ECOLE NORMALE SUPÉRIEURE DE L'ENSEIGNEMENT
TECHNIQUE DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

Rapport mini projet en langage C

Etude de la performance des algorithmes de tri

Réalisé par :

Chaymae AIT BENALLA

Khadija AMECHATE

Wijdane HACHANI

Mariam SATI

Encadré par :

Pr. Mohamed QBADOU

Filière :

II-Cybersécurité et Confiance Numérique

Année scolaire :2023/2024

Sommaire

Table des matières

I. Introduction

Les tableaux présentent un élément fondamental en programmation, car ils permettent de stocker et de manipuler des collections de données de manière efficace. En langage C, les tableaux sont des structures de données essentielles qui offrent une grande flexibilité pour stocker des éléments de même type. L'une des opérations les plus courantes que l'on effectue sur les tableaux est le tri. Le tri d'un tableau consiste à réorganiser ses éléments dans un ordre particulier, généralement croissant ou décroissant, pour faciliter la recherche et l'analyse de données.

Le tri de tableaux en C est une tâche courante et essentielle pour de nombreuses applications. Il existe plusieurs algorithmes de tri efficaces, chacun adapté à des besoins spécifiques en termes de performances et de contraintes. Parmi les algorithmes de tri les plus couramment utilisés, on trouve le tri à bulles, le tri par insertion, le tri par sélection, le tri rapide (quicksort), et le tri fusion (merge sort), entre autres. Chacun de ces algorithmes présente des avantages et des inconvénients, et le choix de l'algorithme dépend souvent du contexte d'application, de la taille du tableau et des performances requises.

Dans le rapport en question on va s’intéresser à l’étude et l’analyse de la performance de plusieurs algorithmes de tri. Afin de réussir cette étude, on a pu réaliser des présentations graphiques explicatives par le bais du langage C et l’outil GNUPLOT.

II. Algorithmes de tri

1)Tri a bulle :

Le tri à bulles, également connu sous le nom de "Bubble Sort" en anglais, est un algorithme de tri simple utilisé pour trier des éléments dans un tableau ou une liste. Il tire son nom du comportement des éléments à trier, qui "remontent" ou "descendent" à leur position correcte, comme des bulles d'air remontant à la surface dans un liquide.

L'idée fondamentale derrière le tri à bulles est de comparer deux éléments adjacents dans le tableau et de les échanger s'ils sont dans le mauvais ordre. Ce processus de comparaison et d'échange est répété pour chaque paire d'éléments adjacents dans le tableau, de gauche à droite, jusqu'à ce que le tableau soit entièrement trié.

2)Tri Rapide

Le tri rapide, également connu sous le nom de "quicksort" en anglais, est un algorithme de tri très efficace. Il a été développé par Tony Hoare en 1960 et est largement utilisé dans la pratique pour trier des tableaux en raison de sa rapidité et de ses bonnes performances en moyenne. Le tri rapide est un exemple d'algorithme de tri "diviser pour régner" qui divise le tableau en sous-tableaux plus petits, les trie, puis les fusionne pour obtenir un tableau trié.

3)Tri par insertion

Le tri par insertion, également connu sous le nom de "insertion sort" en anglais, est un algorithme de tri simple et intuitif. Il est efficace pour trier de petites quantités de données ou lorsque la plupart des éléments sont déjà triés. Le tri par insertion fonctionne en construisant un tableau trié un élément à la fois en insérant chaque élément non trié à sa place correcte dans le tableau trié.

4)Tri par fusion

Le tri fusion, également connu sous le nom de "merge sort" en anglais, est un algorithme de tri très efficace qui suit le paradigme "diviser pour régner". Il est connu pour sa stabilité, sa performance en temps constant et son efficacité, ce qui en fait un choix populaire pour trier de grandes quantités de données.

5)Tri par sélection

Le tri par sélection, également connu sous le nom de "selection sort" en anglais, est un algorithme de tri simple qui fonctionne en sélectionnant l'élément le plus petit (ou le plus grand, selon le besoin) du tableau non trié à chaque itération, puis en l'échangeant avec l'élément à sa position correcte dans le tableau trié.

6)Tri par tas

Le tri par tas, également connu sous le nom de "Heap Sort" en anglais, est un algorithme de tri basé sur une structure de données appelée un "tas" ou "heap". Il s'agit d'un algorithme de tri efficace avec une complexité en temps de O(n log n) dans le pire des cas, ce qui en fait un choix solide pour trier de grandes quantités de données.

7)Tri Shell

Le tri Shell, également connu sous le nom de tri par insertion à intervalles, est un algorithme de tri qui vise à améliorer la performance du tri par insertion, notamment lorsque la liste à trier est grande. Cet algorithme a été proposé par Donald L. Shell en 1959.

L'idée de base derrière le tri Shell est de diviser la liste en plusieurs sous-listes, puis de trier ces sous-listes à l'aide du tri par insertion. À mesure que le tri progresse, les sous-listes deviennent de plus en plus petites, jusqu'à ce que finalement, toute la liste soit triée. Les intervalles entre les éléments à trier diminuent progressivement, ce qui permet de réorganiser les éléments de manière plus efficace que le tri par insertion traditionnel.

III. Outils de réalisation

1)Langage C

Le langage C'est un langage de programmation de haut niveau qui a été développé au laboratoire Bell de l'AT&T à la fin des années 1960 par Dennis Ritchie. Il est réputé pour sa simplicité, son efficacité et sa portabilité, ce qui en fait l'un des langages de programmation les plus influents et largement utilisés dans le domaine de l'informatique. Le langage C se caractérise par sa syntaxe concise et expressive, sa capacité à offrir un contrôle direct sur la mémoire de l'ordinateur, ainsi que sa portabilité, ce qui signifie que le code C peut être écrit une fois et exécuté sur différentes plates-formes sans nécessiter de modifications majeures. Il est largement utilisé dans le développement de systèmes d'exploitation, de logiciels embarqués, de pilotes de périphériques, de jeux, de logiciels d'entreprise et bien d'autres domaines. Le langage C est accompagné d'une bibliothèque standard riche qui contient des fonctions pour effectuer de nombreuses tâches courantes, ce qui en facilite l'utilisation et la programmation modulaire.

2)GNUPLOT

GNUplot est un logiciel open source de création de graphiques et de traçage de graphiques en ligne de commande largement utilisé pour visualiser et représenter graphiquement des données. Il offre un contrôle complet sur l'apparence des graphiques et prend en charge divers types de graphiques en 2D et 3D. GNUplot est

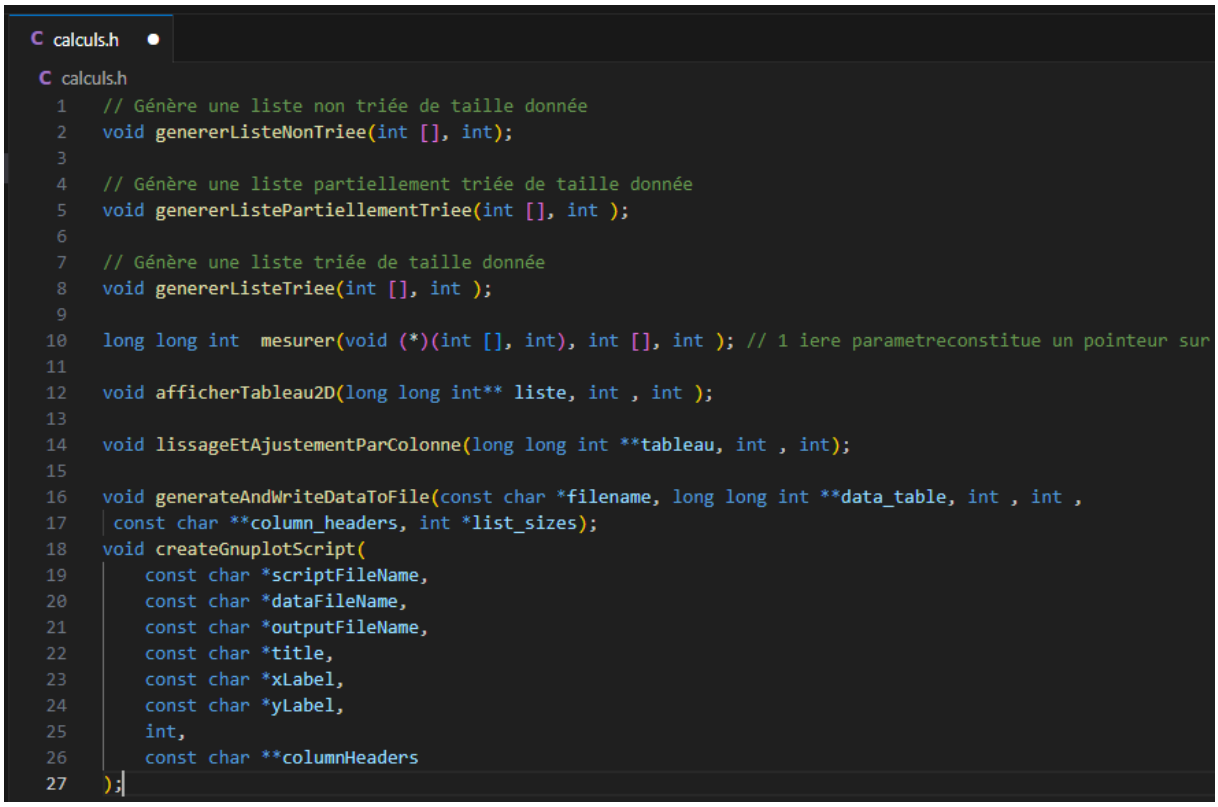
utilisé dans divers domaines, notamment l'analyse de données, la recherche scientifique et la génération de graphiques pour la visualisation de données.

3)Environnement de travail VS code :

Visual Studio Code est un éditeur de code simplifié, qui est gratuit et développé en open source par Microsoft. Il fonctionne sous Windows, mac OS et Linux. Il fournit aux développeurs à la fois un environnement de développement intégré avec des outils permettant de faire avancer les projets techniques, de l'édition, à la construction, jusqu'au débogage.

4)Principales fonctions

Afin de réaliser le projet en question, on a créé plusieurs fonctions à savoir :



- genererListeNomTriee : la fonction qui permet de générer une liste non triée pour obtenir son temps d'exécution.
- genererListePartiellementTriee : permet de générer une liste partiellement triée.
- genererListeTriee : génère une liste triée.
- mesurer : c'est la fonction qui a le rôle de calculer le temps d'exécution de chaque algorithme de tri afin de réaliser les courbes.
- afficherTableau2D : juste pour afficher un tableau de deux dimensions.

- lissageETAjustementParColonne : permet d'ajuster les valeurs afin d'obtenir une courbe ajustée.
- generateAndWriteDataToFile : pour extraire les valeurs d'exécution et les mettre dans un fichier texte. Voilà ce fichier-là :

List Size	triBulle		triSelection		triInsertion		triRapide	triFusion	triTas	triShell
9	600	600	400	400	800	700	500			
12	900	800	400	700	1200	900	600			
15	1300	900	400	800	1200	1000	600			
18	1700	1200	700	1100	1400	1100	1000			
21	1700	1500	700	1100	1700	1400	1100			
24	2700	1700	700	1400	2000	1800	1200			
27	3500	2200	1000	1500	2000	1800	1600			
30	3600	2600	1000	1900	2500	2100	2000			
33	5100	3300	1400	2000	3000	2500	2400			
36	5500	3500	1600	2400	3000	2500	2600			
39	5800	4100	1700	2600	3500	3100	2600			
42	6600	4600	2000	2600	4000	3300	2800			
45	8600	5200	2300	3200	4000	3800	3200			
48	9200	5800	2500	3200	4000	3800	6600			
51	10400	6300	2900	3600	4400	4100	6600			
54	12100	7200	3400	3700	5200	4800	6600			
57	13100	8000	3600	4200	5200	4900	6600			
60	16100	8800	4200	4800	6000	4900	6600			
63	16700	9600	4500	4800	8600	5800	6600			
66	17300	10000	4700	5100	8600	5800	6600			
69	25500	11000	5300	5100	8600	6000	6600			
72	25500	11800	5800	5600	8600	6100	6600			
75	26200	12500	6500	5600	8600	6500	6600			
78	26200	13500	6900	5900	8600	6500	7200			
81	26200	14500	7700	5900	8600	6900	7300			
84	26200	15400	7700	6600	11200	7600	8000			
87	36200	16900	8500	7100	11200	8100	8000			
90	36200	17500	8800	7100	11200	8100	8000			
93	57500	17500	8800	7100	11200	8100	8000			
96	57500	17500	8800	7100	11200	8100	8000			
99	57500	18400	8900	7100	11200	8200	11000			
102	57500	18900	9300	7400	11200	8400	11000			
105	57500	19900	10000	7400	11200	8900	11000			
108	57500	22000	10800	7700	12400	9000	11000			
111	57500	22800	11500	7900	12400	9100	11000			
114	57500	23200	12500	8000	12400	11800	11000			
117	57500	24600	12900	8000	12400	11800	11000			
120	57500	25400	12900	8700	12400	11800	11800			
123	57500	26400	13400	8700	12400	11800	11800			

- CreateGnuplotScript : est utilisée pour générer un script pour GNUPLOT.

En plus des fonctions précédentes, nous avons créé les fonctions suivantes permettant le tri d'un tableau avec différents algorithmes :

methodesTri.h

```c
void echanger(int [], int , int);
int partition(int [], int , int ) ;
void triRapideRecursif(int [], int , int ) ;
void triRapide(int [], int );

//tri Fusion
void fusion(int [], int , int , int );
void triFusionRecursif(int [], int , int );
void triFusion(int [], int );

//tri par tas
void entasser(int [], int , int );
void triTas(int [], int );

// Tri Shell
void triShell(int [], int);

// Déclarer une structure pour le tableau de pointeurs de fonctions
typedef struct {
    void (*functions[7])(int *, int);
} SortFunctions;

// Déclarer une instance de la structure
SortFunctions sort_functions;
```

IV. Visualisation

Apres l'exécution du programme en demandant la saisie de la taille du tableau, nombre de méthodes et le pas avec lequel on veut tracer la courbe :

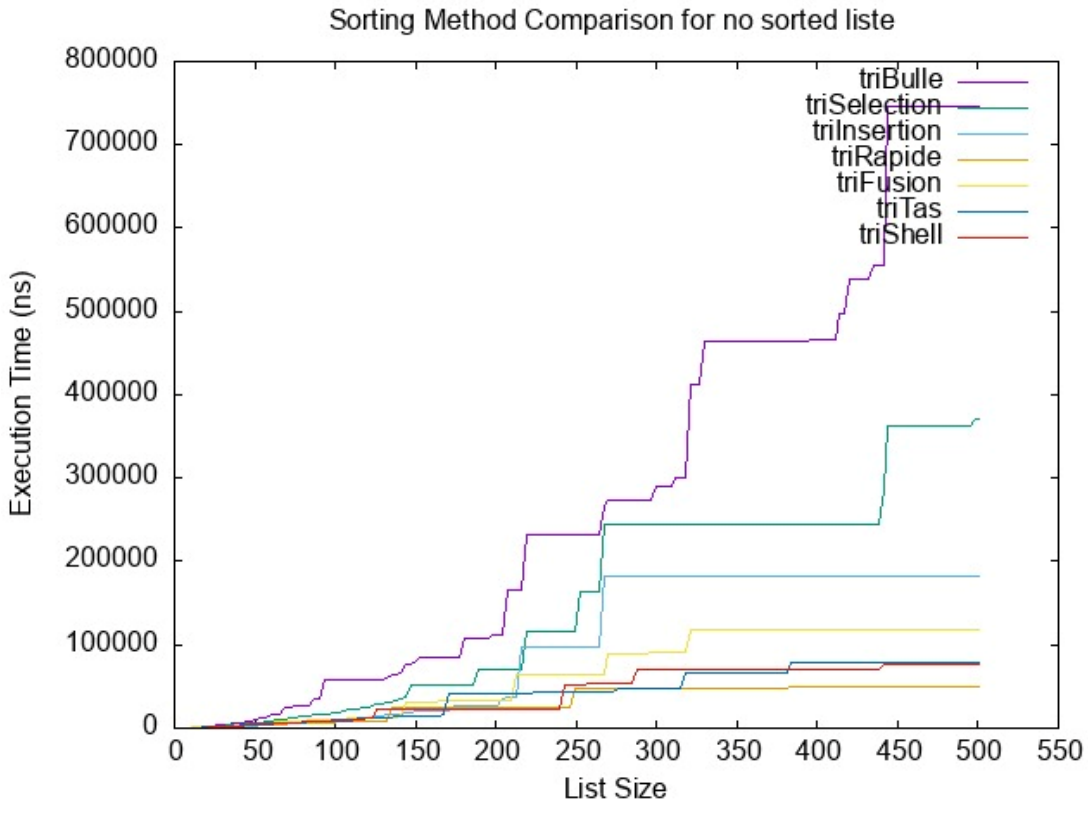
```
\maron\OneDrive\Desktop\ENSETM (EPM)  II CCN 2023-2026\S1\Algorithmique & programmation C\cours EXEs\08_mini-projet Mine> gcc -o mon_programme .\calculs.c .\methodesTri.c .\
ion.c
\maron\OneDrive\Desktop\ENSETM (EPM)  II CCN 2023-2026\S1\Algorithmique & programmation C\cours EXEs\08_mini-projet Mine> .\mon_programme
ts, sachant que le Max est 10 000 : 500
ethds, sachant que le Max est 7  :7
as, sachant que 0<pas<=pts  :3
```

Trois images ont été généré dans notre projet :

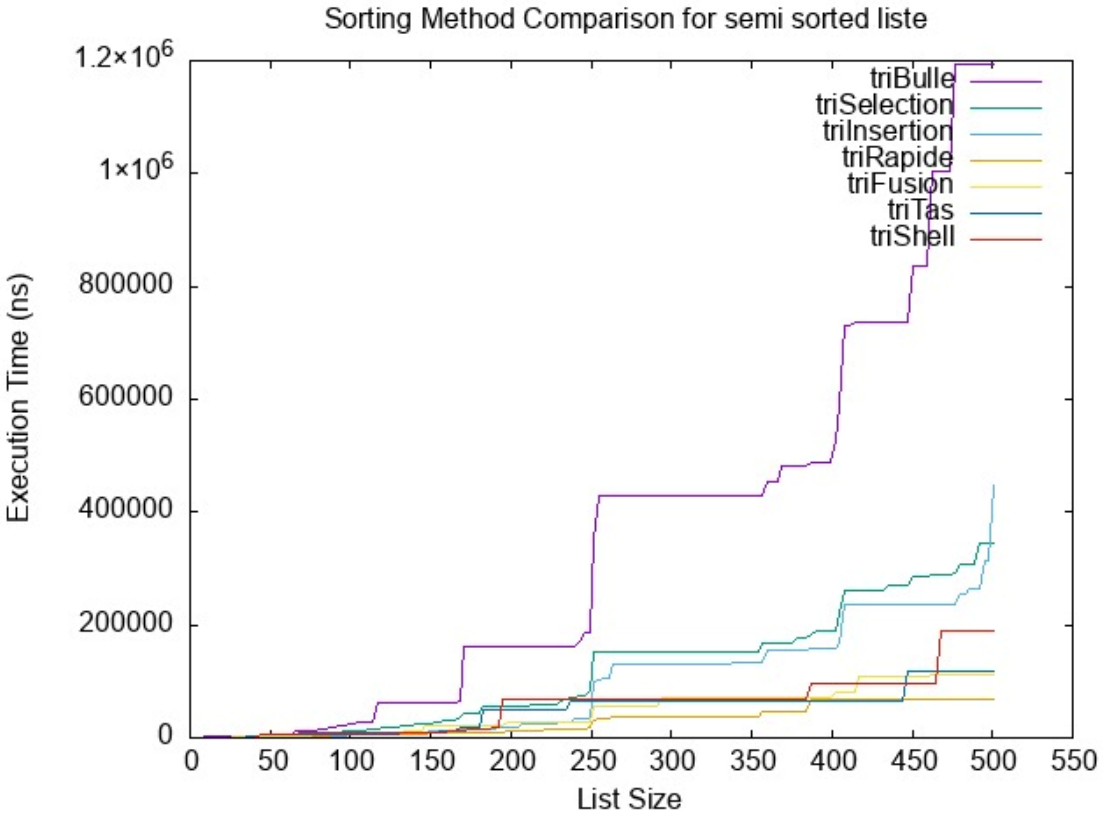
08_application	✓	07/11/2023 15:50	C Source File	7 KB
calculs	✓	07/11/2023 15:21	C Source File	5 KB
calculs	✓	07/11/2023 15:02	C Header Source F...	2 KB
data1	✓	07/11/2023 15:50	Document texte	8 KB
data2	✓	07/11/2023 15:50	Document texte	8 KB
data3	✓	07/11/2023 15:50	Document texte	8 KB
methodesTri	✓	01/11/2023 23:06	C Source File	6 KB
methodesTri	✓	01/11/2023 22:09	C Header Source F...	1 KB
mon_programme	✓	07/11/2023 15:50	Application	199 KB
output1	✓	07/11/2023 15:56	PNG File	8 KB
output2	✓	07/11/2023 15:50	PNG File	9 KB
output3	✓	07/11/2023 15:56	PNG File	9 KB
plotScript1	✓	07/11/2023 15:50	gnuplot command...	1 KB
plotScript2	✓	07/11/2023 15:50	gnuplot command...	1 KB
plotScript3	✓	07/11/2023 15:50	gnuplot command...	1 KB

On a obtenu les résultats suivants :

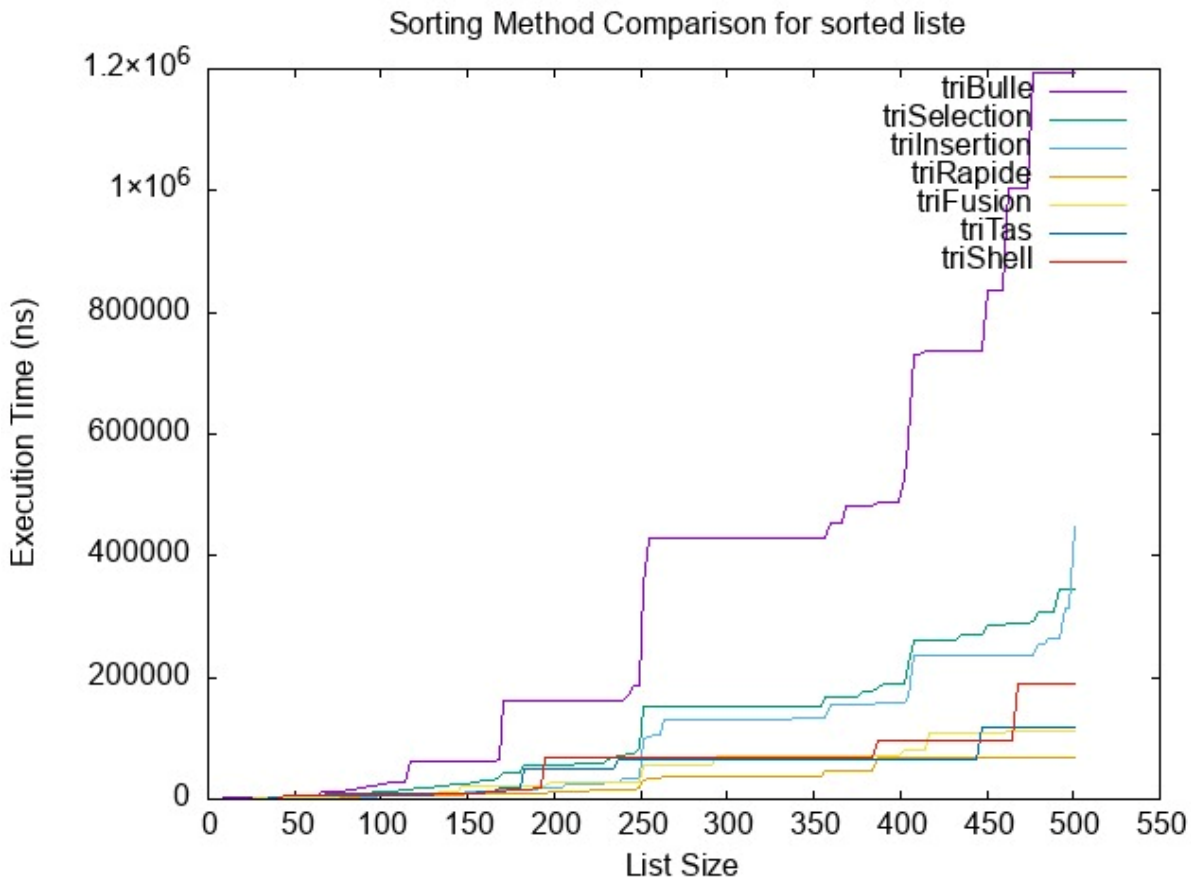
- Pour un tableau non trié :



- Pour un tableau semi trié :



- Pour un tableau trié :



V. Conclusion :

Après avoir examiné les courbes précédentes, on remarque que, pour des tableaux de taille inférieure à 100 éléments, les temps d'exécution de tous les algorithmes de tri sont approximativement équivalents. Cependant, à mesure que la taille du tableau augmente, le tri à bulles devient de loin l'algorithme le plus lent, tandis que le tri rapide reste le plus rapide. En revanche, les tris par sélection et par insertion requièrent un temps d'exécution légèrement inférieur à celui du tri à bulles, mais supérieur à celui du tri rapide. D'un autre côté, les temps d'exécution des tris par fusion, par tas et Shell se rapprochent de celui du tri rapide, mais ils diffèrent dans des cas spécifiques en fonction de la taille du tableau. Par exemple, pour des tableaux de taille supérieure à 450 éléments, le tri Shell devient plus lent par rapport aux autres algorithmes, qui conservent des performances similaires.