Analyse des Algorithmes de Tri

HANNIZ Noureddin Anis
L2 Informatique Groupe 2

1. Le Projet

Ayant reçu le projet sur la complexité des algorithmes de tri. Dans ce compte rendu nous allons comparer 5 implémentations de tri :

- *Deux versions du Tri à Bulles*
- *Tri par Sélection*
- *Tri Rapide*
- *Tri par Dénombrement*

Ces tris s'effectueront sur des tableaux d'éléments. Les éléments comportent un champ clé (entier positif ou nul) et un champ contenu .

Vous trouverez en annexe toutes les implémentations de ces derniers en fichier Java.

Dans un premier temps, nous allons les présenter et expliquer brièvement leurs principes. Ensuite, nous les comparerons grâce à un comptage des opérations élémentaires, ainsi que par la récupération du temps d'exécution qui sera représenté sur un graphique.

2. Présentation des Algorithmes & Principes

Tri à Bulles

Le tri à bulle est un algorithme qui consiste à comparer deux éléments consécutifs d'un tableau et à les échanger si nécessaire.
Cette opération est répétée jusqu'à ce que le tableau soit trié.

Deux versions

- V1 : diminution à chaque itération de la taille des sous tableaux à trier.
- V2 : détection que le tri est terminé.

Tri par Sélection

Le tri par sélection est un algorithme de tri qui consiste à trouver le plus petit élément d'un tableau et à le placer en première position.

Ensuite, on cherche le plus petit élément du tableau restant et on le place en deuxième position. On continue ainsi jusqu'à ce que le tableau soit trié.

Principes

- Recherche du plus petit élément du tableau
- Déplacement de cet élément en début
- Trier le reste du tableau

Tri Rapide

Le tri rapide est un algorithme de tri qui utilise la méthode de la division et de la conquête.

Il consiste à sélectionner un élément du tableau, appelé pivot, et à réorganiser le tableau de telle sorte que tous les éléments inférieurs au pivot soient placés avant ce dernier et que tous les éléments supérieurs au pivot soient placés après ce dernier.

Principe :

- Choix d'un élément pivot
- Partage le tableau en deux sous tableaux de tailles identiques contenant :
- Les éléments inférieurs au pivot
- Les éléments supérieurs au pivot
- Trie les deux sous tableaux

Tri par Dénombrement

Le tri par dénombrement est un algorithme de tri qui fonctionne en deux étapes. Dans la première étape, on compte le nombre d'occurrences de chaque élément dans le tableau. Dans la deuxième étape, on reconstruit le tableau en parcourant le tableau de comptage et en plaçant chaque élément le nombre de fois indiqué par le tableau de comptage.

Principes

- N'est pas un tri par comparaisons
- Utilise un tableau intermédiaire de taille "valeur maximum"
- Utilise un deuxième tableau intermédiaire comme accumulateur

3.Comparaison des Algorithmes

Comptage des Opérations élémentaires

J'ai d'abord considéré ceci :

- « n » la taille du tableau
- Une affectation = 1
- Une incrémentation = 1
- Return = 1
- Une comparaison/test = 1
- Opération arithmétique = 1
- Appel de fonction = 1 + le coût de ses opérations
- Écriture/lecture d'une donnée = 1
- Donnée en entrée/paramètre d'une fonction = 1

En comptant le nombre d'opérations élémentaires des fonctions de tri (présentes sur la classe Java triTab), nous obtenons les résultats suivants:

Algorithmes	tri à bulles V1	tri à bulles V2	Sélection	tri Rapide	Dénombrement
NbrOE	15	16	12	40	30

Je me suis rendu compte ensuite qu'il est mentionné sur le sujet du projet qu'il fallait compter que : « les comparaisons de clés, affectations (clés ou éléments) »

Ainsi, j'ai tout refait et nous obtenons ces résultats

Algorithmes	tri à bulles V1	tri à bulles V2	Sélection	tri Rapide	Dénombrement
NbrOE	8	11	6	24	14

Complexité

- **Tri Bulle v1**

  Au pire => Complexité en $O(n^2)$

- **Tri Bulle v2**

  Au pire => Complexité en $O(n^2)$

- **Tri par Sélection**

  Au pire : chaque élément est placé à la fin => Complexité en $O(n^2)$

- **Tri Rapide**

  En moyenne/Au mieux => Complexité en O(n.log2(n))

  Au pire => Complexité en $O(n^2)$

- **Tri par Dénombrement**

  Au pire => Complexité en O(n)

Temps d'exécution

La fonction «System.currentTimeMillis()» de Java, nous permettra de déduire le temps nécessaire pour exécuter toutes les fonctions de tri citées précédemment.

Cependant, il faut noter que ces résultats dépendent de plusieurs facteurs, tels que la taille du tableau, le nombre d'éléments désordonnés, le matériel et le système d'exploitation sur lequel le programme s'exécute, ainsi que de la mise en œuvre spécifique de chaque version. *(Nous reverrons tout ceci en Conclusion)*

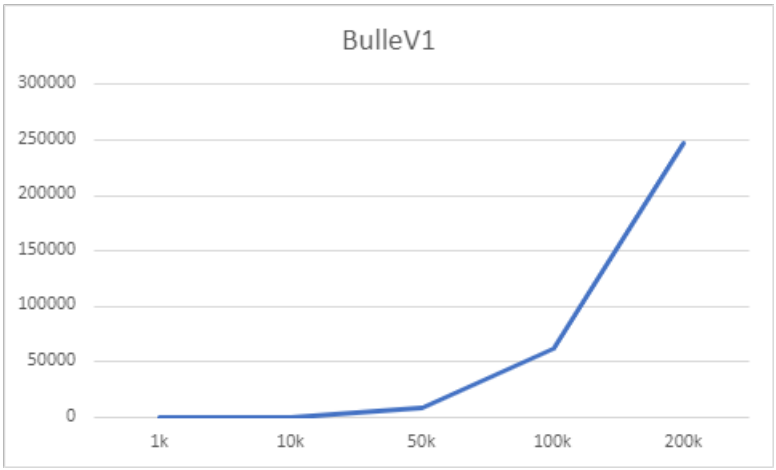
Nous avons aussi pris le temps de faire un constructeur qui crée des tableaux d'éléments par copie afin d'effectuer ces tris sur les mêmes données.

Nous avons obtenu ces résultats :

- *Soit « n » la taille du tableau*

Tri à bulle V1

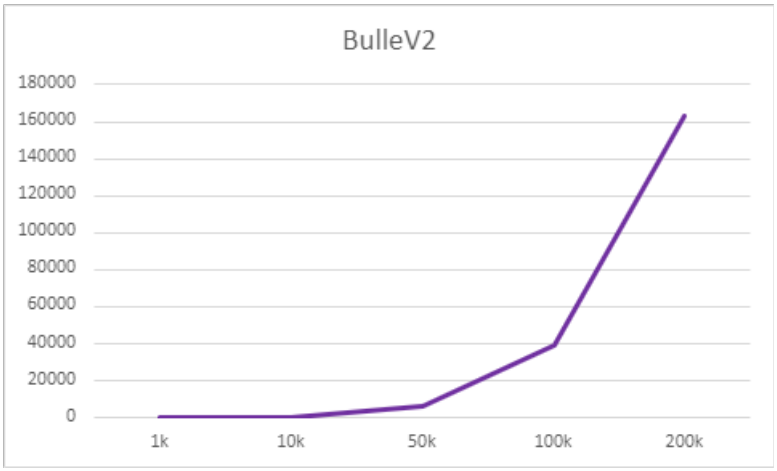
Pour : n=1K : 19ms
n=10K : 518 ms
n=50K : 8877ms
n=100K : 61 686ms
n=200K : 247882ms



Le temps d'exécution augmente très rapidement avec la taille du tableau à trier. Pour un tableau de 200 000 éléments, il faut 247 882 ms, soit plus de 4 minutes.

Tri à bulle V2

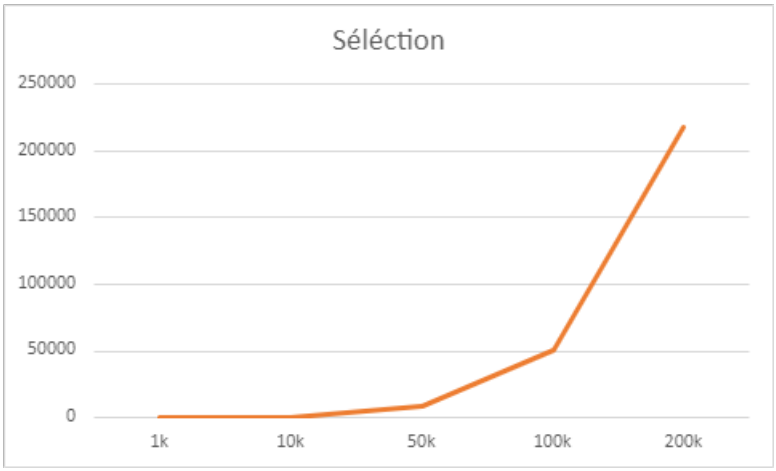
Pour : n=1K : 15ms
n=10K : 298 ms
n=50K : 5706ms
n=100K : 39 269ms
n=200K : 162737ms



Le temps d'exécution est nettement amélioré par rapport à la version 1, mais reste assez long pour des tableaux de grande taille.

Tri par Sélection

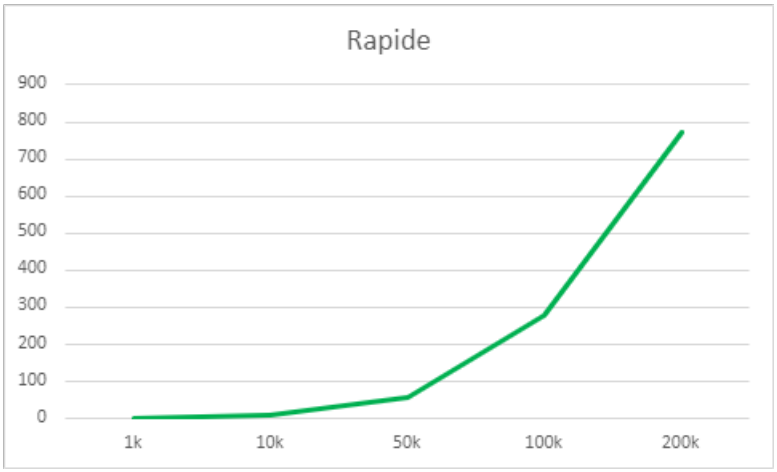
Pour : n=1K : 37ms
n=10K : 446ms
n=50K : 8134ms
n=100K : 50233ms
n=200K : 218153ms



Le temps d'exécution est plus court que le tri à bulle V1, mais encore assez long pour des tableaux de grande taille.

Tri Rapide

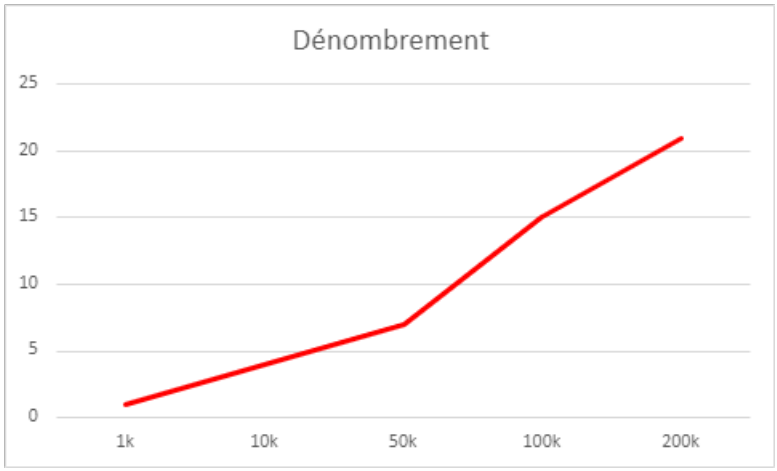
Pour : n = 1K : 2ms
n=10K : 10ms
n=50K : 57ms
n=100K : 276 ms
n=200K : 772ms



C'est l'algorithme de tri le plus rapide parmi ceux proposés plus haut. Il est nettement plus rapide que les autres pour des tableaux de grande taille.

Tri par Dénombrement

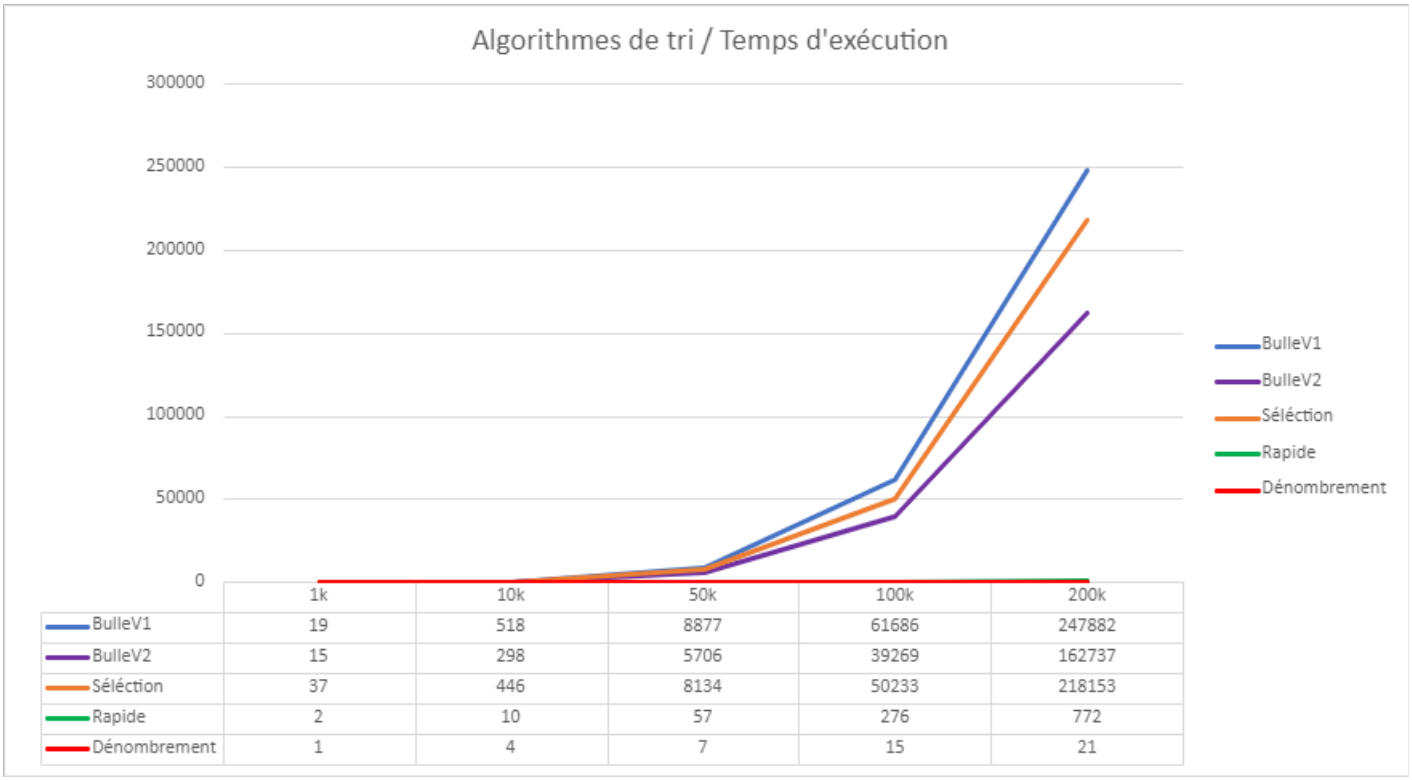
Pour : n = 1K : 1ms
n = 10K : 4ms
n = 50K : 7ms
n= 100K :15ms
n= 200K : 21ms



Avec des temps d'exécution compris entre 1ms pour une taille de 1 000 éléments et 21ms pour une taille de 200 000 éléments, le tri par dénombrement est en effet le plus rapide parmi tous les algorithmes de tri présentés.

4.Bilan :

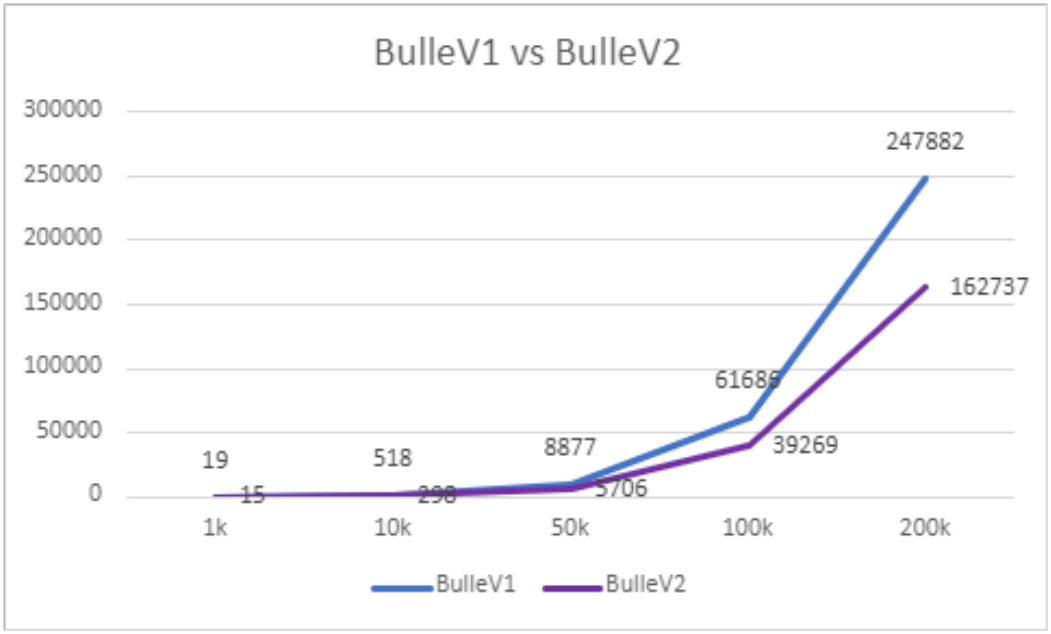
Graphe Global



	1k	10k	50k	100k	200k
BulleV1	19	518	8877	61686	247882
BulleV2	15	298	5706	39269	162737
Séléction	37	446	8134	50233	218153
Rapide	2	10	57	276	772
Dénombrement	1	4	7	15	21

- Il est clair que le Tri Rapide et le Tri par Dénombrement sont les plus performants.
- Le Tri à Bulle V1 est le moins performant, prenant plusieurs minutes pour trier un tableau de 200K éléments. Le Tri par Sélection et Tri à Bulle V2 sont également relativement lents pour des tailles de données importantes.
- Le Tri par Dénombrement est de loin le plus rapide pour trier des tableaux d'entiers de toutes tailles, avec des temps d'exécution de seulement quelques millisecondes pour des tableaux de 200K éléments.
- Le Tri Rapide est également très performant, même pour des tailles de données importantes, avec un temps d'exécution d'environ 700ms pour trier un tableau de 200K éléments.

Tri BulleV1 vs Tri BulleV2

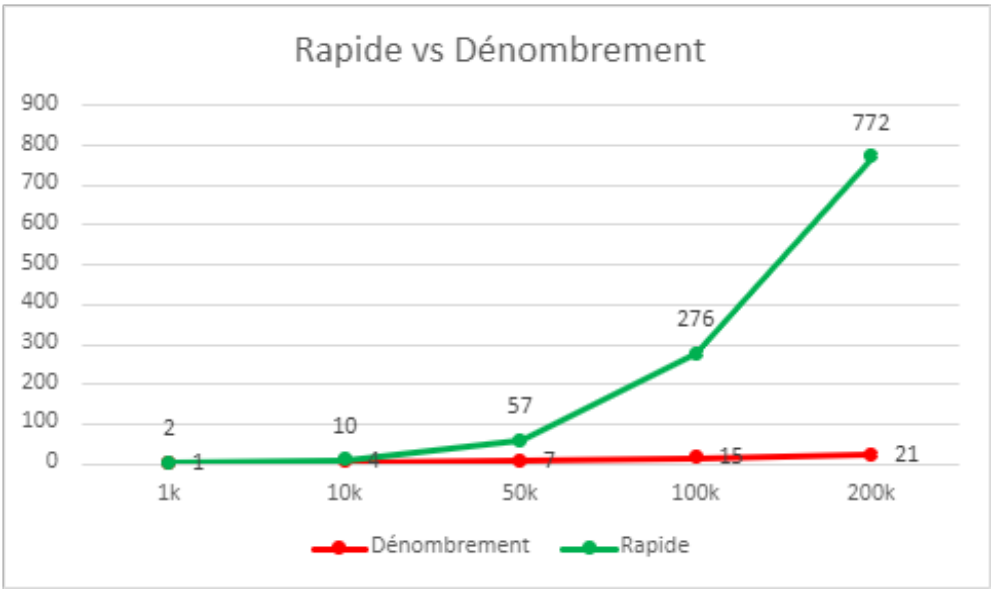


La version 2 du tri à bulles est plus rapide que la version 1, en particulier pour les tableaux de grande taille. Les optimisations réduisent le nombre d'itérations nécessaires pour effectuer le tri, ce qui peut améliorer les performances.

Cependant, pour de petits tableaux ou pour des éléments déjà presque triés, les différences de temps d'exécution entre les deux versions sont minimes.

En somme, la version 2 du tri à bulles est plus efficace en termes de temps d'exécution.

Tri Rapide vs Tri par Dénombrement



- Le tri par dénombrement est un algorithme de complexité linéaire, c'est-à-dire que le temps d'exécution est proportionnel à la taille du tableau à trier.
- Le tri rapide est un algorithme de complexité quadratique, c'est-à-dire que le temps d'exécution est proportionnel au carré de la taille du tableau à trier.

Cela signifie que le tri par dénombrement est plus rapide que le tri rapide.

Cependant, le tri par dénombrement n'est pas adapté à tous les cas.

Par exemple, si les éléments du tableau sont des nombres réels, il faut d'abord les transformer en entiers.

Cette transformation peut être très coûteuse en temps d'exécution.

Dans notre cas, le champ « clé » des éléments utilisés sont est de type entier.

5.Conclusion

Les résultats obtenus illustrent l'importance de choisir le bon algorithme de tri en fonction du problème à résoudre et de la taille des données à trier. Dans la plupart des cas, le Tri Rapide et le Tri par Dénombrement sont des choix sûrs pour obtenir des performances optimales.

Le Tri par Dénombrement est particulièrement efficace pour trier des tableaux d'entiers dont les valeurs sont uniformément distribuées, alors que le Tri Rapide est plus adapté à des tableaux avec des éléments répartis de manière plus aléatoire.

Les temps d'exécution dépendent également de la puissance de l'ordinateur sur lequel l'algorithme est exécuté. Les résultats obtenus ici ont été mesurés sur un ordinateur particulier et pourraient varier sur d'autres machines.

La performance d'un algorithme de tri ne doit pas être considérée de manière isolée, mais plutôt dans le contexte de l'ensemble de l'application. Par exemple, le Tri par Dénombrement peut être plus rapide que le Tri Rapide pour trier des tableaux d'entiers, mais il peut également nécessiter plus de mémoire pour stocker les tableaux intermédiaires, ce qui peut être un facteur limitant pour certaines applications.

En résumé, le choix de l'algorithme de tri dépendra des caractéristiques des données à trier, des contraintes de performance et de mémoire de l'application, ainsi que de la puissance de l'ordinateur sur lequel l'algorithme est exécuté.