ENSET
ECOLE NORMALE SUPÉRIEURE DE L'ENSEIGNEMENT TECHNIQUE DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA
المدرسة العليا لأساتذة التعليم التقني المحمدية

# Comparaison des algorithmes de tri en C

Compte Rendu

Encadré par : Pr. Qbadou Mohammed

Réalisé par : Mohamed ANADDAM

Filière : II-CCN

What in me is dark , illumine !

# Introduction :

Mon projet, intitulé "Comparaison des Algorithmes de Tri en langage C", vise à explorer et comparer différents algorithmes de tri. Ces algorithmes incluent le tri à bulles, le tri par sélection, le tri par insertion, le tri fusion, le tri rapide et le tri par tas, tous écrits en langage C.

Je chercherai à évaluer ces algorithmes pour comprendre comment chacun se comporte en fonction de la taille des données à trier. Pour cette comparaison, j'ai implémenté les fonctions de tri dans mon code.

À la fin de mon analyse, j'utiliserai la fonction gaussienne et GNU Plot pour représenter graphiquement les performances de ces algorithmes de tri. Ces graphiques m'aideront à visualiser et à comprendre les différences de performances entre les algorithmes en fonction de la taille des données.

Ce projet est conçu pour m'aider à saisir quel algorithme de tri pourrait être le plus efficace dans divers scénarios, ce qui est crucial pour comprendre comment choisir la meilleure approche de tri en situations réelles.

# Objectifs :

- ✓ Implémentation des Algorithmes de Tri :
  - ❖ tri à bulles (Bubble Sort)
  - ❖ tri par sélection (Selection Sort)
  - ❖ tri par insertion (Insertion Sort)
  - ❖ tri fusion (Merge Sort)
  - ❖ tri rapide (Quick Sort)
  - ❖ tri par tas (Heap Sort)
- ✓ Comparaison des Performances :
- ✓ Analyse des Différents Scénarios:
  - ❖ dans des scénarios de tri aléatoires .
  - ❖ dans les meilleurs cas, où les données sont déjà triées.
  - ❖ les pires cas, où les données sont triées dans l'ordre inverse.
  - ❖ dans des scénarios mixtes où les données présentent un mélange d'ordres.
- ✓ Utilisation de la Fonction Gaussienne
- ✓ Création de Graphiques avec GNU Plot

# Méthodologie :

## 1. Implémentation des Algorithmes de Tri :

Dans le dossier "Project C", plusieurs algorithmes de tri ont été implémentés. Dans le fichier "sorting_functions.h", les algorithmes suivants ont été déclarés :

- Tri à bulles (Bubble Sort)
- Tri par sélection (Selection Sort)
- Tri par insertion (Insertion Sort)

- Tri fusion (Merge Sort)
- Tri rapide (Quick Sort)
- Tri par tas (Heap Sort)

Ces algorithmes ont ensuite été implémentés dans le fichier "sorting_functions.c" en langage C. Chaque algorithme a été rigoureusement testé et vérifié pour son bon fonctionnement.

## 2. Mesure des Performances des Algorithmes :

Dans le fichier "project.c", un script a été développé pour mesurer les performances de chaque algorithme en fonction de différentes tailles de tableau et de scénarios variés. Plusieurs étapes clés ont été entreprises :

- ✓ **Génération de Tableaux** : Des fonctions ont été créées pour générer des tableaux de différentes tailles et types, notamment :
  - Tableaux aléatoires
  - Tableaux triés par ordre croissant (meilleur cas)
  - Tableaux triés par ordre décroissant (pire cas)
  - Tableaux mélangés
- ✓ **Mesures de Temps** : En utilisant la fonction clock(), le temps d'exécution de chaque algorithme de tri a été mesuré pour différentes tailles de tableau et scénarios. Le temps d'exécution de chaque algorithme a été enregistré pour :
  - Tableaux aléatoires
  - Scénarios du meilleur cas
  - Scénarios du pire cas
  - Scénarios mixtes
- ✓ Stockage des Données : Les données de temps ont été stockées dans des fichiers CSV distincts pour chaque scénario, facilitant ainsi l'analyse ultérieure.

## 3. Représentation Visuelle des Données :

De plus, dans le fichier "sorting_plots.plt", des scripts GNU Plot ont été rédigés pour générer des représentations visuelles sous forme d'images, affichant les performances de chaque algorithme dans différents scénarios :

- Représentation graphique des tableaux aléatoires
- Représentation graphique des scénarios du meilleur cas
- Représentation graphique des scénarios du pire cas
- Représentation graphique des scénarios mixtes

Chaque image correspond à l'un des scénarios mentionnés et offre une compréhension visuelle de la performance de chaque algorithme en fonction de la taille du tableau dans ce scénario spécifique.

# Résultats :

Les performances des algorithmes de tri ont été évaluées à travers différentes tailles de tableaux et types de données, et les résultats sont présentés ci-dessous sous forme de tableaux et de graphiques.

### Données de Performance :

Les données de performance collectées à partir des fichiers CSV ("random_data.csv", "best_case_data.csv", "worst_case_data.csv", "mixed_case_data.csv") sont résumées dans les tableaux suivants :

### Tableau 1 : Temps d’Exécution pour les Tableaux Aléatoires:

| Taille du Tableau | Tri à Bulles | Tri par Sélection | Tri par Insertion | Tri Fusion | Tri Rapide | Tri par Tas |
|---|---|---|---|---|---|---|
| 100 | 0.000040 | 0.000018 | 0.000002 | 0.000013 | 0.000039 | 0.000017 |
| 1000 | 0.002991 | 0.001473 | 0.000005 | 0.000088 | 0.002059 | 0.000153 |
| 10000 | 0.272852 | 0.127205 | 0.000068 | 0.001028 | 0.227054 | 0.002636 |

## Tableau 2 : Temps d'Exécution pour les Scénarios du Meilleur Cas :

| Taille du Tableau | Tri à Bulles | Tri par Sélection | Tri par Insertion | Tri Fusion | Tri Rapide | Tri par Tas |
|---|---|---|---|---|---|---|
| 100 | 0.000020 | 0.000018 | 0.000002 | 0.000011 | 0.000037 | 0.000013 |
| 1000 | 0.001664 | 0.001445 | 0.000005 | 0.000135 | 0.002133 | 0.000188 |
| 10000 | 0.122366 | 0.127454 | 0.000054 | 0.001067 | 0.320251 | 0.002681 |

## Tableau 3 : Temps d'Exécution pour les Scénarios du Pire Cas :

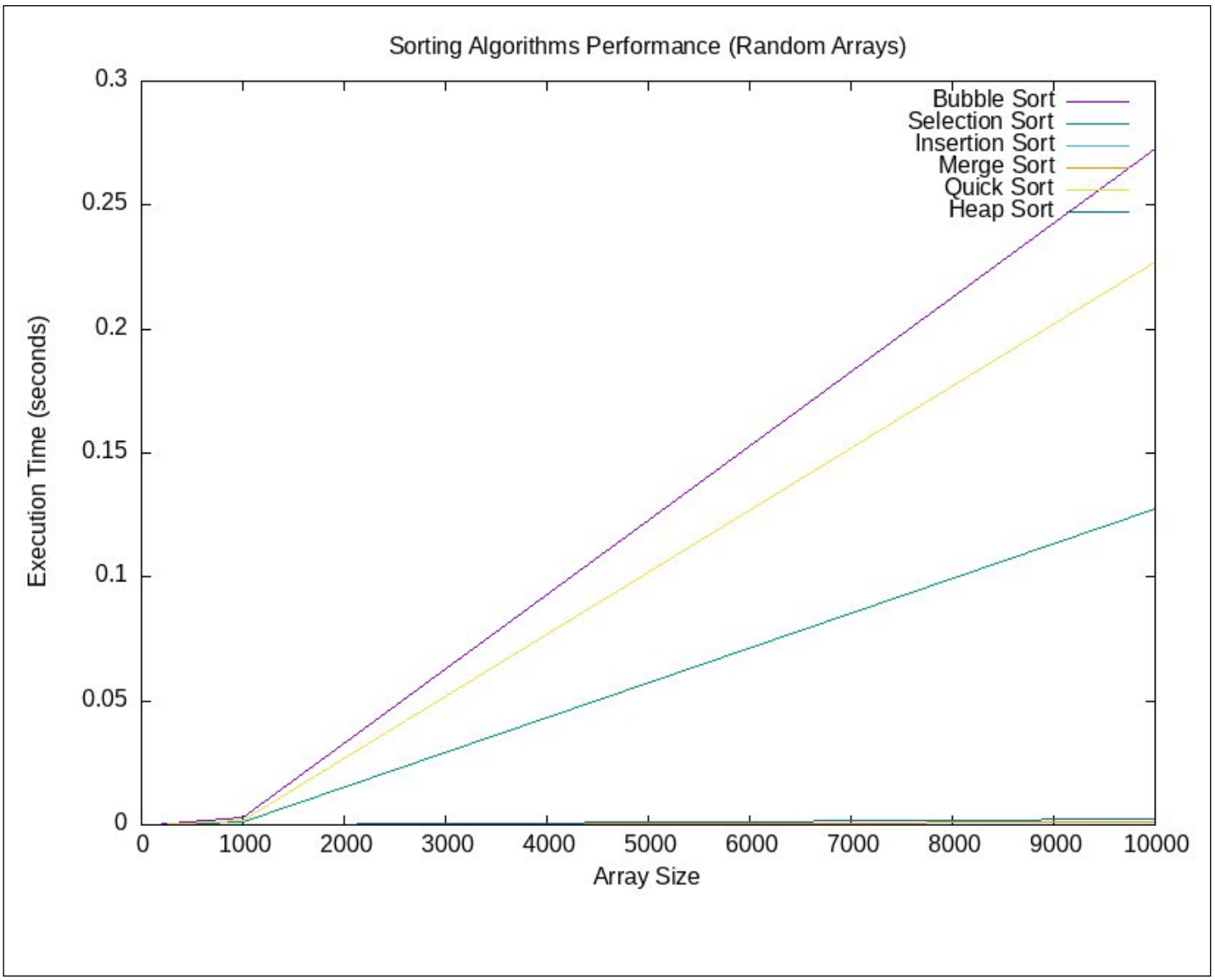
| Taille du Tableau | Tri à Bulles | Tri par Sélection | Tri par Insertion | Tri Fusion | Tri Rapide | Tri par Tas |
|---|---|---|---|---|---|---|
| 100 | 0.000042 | 0.000019 | 0.000023 | 0.000012 | 0.000026 | 0.000013 |
| 1000 | 0.003859 | 0.001537 | 0.001396 | 0.000086 | 0.001706 | 0.000175 |
| 10000 | 0.250341 | 0.134415 | 0.148723 | 0.000966 | 0.221918 | 0.002451 |

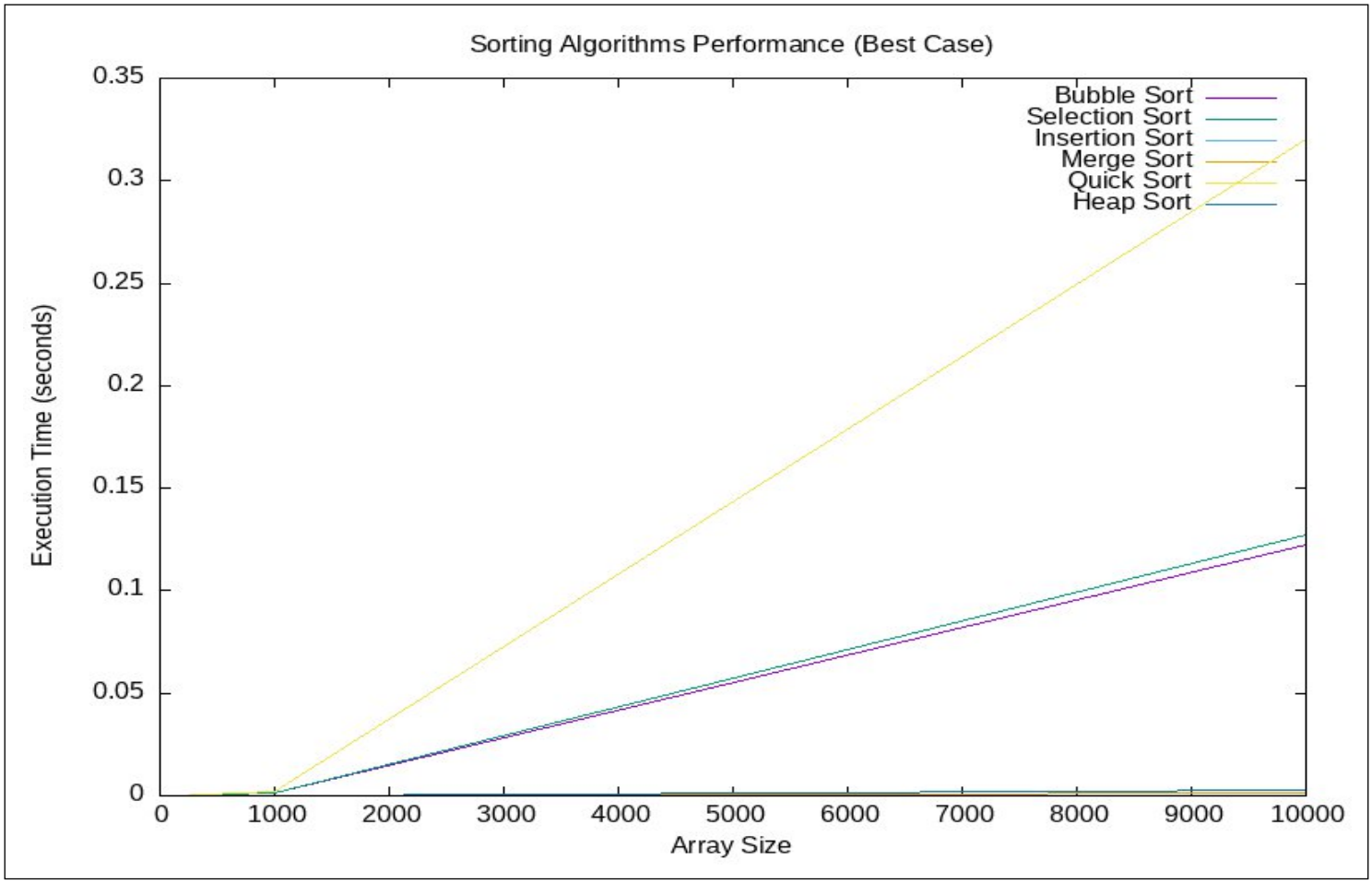
## Tableau 4 : Temps d'Exécution pour les Scénarios Mixtes:

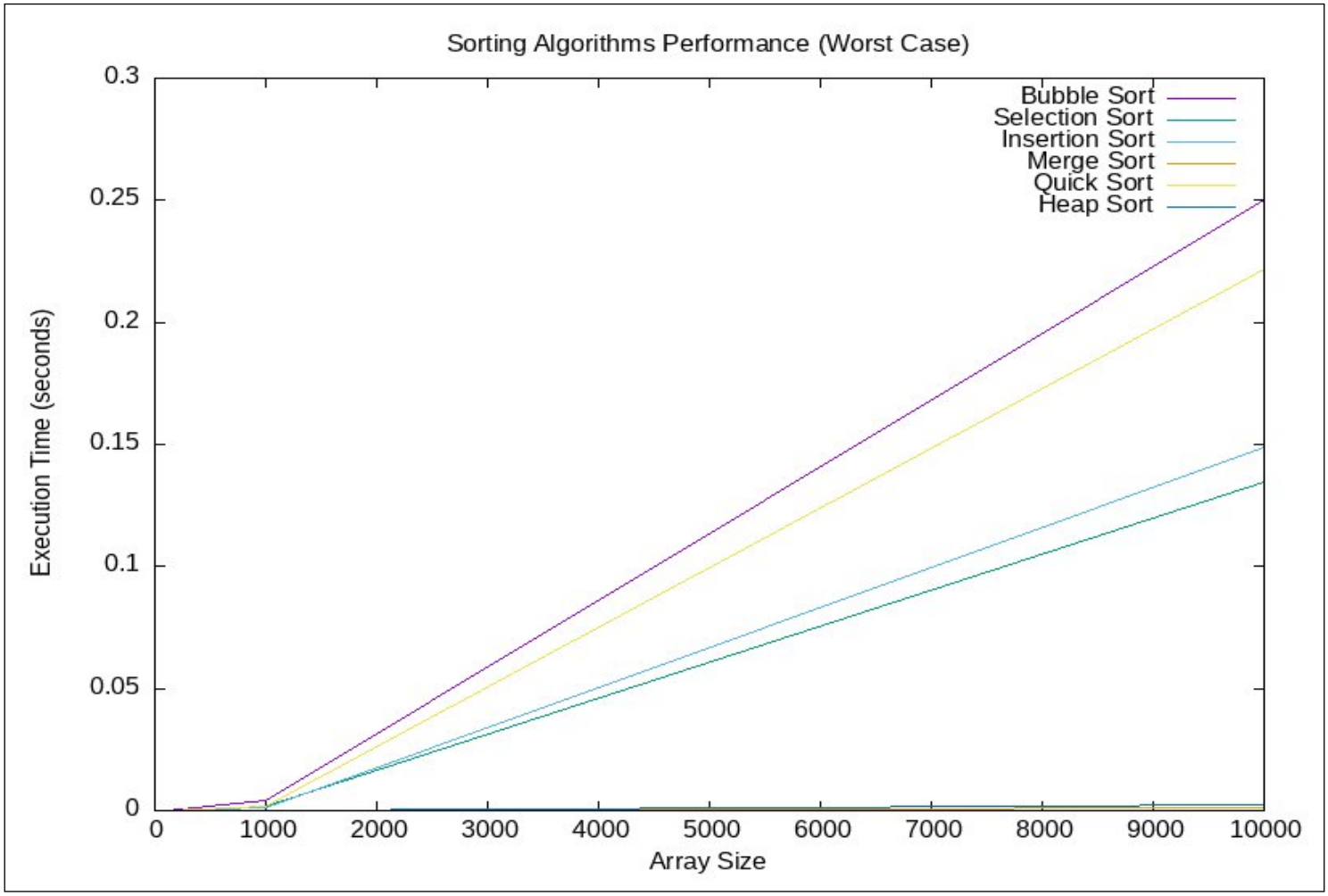
| Taille du Tableau | Tri à Bulles | Tri par Sélection | Tri par Insertion | Tri Fusion | Tri Rapide | Tri par Tas |
|---|---|---|---|---|---|---|
| 100 | 0.000038 | 0.000022 | 0.000014 | 0.000016 | 0.000009 | 0.000016 |
| 1000 | 0.002916 | 0.001401 | 0.000714 | 0.000139 | 0.000090 | 0.000202 |
| 10000 | 0.216120 | 0.130521 | 0.076693 | 0.001642 | 0.001424 | 0.002905 |

## Graphiques de Performance:

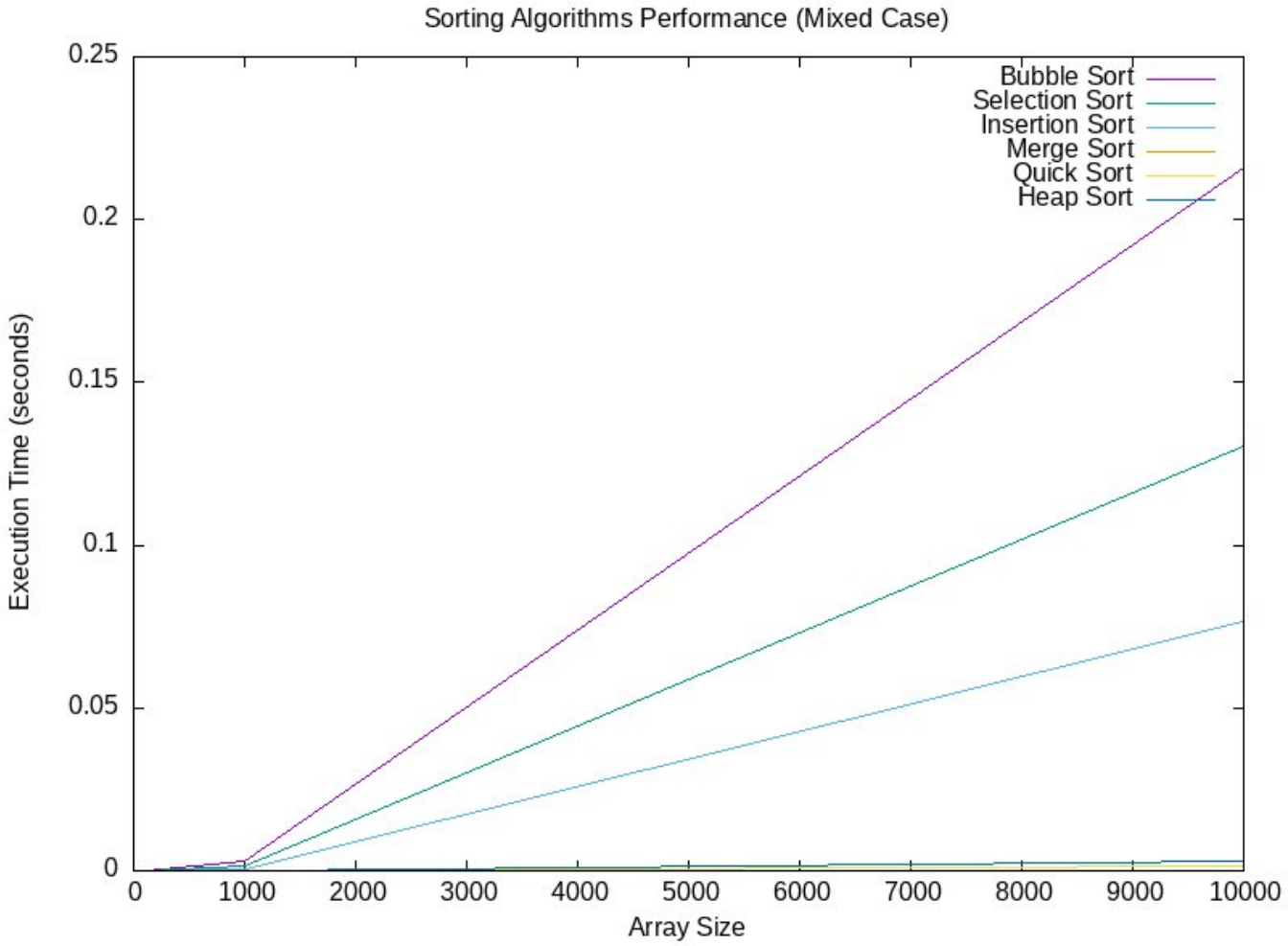
Les graphiques générés à partir du fichier "sorting_plots.plt" illustrent visuellement les performances des algorithmes dans différents scénarios. Les images suivantes résument ces résultats :






What in me is dark , illumine !




## Analyse des Résultats :

L'analyse des performances des différents algorithmes de tri révèle des tendances et des caractéristiques essentielles. Ces résultats sont organisés en fonction des types de données et de la taille des tableaux, avec des observations spécifiques pour chaque scénario.

- Tableaux Aléatoires :
  - Le tri rapide (Quick Sort) s'est avéré très efficace pour les tableaux aléatoires, avec des temps d'exécution compétitifs. Cela s'explique par sa capacité à diviser rapidement les tableaux en sous-tableaux, ce qui réduit le nombre de comparaisons.
  - Le tri fusion (Merge Sort) a également bien performé avec des temps d'exécution raisonnables. Il tire parti de la

stratégie de diviser et conquérir pour trier efficacement des données aléatoires.
    - Le tri par tas (Heap Sort) a montré des performances stables, bien que légèrement moins efficaces que le tri rapide et le tri fusion pour de petites tailles de tableau.
    - Les tri à bulles, tri par sélection et tri par insertion ont été les moins performants pour les tableaux aléatoires. Le tri à bulles est particulièrement inefficace pour les données aléatoires, car il effectue de nombreuses passes et comparaisons.
- Meilleur Cas :
    - le tri par insertion et le tri à bulles ont affiché des performances similaires et les plus rapides, car les données sont déjà partiellement triées. Cependant, ces deux algorithmes restent lents pour les grandes tailles de tableau.
    - Le tri rapide a montré des temps d'exécution compétitifs, en tirant parti de sa stratégie de pivot pour minimiser le nombre de comparaisons.
    - Le tri fusion est resté efficace même dans le meilleur des cas, bien qu'il effectue plus d'opérations que le tri rapide.
    - Le tri par tas et le tri par sélection ont montré des performances légèrement moins efficaces que les autres algorithmes pour les meilleures situations.
- Pire Cas:
    - Les scénarios du pire cas ont mis en évidence les limites du tri à bulles, du tri par sélection et du tri par insertion. Ces algorithmes deviennent très lents lorsque les données sont inversées.
    - Le tri rapide, bien que moins performant que dans les meilleurs cas, a maintenu des temps d'exécution acceptables.
    - Le tri fusion a montré une grande stabilité, avec des temps d'exécution constants quelle que soit la disposition des données.
    - Le tri par tas a maintenu une performance stable, mais est devenu relativement lent pour de grandes tailles de tableau.
- Scénarios Mixtes :

- Les scénarios mixtes ont révélé des performances intéressantes. Le tri rapide a généralement bien performé grâce à sa flexibilité.
- Le tri fusion a également affiché des performances solides pour ces scénarios.
- Le tri à bulles, le tri par sélection et le tri par insertion restent les moins efficaces, même pour des mélanges de données.

Les graphiques générés offrent une visualisation claire de ces performances, ce qui peut faciliter la prise de décision lors du choix de l'algorithme de tri pour une tâche particulière.

### Avantages et Inconvénients :

- Tri Rapide (Quick Sort) : Avantages notables en raison de sa rapidité pour les tableaux aléatoires et les meilleurs cas. Sa principale faiblesse est sa sensibilité aux pires cas et son utilisation de la mémoire.
- Tri Fusion (Merge Sort) : Performances constantes dans la plupart des cas, même si elles nécessitent une mémoire supplémentaire pour la fusion des données.
- Tri par Tas (Heap Sort) : Relativement stable, mais moins efficace pour les petits tableaux en raison de sa nature itérative.
- Tri à Bulles, Tri par Sélection, Tri par Insertion : Moins performants en général, car ils impliquent plus de mouvements et de comparaisons, se montrant plus adaptés pour des petites tailles de tableaux ou des scénarios où les données sont déjà partiellement triées.

# Conclusion :

Dans ce projet, nous avons exploré l'univers des algorithmes de tri, en les mettant à l'épreuve avec des tableaux de différentes tailles et des distributions de données variées. Ce que nous avons découvert, c'est que ces méthodes de tri présentent leurs propres forces et faiblesses, en fonction de la taille du tableau et de la nature des données.

La leçon essentielle à retenir est que le choix de l'algorithme de tri approprié est crucial, en fonction de la tâche spécifique à accomplir. Des facteurs tels que la taille du tableau et la distribution des données jouent un rôle significatif dans la détermination de l'algorithme qui offre les meilleures performances.

Il est important de noter que nos résultats sont adaptés aux tableaux numériques et qu'ils ne s'appliquent pas nécessairement à d'autres types de données, tels que les chaînes de caractères ou les objets personnalisés.

Ce projet nous a fourni des informations précieuses sur le monde des algorithmes de tri, et nous espérons qu'il vous a permis de mieux comprendre ces méthodes fondamentales de l'informatique.

<< What in me is dark , illumine ! >>

Check my github for the code please :

https://github.com/jkome/sorting_alogrithmes_comparaison.git