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

Département Mathématiques et Informatique

Filière : Génie du Logiciel et des Systèmes Informatiques Distribués

Etude comparative sur les algorithmes de tri

Réalisé par :

Fatima Ezzahrae BEN AKKA
Meryem BEN ASSILA

Année Universitaire : 2024/2025

Table des matières

1. Introduction

L'efficacité des algorithmes de tri est un sujet fondamental en informatique, notamment dans l'optimisation des performances des systèmes et applications.

Ce programme en langage C a pour objectif de comparer plusieurs algorithmes de tri classiques, à savoir le tri par sélection, le tri par insertion, le tri à bulles et le tri rapide et tri par fusion. Pour visualiser ces comparaisons, le programme utilise l'outil Gnuplot, qui génère des graphiques basés sur deux critères principaux : la taille des tableaux et le degré de perturbation d'un tableau préalablement trié.

L'utilisateur a la possibilité de définir plusieurs paramètres, tels que les intervalles de taille ou le nombre de perturbations, ainsi que de sélectionner les algorithmes qu'il souhaite comparer. Le programme offre donc une interface flexible pour explorer les performances de chaque algorithme en fonction de différents scénarios.

L'objectif de cette étude est de mieux comprendre les comportements de ces algorithmes selon le critère choisi et de visualiser les résultats sous forme de graphiques permettant une analyse claire et concise des performances.

2. Description des Algorithmes de Tri

a) Tri par Sélection

Le tri par sélection est un algorithme simple où l'on parcourt le tableau pour trouver l'élément minimum (ou maximum) et l'échanger avec l'élément en cours de traitement.

b) Tri par Insertion

Le tri par insertion consiste à construire le tableau trié progressivement, en insérant chaque élément à sa position correcte dans la partie déjà triée du tableau. Ce processus est répété jusqu'à ce que tous les éléments soient triés.

c) Tri à Bulles

Le tri à bulles compare des éléments adjacents et les échange si nécessaire, jusqu'à ce que le tableau soit trié. L'algorithme répète ce processus plusieurs fois jusqu'à ce qu'aucun échange ne soit nécessaire

d) Tri Rapide

Le tri rapide est un algorithme diviser-pour-régner qui utilise un élément pivot pour partitionner le tableau en deux sous-parties : une contenant les éléments inférieurs au pivot et l'autre contenant les éléments supérieurs.

e) Tri par Fusion

Le tri par fusion divise un tableau en deux sous-parties, les trie récursivement, puis fusionne les sous-tableaux triés.

3. Description du Programme

a) Structure générale

Le programme est structuré en plusieurs étapes : la saisie des paramètres par l'utilisateur, la génération des tableaux à trier (soit de manière aléatoire, soit par perturbation d'un tableau trié), l'application des différents algorithmes de tri, et enfin la visualisation des résultats à l'aide de Gnuplot.

Cinq algorithmes de tri sont implémentés : le tri par insertion, le tri à bulles, le tri par sélection, et le tri rapide. Le programme guide l'utilisateur à travers des structures de contrôle pour choisir les paramètres de comparaison (par taille de tableau ou par niveau de perturbation) et sélectionner les algorithmes de tri à comparer.

b) Fonctionnalités principales

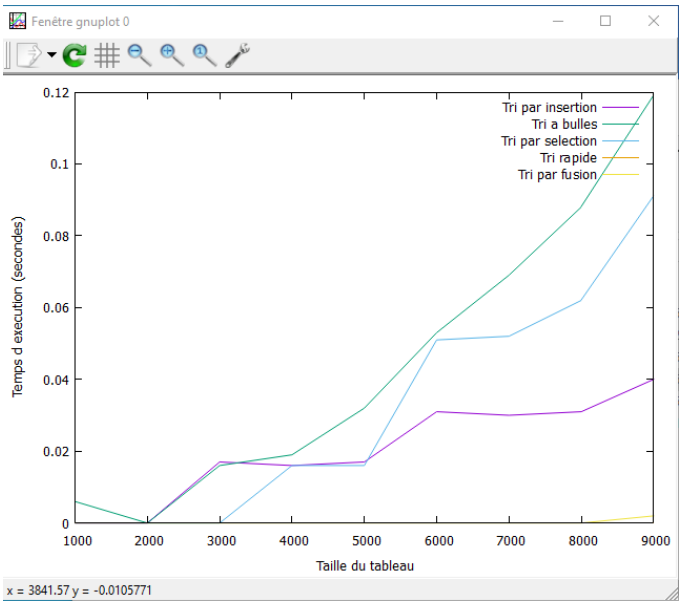
- **Choix du critère de comparaison** : L'utilisateur choisit de comparer les algorithmes soit en fonction de la **taille des tableaux**, soit en fonction de leur **perturbation**.
- **Saisie des paramètres** : La comparaison se fait soit selon **la taille**, dans ce cas l'utilisateur spécifie un intervalle de tailles de tableaux et un pas, ainsi qu'une plage de valeurs pour remplir les tableaux, soit selon **le nombre des perturbations**, le programme génère dans ce cas un tableau initialement trié avec des valeurs continues dans une plage spécifiée par l'utilisateur. Ensuite, l'utilisateur choisit un intervalle et un pas pour le nombre de perturbations à appliquer. Une perturbation consiste à échanger aléatoirement deux éléments du tableau, ce qui simule un désordre partiel du tableau.
- **Choix des algorithmes de tri** : L'utilisateur peut choisir quels algorithmes parmi les cinq (insertion, sélection, bulles, rapide, fusion) il souhaite comparer.

- **Affichage des résultats avec Gnuplot :** Le programme utilise Gnuplot pour afficher des graphiques représentant le temps d'exécution des algorithmes en fonction de la taille des tableaux ou du nombre des perturbations.

4. Résultats et Analyse

a) Comparaison selon la taille

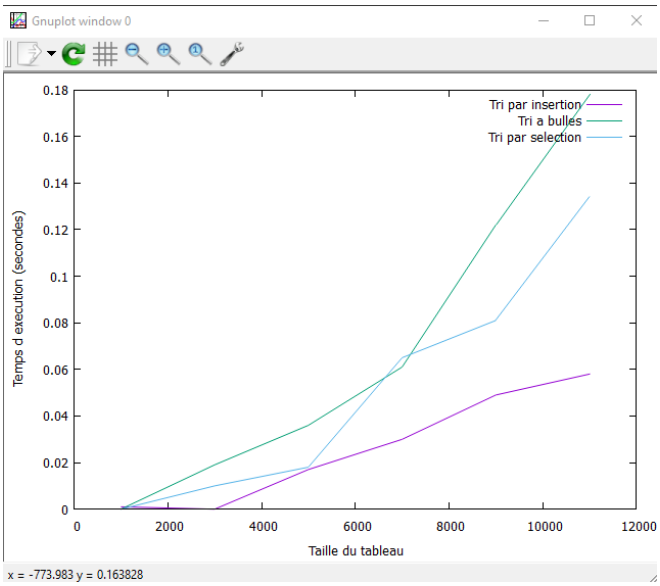
i. *Comparaison des algorithmes de tri traitée par le programme*



Analyse

Dans ce graphe, nous observons les temps d'exécution de trois algorithmes de tri : tri par insertion, tri à bulles, et tri par sélection. Tandis que les algorithmes tri rapide et tri par fusion ne figurent pas dans ce graphe, car ils se révèlent être très rapides par rapport aux autres.

ii. *Comparaison* des algorithmes de tri (tris par sélection, tri à bulles, tri par insertion ; de 1.000 jusqu'à 11.000) :



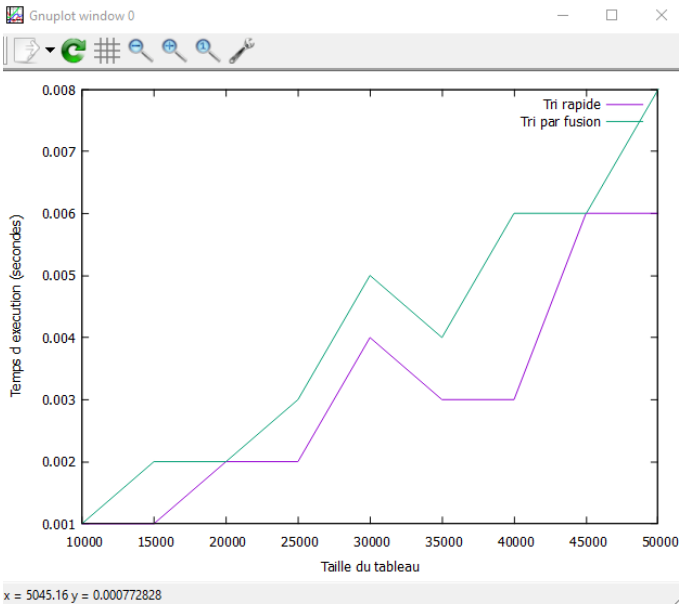
Analyse

Tri à bulles (courbe en haut, couleur verte) :
Le tri à bulles a la courbe la plus haute, indiquant qu'il est le plus lent des trois algorithmes. Cela correspond bien à sa complexité théorique de $(O(n^2))$. Il nécessite des comparaisons et des échanges répétés entre éléments voisins, ce qui devient rapidement inefficace pour de grandes tailles de tableaux.

Tri par sélection (courbe au milieu, couleur bleue) :
Le tri par sélection présente une courbe intermédiaire. Bien que cet algorithme ait également une complexité de $(O(n^2))$, il est généralement un peu plus rapide que le tri à bulles dans la pratique car il effectue un nombre réduit des échanges. Cependant, la croissance du temps d'exécution suit la même tendance quadratique, indiquant que la différence de performance entre les deux n'est pas énorme pour des tableaux de grande taille.

Tri par insertion (courbe en bas, couleur violette) :
Le tri par insertion est le plus rapide des trois, comme le montre la courbe la plus basse. Même si sa complexité théorique soit aussi en $(O(n^2))$, mais il semble bien se comporter par rapport aux autres algorithmes.

iii. Comparaison des algorithmes de tri (Tris par Fusion et Tri Rapide ; de 10.000 jusqu'à 50.000) :



Analyse

Tri rapide (courbe verte) :

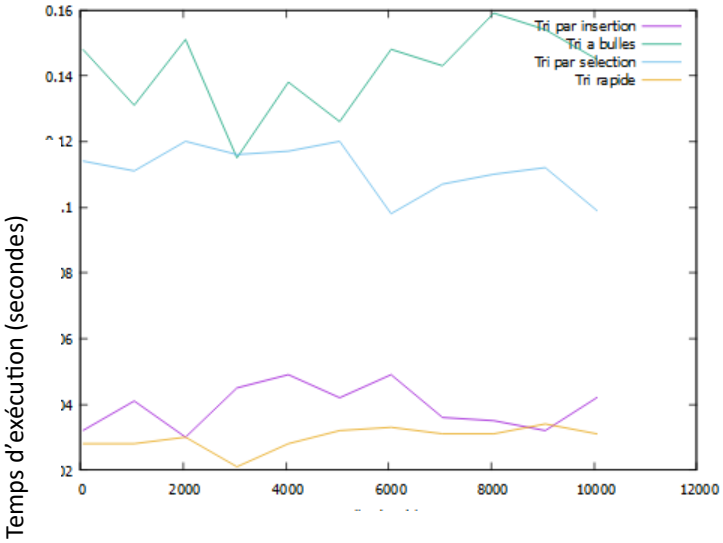
Le tri rapide montre une croissance modérée du temps d'exécution par rapport à la taille du tableau. La courbe est relativement plate au début, puis elle commence à augmenter lentement mais régulièrement. Cela correspond bien à la complexité théorique ($O(n/\log n)$) du tri rapide dans le meilleur et le cas moyen

Tri par fusion (courbe violette) :

Le tri par fusion présente également une croissance modérée, mais la courbe est légèrement plus rapide que celle du tri rapide. Cela indique que le tri par fusion est également un algorithme de complexité ($O(n/\log n)$), mais il semble plus lent que le tri rapide pour les tailles de tableaux données.

b) Comparaison selon le nombre des perturbations

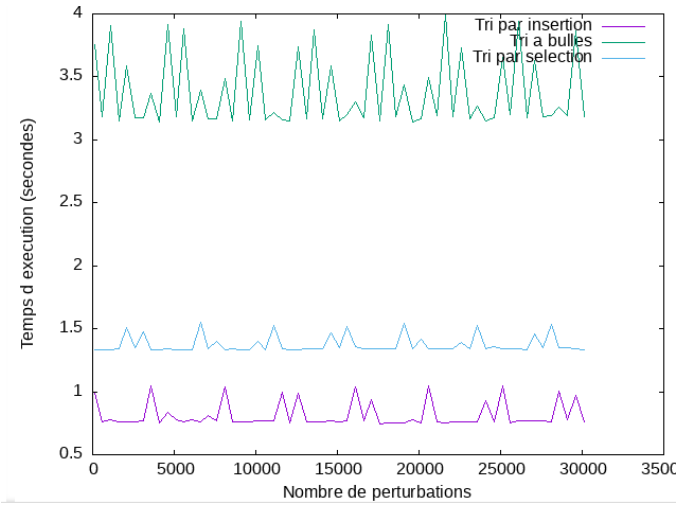
(Nombre des déplacements effectués sur le tableau)



Analyse

Malgré l'augmentation progressive des perturbations, les résultats montrent des courbes presque constantes, sans différence notable dans les temps d'e Nombre de perturbations nes.

Afin d'explorer cette anomalie, l'intervalle des perturbations a été élargi et le programme a été exécuté sur Google Colab, une plateforme offrant une gestion plus efficace des ressources mémoire pour permettre un nombre de perturbations plus élevé.



Toutefois, même avec ces ajustements, les courbes restent inchangées, ce qui suggère que le problème pourrait être lié à la nature des perturbations ou à une autre inefficience du code.

5. Conclusion

L'étude comparative des algorithmes de tri a révélé des différences marquées en termes de performance et de stabilité en fonction de la taille des tableaux et du niveau de perturbation. Le tri rapide et le tri par fusion se sont montrés les plus performants, offrant des temps d'exécution constants et faibles. Les algorithmes comme le tri à bulles et le tri par sélection sont les moins performants, présentant une forte augmentation du temps d'exécution à mesure que la taille de tableau augmente. Ces résultats soulignent l'importance de choisir l'algorithme de tri en fonction de la nature et de l'état initial des données.