CSE311 PROGRAMMING TERM PROJECT FINAL REPORT

In this project I’m comparing Kruskal and Prim’s Algorithms.Both are Greedy algorithms to find the MST which is a tree that has the following characteristics:

1. acyclic
2. connected
3. undirected graph.

I used the following way the construct my Kruskal Algorithm:

metin, ekran görüntüsü, yazı tipi, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

I wrote the code for Kruskal Algo in C language. I used qsort to sort the edges of E by nondecreasing weight w. Kruskal has Θ(|V|log|V|) time complexity for sparse graphs and Θ(|V|^2log|V|) time complexity for dense graphs.

I used the following way the construct my Kruskal Algorithm:

metin, yazı tipi, ekran görüntüsü, cebir içeren bir resim

Açıklama otomatik olarak oluşturuldu

I wrote the code for Prim Algo in C language. Prim algorithm has Θ(|V|^2) time complexity.

INPUT STYLE

To compare these two algoritms I created adjacency matrices firstly.

I use the following input format for both of the algorithms:

For example 5 at the beginning represents that the graph is a V×V matrix

Here is an example :

5

0 2 0 6 0

2 0 3 8 5

0 3 0 0 7

6 8 0 0 9

0 5 7 9 0

MATRIX GENERATOR

I’m using matrix generator written by Oğuzhan İçelliler. Code will be executed by typing

“./a.out 1000 0” 1000 means generate 1000x1000 matrix 0 means generate MST.

“./a.out 1000 1” if we give 1 that means increasing the density comparing giving 0.

The graph that we created will be atleast MST.The logic for this code is basically like

the following:

1. Type the matrix size (V x V) for example typing 500 will create 500x500 adjacent matrix.
2. Randomly select a number between 1 and 100 (inclusive).
3. If the chosen number is less than or equal to the user's chance value, select a random number between 1 and 9 (inclusive) and assign it to the current edge's value.



TESTING ENVIRONMENT

CPU: AMD Ryzen 7 5800X (8 cores (physical), 16 threads (logical))

OS: Windows 10 Pro

Frequency: With a base clock speed of 3.8 GHz and a max boost clock speed of 4.7 GHz

Windows Version: 22H2

RAM: 16 GB (2×8 dual channel) 3200 MHZ

Cache L1: 64K (per core)

Cache L2: 512K (per core)

Cache L3: 32MB

EMPIRICAL ANALYSIS

I will be running each for 5 times but first one will be discarded due to caching issues.



All programs will be compiled with GCC version 11.3.0 and with default level of optimization.

My first experiment will be like the following. Constant density (2) with increasing matrix size

1. 500x500 adjacency matrix to compare my Kruskal and Prim’s Algorithms.

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. 1500x1500 adjacency matrix to compare my Kruskal and Prim’s Algorithms.

“density\_2\_adj\_matrix1500.txt”

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. 2500x2500 adjacency matrix to compare my Kruskal and Prim’s Algorithms.

“density\_2\_adj\_matrix2500.txt”

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. 3500x3500 adjacency matrix to compare my Kruskal and Prim’s Algorithms.

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. 4500x4500 adjacency matrix to compare my Kruskal and Prim’s Algorithms.

“density\_2\_adj\_matrix4500.txt”

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

FIRST EXPERIMENT CHART

metin, ekran görüntüsü, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

“Average values are used for the graph (Kruskal & Prim’s)”

My Second experiment will be like the following. Varying density with constant matrix size.

1. Density (0) “which means MST” with 3500x3500 matrix
2. “density\_0\_adj\_matrix3500.txt”

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Density (1) with 3500x3500 matrix

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Density (2) with 3500x3500 matrix

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Density (3) with 3500x3500 matrix

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Density (8) with 3500x3500 matrix

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Density (9) with 3500x3500 matrix

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Density (10) with 3500x3500 matrix

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

SECOND EXPERIMENT CHART

“Average values are used for the graph (Kruskal & Prim’s)”

My third experiment will have increased matrix size compared to second experiment.

1. Density (0) “which means MST” with 10.000x10.000 matrix “density\_0\_adj\_matrix10000.txt”

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Density (1) with 10.000x10.000 matrix

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Density (2) with 10.000x10.000 matrix

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Density (3) with 10.000x10.000 matrix

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Density (8) with 10.000x10.000 matrix

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Density (9) with 10.000x10.000 matrix

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Density (10) with 10.000x10.000 matrix

“density\_10\_adj\_matrix10000.txt”

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

THIRD EXPERIMENT CHART

“Average values are used for the graph (Kruskal & Prim’s)”

My thoughts about Empirical Analysis:

1) Kruskal algorithm performs worse than I expected considering especially my 2nd and 3rd experiments.Kruskal algorithm only performs better than Prim’s algorithm with MST. Theoretically in sparse graphs Kruskal should perform better considering it’s Time complexity is Θ(|V|log|V|) and Prim’s algo’s time complexity is Θ(|V|^2).

2) The code that I got from “geeksforgeeks” was using qsort to do the sorting required for Kruskal algorithm. qsort() is a sorting algorithm from the C standard library (stdlib.h) and it’s not the same thing with quick sort algorithm it can even be not-in-place sorting algorithm.That may causing the efficiency problem for Kruskal algorithm.

3) I tried shuffling for my Kruskal algorithm to see if it is performing better or not. It didn’t solve my problem with the Kruskal algorithm’s performance. It performed worse.

Without shuffing (Kruskal) with shuffling (Kruskal)

metin, yazı tipi, ekran görüntüsü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, yazı tipi, ekran görüntüsü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

I used “density\_0\_adj\_matrix5000.txt”

4) I checked if I did any I/O operations in Kruskal algorithm but that wasn’t the problem also.

5) I also tried to optimize KruskalAlgo function by giving edge set to it as a parameter. It increased the performance but still it wasn’t enough. I kept this updated version in my code.

6) I checked if there is a memory leak but I fixed it with using malloc.

7) After all the check operations I couldn’t find the main reason for Kruskal’s bad performance.

References:

For Kruskal Algorithm:

<https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/>

For Prim’s Algorithm:

<https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/>

Prim’s Algorithm’s and Kruskal Algorithm’s Pseudocodes:

Levitin, A. (2014). Introduction to the Design and Analysis of Algorithms: International Edition

Pearson Higher Ed.

Name-Surname: Burak Eymen Çevik Student ID:20200702123