|  |  |
| --- | --- |
| 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 : Les tris et leur code 2](#_Toc180132177)

[Introduction 2](#_Toc180132178)

[1. Programmation des algorithmes de tri 2](#_Toc180132179)

[2. B 3](#_Toc180132180)

[3. C 3](#_Toc180132181)

# Chapitre I : Les tris et leur code

## 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 des algorithmes de tri

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

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 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) dans le pire des cas.

Une image contenant capture d’écran, texte

Description générée automatiquement

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

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.

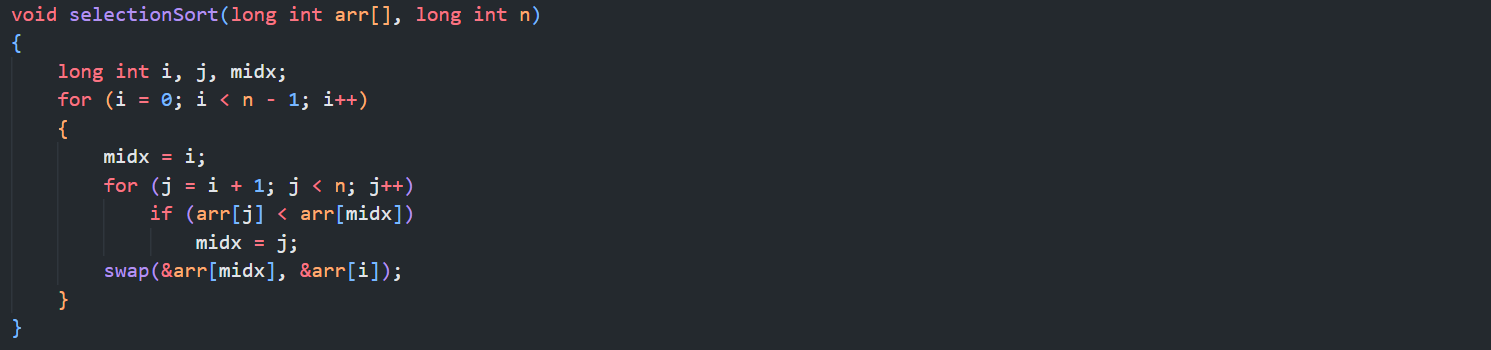


Figure 3: Code de tri par sélection (Selection Sort)

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.

Premièrement, il y a le tri fusion. Cet algorithme 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(n log 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) :

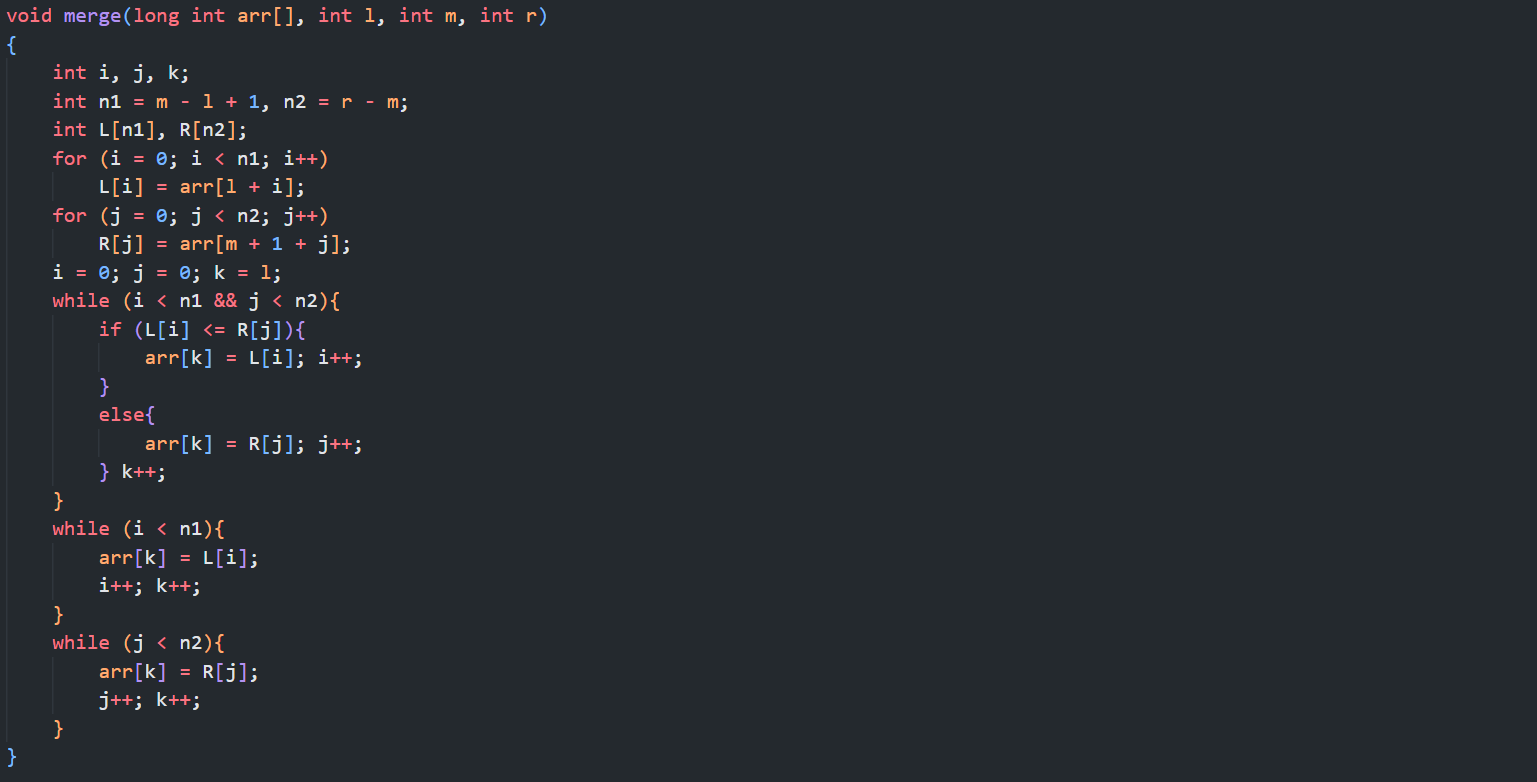


Figure 4: Code de la fonction de fusion

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)

## B

## C