variable temporaire.

Résultats obtenus après tests

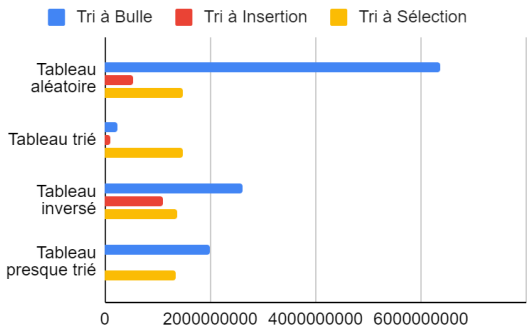
Après tout les tests effectués, on obtient divers résultats nous permettant d'obtenir le tableau suivant :

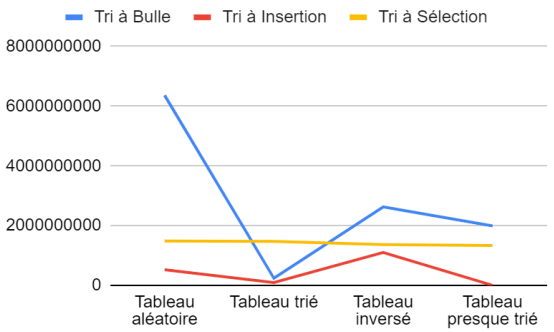
	Tri à Bulle	Tri à Insertion	Tri à Sélection
Tableau aléatoire	6358212273	529410072,7	1486109845
Tableau trié	240319036,4	98150709,09	1474784691
Tableau inversé	2627148691	1103033236	1368364000
Tableau presque trié	1991650136	4006372,727	1336232973

Ce tableau affiche la **moyenne des durées obtenues** (en nanosecondes) pour chaque tableau et chaque type de tri.

Nous pouvons obtenir différents graphiques nous permettant d'apercevoir des différences significatives :

Premièrement, nous étudions les différences de temps des tris en **fonction de chaque tri**.

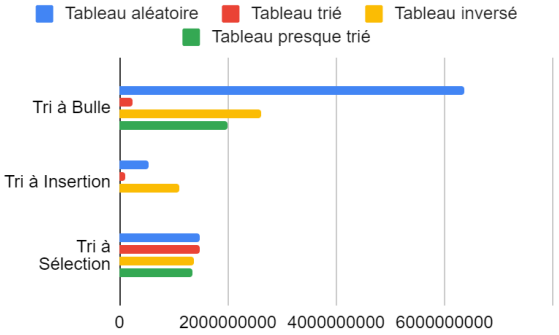


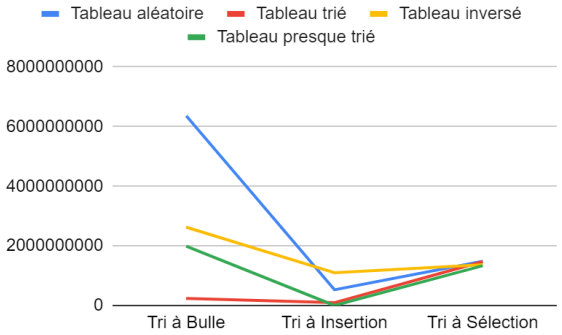


Ces graphiques nous apprennent que le tri à **Insertion** est **meilleur** dans tout les domaines, que le tri à **Sélection** est **constant** dans ses valeurs (peut importe le type de tableau) et que le tri a **Bulle** est globalement **très lent,** sauf dans le cas où le tableau serait déjà trié (explicable par la méthode du tri a Bulle, ne remontant aucune valeur car déjà trié).

Le tri à Insertion serait donc la méthode de tri la plus optimisée dans un cas général !

Ensuite, étudions les différences de temps entre les différents tris, en fonction du **type de tableau utilisé**.





Ces graphiques nous apprennent que le tri à **Bulle** reste la méthode la **moins optimisée**, à part dans le cas d'un tableau trié par avance.
Le tri à **Insertion** est, globalement, la méthode la **plus rapide** pour tous les types de tableaux (et possède même un temps de tri du tableau presque trié si petit qu'il ne figure pas dans le graphique !)
Concernant le tri a Selection, celui-ci possède des temps moyens et plutôt constants. Utiliser un tableau déjà trié ne rend pas la méthode plus rapide.

En conclusion, le tri à **Sélection** est la méthode de tri la **plus rapide et optimisée** pour du triage de tableaux, peu importe le type du tableau utilisé (aléatoire, trié/presque trié, inversé).