|  |  |
| --- | --- |
| Etude visuelle sur les algorithmes de tri  En utilisant le langage de programmation C | YOUNESS HATTABI  Étudiant de première année filière GLSID  YOUSSEF NAZIH  Étudiant de première année filière GLSID |

Table des matières

[Chapitre I : Programmation en C et GNUplot 2](#_Toc180182216)

[Introduction 2](#_Toc180182217)

[1. Programmation en C 2](#_Toc180182218)

[1.1. Le tri à bulles 2](#_Toc180182219)

[1.2. Le tri par insertion 2](#_Toc180182220)

[1.3. Le tri par sélection 3](#_Toc180182221)

[1.4. Le tri fusion 3](#_Toc180182222)

[1.5. Le tri rapide 4](#_Toc180182223)

[1.6. Le tri par dénombrement 5](#_Toc180182224)

[1.7. La fonction main 5](#_Toc180182225)

[2. B 6](#_Toc180182226)

[3. C 6](#_Toc180182227)

# Chapitre I : Programmation en C et GNUplot

## Introduction

Dans ce chapitre, nous allons explorer le code source qui alimente à la fois les algorithmes de tri et l'affichage graphique. Nous allons programmer en C et utiliser Gnuplot pour les graphes.

## Programmation en C

Nous avons choisi de programmer six algorithmes de tri : Bubble, Insertion, Sélection, Merge, Quick et Counting :

### Le tri à bulles

Le tri à bulles (ou "Bubble sort") est un algorithme de tri simple qui fonctionne en parcourant un tableau plusieurs fois. À chaque passage, il compare deux éléments adjacents et les échange s'ils sont dans le mauvais ordre. Ce processus est répété jusqu'à ce que le tableau soit entièrement trié. Bien qu'il soit facile à comprendre, cet algorithme est inefficace pour de grands tableaux à cause de sa complexité en O(n²).

Une image contenant capture d’écran, texte

Description générée automatiquement

Figure 1: Code de tri à bulles (Bubble sort)

### Le tri par insertion

Le tri par insertion est un algorithme de tri qui construit progressivement un tableau trié en déplaçant les éléments un par un. À chaque itération, il prend un élément du tableau non trié et le place à la position correcte dans la partie triée. Ce processus se répète jusqu'à ce que tous les éléments soient triés. Le tri par insertion est efficace pour les petits tableaux ou les tableaux presque triés, avec une complexité en O(n²).

Une image contenant capture d’écran, texte

Description générée automatiquement

Figure 2: Code de tri par insertion (Insertion Sort)

### Le tri par sélection

Le tri par sélection est un algorithme de tri qui divise le tableau en deux parties : une partie triée et une non triée. À chaque itération, il recherche l'élément le plus petit (ou le plus grand) dans la partie non triée, puis l'échange avec le premier élément de cette partie. Ce processus est répété jusqu'à ce que tout le tableau soit trié. Le tri par sélection a une complexité en O(n²) et est peu efficace pour les grands tableaux.

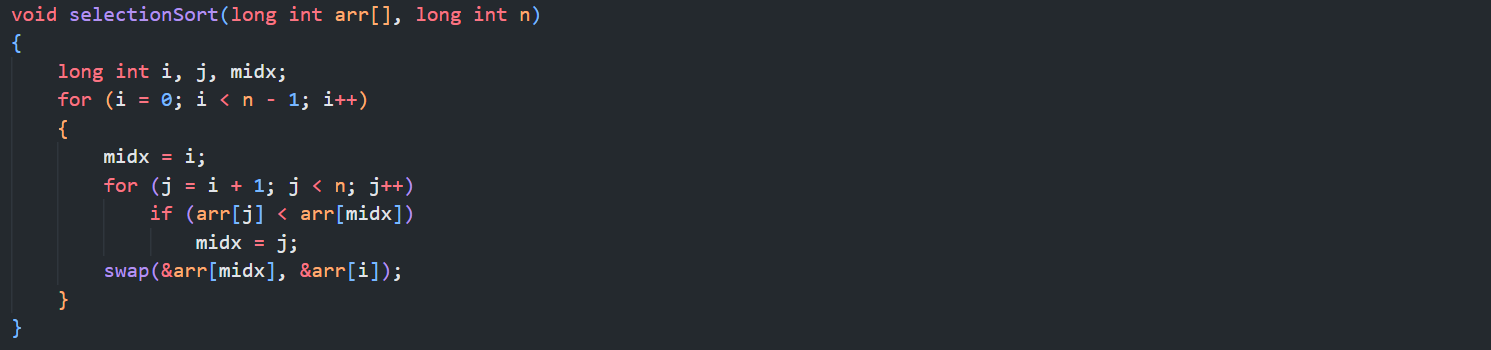


Figure 3: Code de tri par sélection (Sélection Sort)

Ces trois premiers algorithmes de tri sont très faciles à programmer et à implémenter dans la plupart des langages, mais souffrent de performances médiocres. En revanche, les trois suivants sont beaucoup plus difficiles à programmer, mais offrent des temps d'exécution bien plus rapides.

### Le tri fusion

Premièrement, il y a le tri fusion. Cet algorithme est basé sur la méthode "diviser pour régner" et divise récursivement le tableau en sous-tableaux jusqu'à ce qu'ils aient une taille de 1, puis les fusionne tout en les triant. Il est efficace avec une complexité en O(nlog n), mais nécessite plus de mémoire pour la fusion.

Pour l'implémenter dans notre code, nous divisons la fonction en deux parties : une fonction de fusion (merge) :

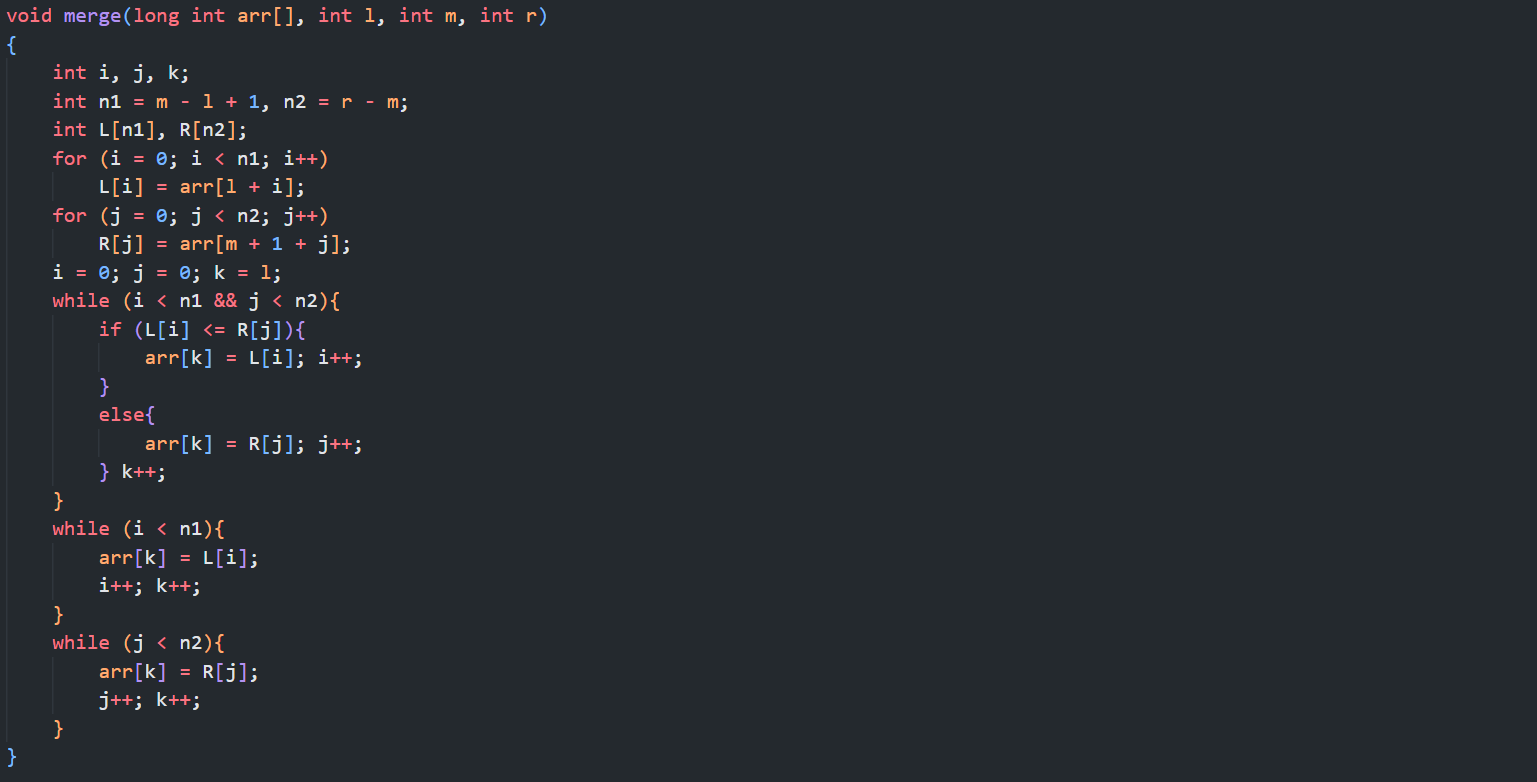


Figure 4: Code de la fonction de fusion

Et pour la deuxième partie on a une fonction principale de tri fusion (mergeSort) :

Une image contenant capture d’écran, texte

Description générée automatiquement

Figure 5: Code de tri par fusion (Merge Sort)

### Le tri rapide

Le tri rapide (Quick Sort) est un algorithme de tri efficace aussi basé sur la méthode "diviser pour régner". En C, il fonctionne en sélectionnant un "pivot" à partir du tableau et en réorganisant les éléments de manière que les éléments plus petits que le pivot soient placés à gauche et ceux plus grands à droite. Ensuite, il applique récursivement le même processus aux sous-tableaux à gauche et à droite du pivot. Ce processus permet de trier le tableau en divisant successivement le problème en sous-problèmes plus petits. L'algorithme a une complexité moyenne de O (n log n).

Pour l'implémenter dans notre code, nous divisons la fonction en deux parties : une fonction de partition (partition) :

Une image contenant capture d’écran, texte

Description générée automatiquement

Figure 6: Code de la fonction de partition

Et pour la deuxième partie on a une fonction principale de tri rapide (quickSort) :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 7 : Code de tri rapide (Quick Sort)

### Le tri par dénombrement

Le tri par dénombrement (Counting Sort) est un algorithme qui trie des nombres en comptant combien de fois chaque valeur apparaît (**fréquence**). On crée un tableau pour stocker ces fréquences, puis on utilise ces informations pour remettre les nombres dans le bon ordre. Il est rapide si les nombres à trier sont dans une plage de valeurs **limitée**, car il parcourt les données en fonction de ces fréquences.

Ce qui différencie le tri par dénombrement des cinq premiers algorithmes mentionnés, c'est qu'il est non comparatif. Sa complexité est O(n + r), où n est le nombre d'éléments et r la différence entre la plus grande et la plus petite valeur.

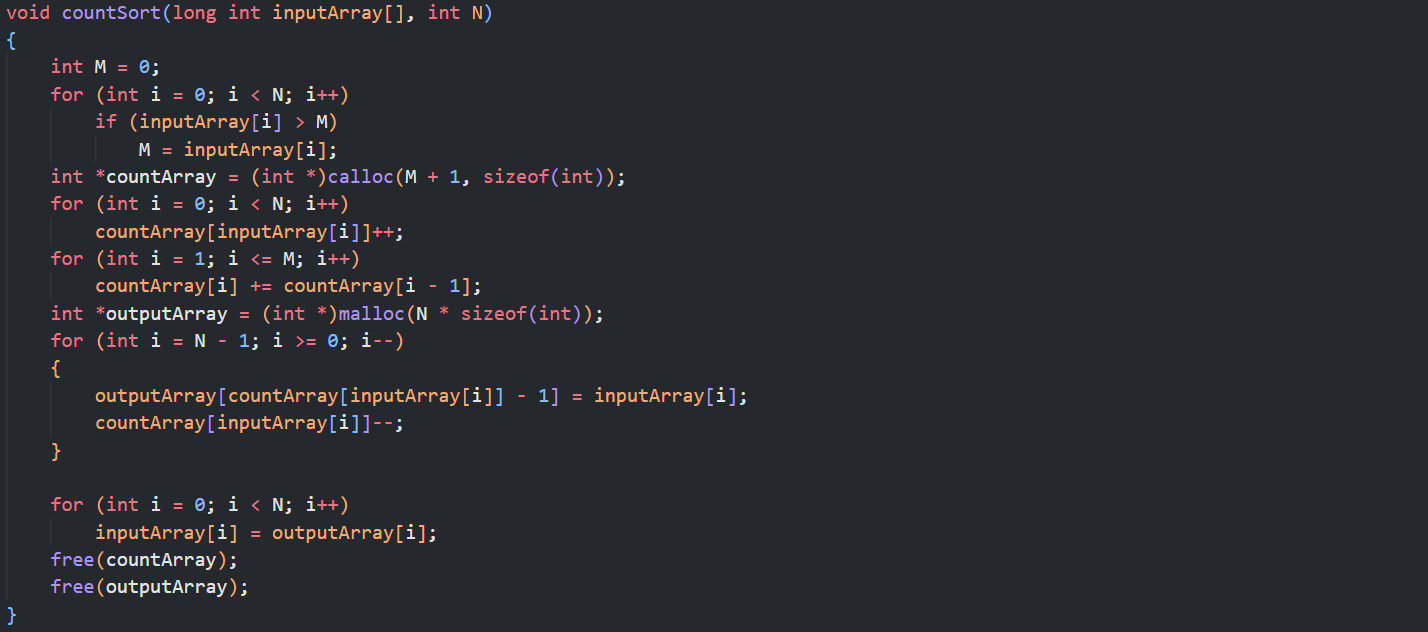


Figure 8 : Le code de tri par dénombrement

### La fonction main

Dans la fonction main, nous avons choisi de suivre les évolutions à travers 10 tableaux, dont les tailles varient de 10 000 à 100 000 et imprimer les resultats.

Une image contenant texte, capture d’écran

Description générée automatiquement

Figure 9 : Le code de fonction main

## Programmation en GNUplot

## B

## C