ANKARA YILDIRIM BEYAZIT UNIVERSITY

Faculty of Engineering and Natural Sciences
Department of Computer Engineering

The Shortest Path on Map

Group:
Group 8

Supervisor:
Asst. Prof. Fadi YILMAZ

CENG - 303 Design and Analysis of Algorithms

January 18, 2023

ETHICAL DECLARATION

We hereby declare that, in this Term Project which has been prepared,

- All data, information and documents are obtained in the framework of academic and ethical rules,
- All the materials that have been utilized are fully cited and referenced,

and in any contrary case of above statements, We accept to renounce all our legal rights.

Group Members:
20050151012 - Barış Şahin
20050111060 - Denis Nazım Hikmet Gerçek
19050111033 - Kübra Yıldız
20050151002 - Muhammed Yasin Kayan
19050111020 - Yakup Ergün

GitHub:
https://github.com/MuhammedAYBU2020/CENG303_Termproject_Group8

Overleaf (Report Read Only):
https://www.overleaf.com/read/zdxfkrkngpwp

ACKNOWLEDGEMENTS

Firstly, We would like to thank Asst. Prof. Fadi YILMAZ for everything he told us from the beginning to the end of the term, within the scope of CENG - 303 Design and Analysis of Algorithms, which is one of the 2022 - 2023 Academic Year Fall Semester courses.

Special thanks go to our friends who have contained the hectic moments and stress we have been through during the course of the research project.

We thank the Ankara Yildirim Beyazit University for giving us the grand opportunity to work as a team which has indeed promoted our team work spirit and communication skills. We also thank the individual group members for the good team spirit and solidarity.

January 18, 2023 **Group - 8**

FINDING SHORTEST PATH IN A LARGE SCALE GRAPH

Aim of the project is to find shortest path between two vertices in a very large graph using well-known algorithms. In this project US Road Map graph file available public [1] is used. Graph contains 129164 vertices and 165435 edges.

In the requirement analysis and design step of our work:

- We searched for the map datasets that stores the nodes in a graph representation
- We investigated the shortest path algorithms that we can use to apply on our graph
- We decided to select simple text based graph dataset that we can use straightforward and mostly focus on the shortest path finding on a large dataset.

In the first step of our program, edges in the text file is read and graph is created. Here we can briefly describe our graph data structure implementation:

- AdjacencyListNode has vertex and weight attriibute
- AddEdge function adds Node to the Adjacency list of the related node
- Graph is stored in a 2D ArrayList of AdjacencyListNode

Dijkstra Shortest Path algorithm is used to find the shortest distance and the path between given source and destination. Dijkstra algorithm calculates distance/path from one source to all destinations. Implementation of the function is changed to return the distance to one determined destination. And it also prints the path to that destination.

Details of the dijkstra function is as follows:

Parameters:

- V: number of vertices in the graph
- graph: adjacency list of the graph
- src: source node
- target: destination node

Returns:

- distance value to target

Algorithm:

- Uses distance array and parents array for all destinations
- Uses PriorityQueue of AdjacencyListNodes items
- Adds source node to queue with distance zero
- While queue contains an element
  - Dequeue an element as current
  - For all adjacent vertices of the current node
    - If distance to selected element is smaller (using this current)
      - Updated the distance array and parent array
      - Add this element to the queue with smaller distance
- If target is reachable from the source node
  - Prints the path to target using parents array by recursive distance function
- Return distance value to target

PrintPath helper function:

- function to print shortest path from src to dest using parents array
- Recursively call by parent of itself until reaching source node

BFS Implementation:

The code starts with the definition of a class called BFS, which has a private ArrayList of ArrayList of integers called "graph", which is used to store the graph. The class has several methods such as "addEdge", "addVertex", "getShortestPath", and "bfs", which are used to perform various operations on the graph.

The "addEdge" method is used to add an edge between two vertices in the graph, represented by the integers "i" and "j", respectively. The "addVertex" method is used to add a new vertex to the graph. The "getShortestPath" method is used to find the shortest path between two vertices in the graph, represented by the integers "s" and "dest" respectively, and the number of vertices in the graph "v".

The "bfs" method is a modified version of the BFS algorithm that stores the predecessor of each vertex in an array called "pred" and its distance from the source in an array called "dist". This method takes the following parameters: "src", "dest", "v", "pred", and "dist".

The code uses a queue, implemented as a LinkedList, to maintain the queue of vertices whose adjacency list is to be scanned as per the normal BFS algorithm. A boolean array called "visited" is also used to store the information of whether a vertex has been visited or not.

In the while loop, the code uses the remove() method to remove the first vertex from the queue, and then iterates through all its adjacent vertices, marking them as visited and updating their distances and predecessors if they haven't been visited before. The stopping condition of the while loop is when the destination vertex is reached.

SAMPLE RUN OF THE TERM PROJECT

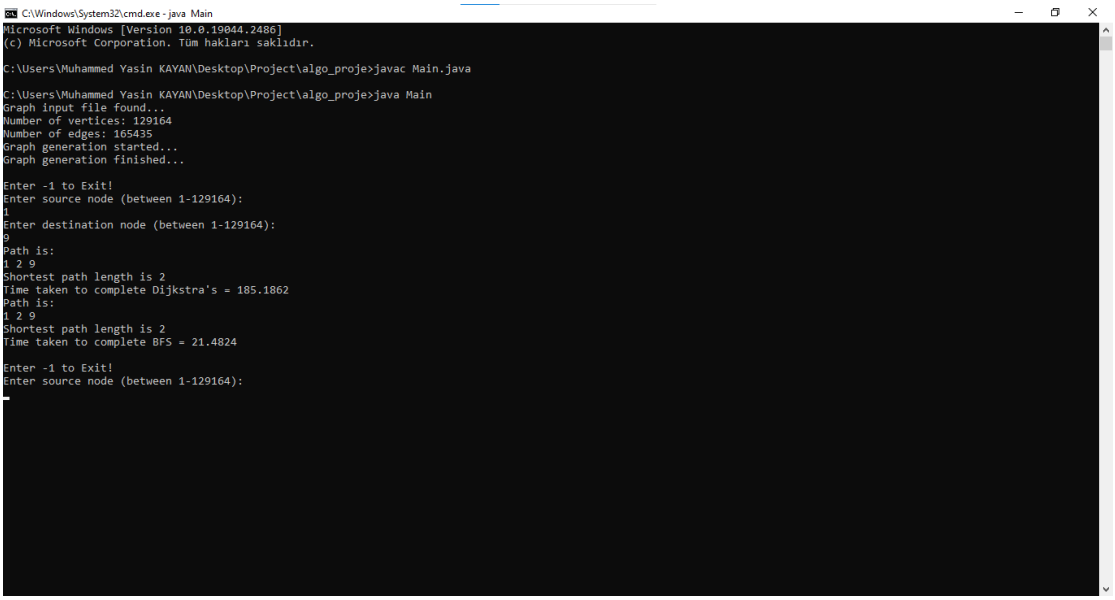

Figure 1: Screenshot_1.png

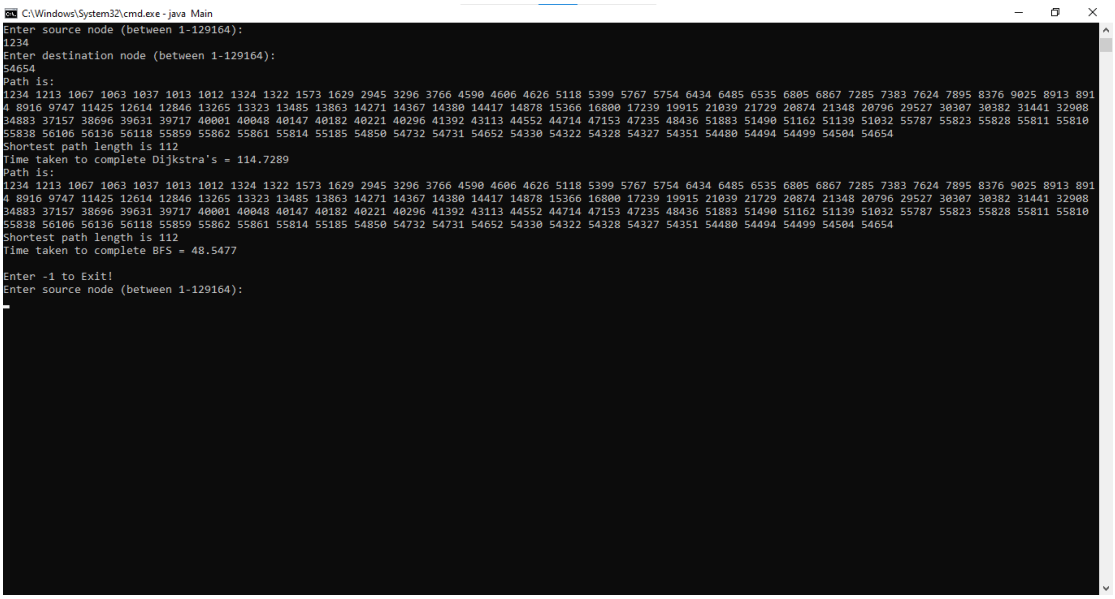

Figure 2: Screenshot_2.png

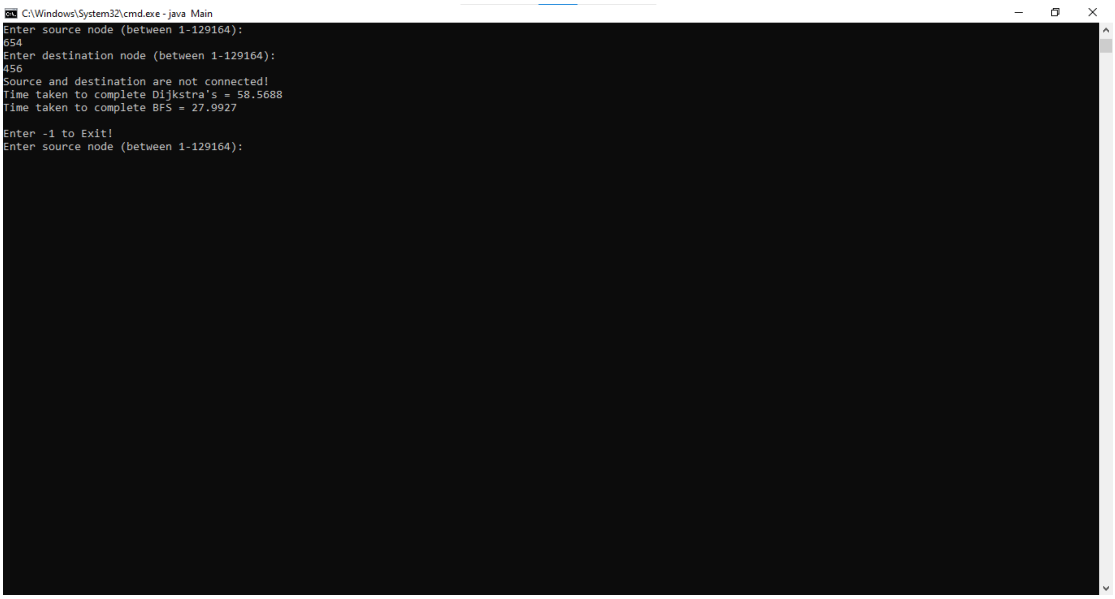

Figure 3: Screenshot_3.png

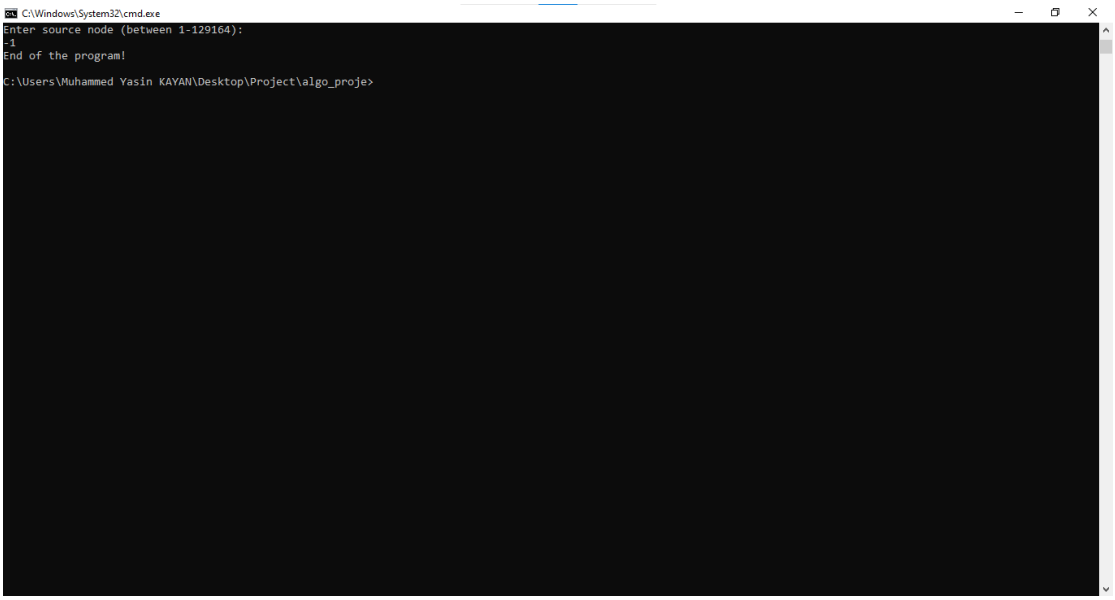

Figure 4: Screenshot_4.png

REFERENCES

[1] https://networkrepository.com/road-usroads.php