MODELLING GOOGLE MAP USING DIJKSTRA'S ALGORITHM

A PROJECT REPORT

BACHELOR OF TECHNOLOGY IN

INFORMATION TECHNOLOGY

Submitted by:
YAHYA BARE HAJON (2K19/IT/145)
BRUNO ADUL OMONDI(2K18/IT/038)

Under the Supervision of
Ms. Swati Sharda,
Faculty supervisor

DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of
Engineering) Bawana Road, Delhi
110042
DECEMBER, 2020
DELHI TECHNOLOGICAL UNIVERSITY

CANDIDATE’S DECLARATION

We hereby declare that project report entitled **“MODELLING GOOGLE MAP USING DIJKSTRA’S ALGORITHM”** implemented using Java submitted by (Yahya Bare Hajon, 2K19/IT/145) and (Bruno Adul, 2K18/IT/038) to Delhi Technological University (DTU), Delhi is a record of original work done under the guidance of Prof. Swati Sharda for the course of **DISCRETE STRUCTURES**. All the codes and implementations are completely written by us.

Yahya Bare Hajon (2K19/IT/145)

Bruno Adul Omondi (2K18/IT/038)

Date: 2nd Dec 2020

Place: New Delhi

DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering) Bawana Road, Delhi –110042

ACKNOWLEDGMENT

We would like to express our sincere gratitude to our teacher, **Ms.Swati sharda** for providing her inevitable guidance,comments and suggestions that helped us in doing a lot of research and accomplishing this mini project.

Also ,we would like to express our deepest appreciation towards all the resources that have provided us the possibility to make progress in our project.

ABSTRACT

Google Maps is based on a very simple but incredibly effective algorithm: **the Dijkstra algorithm**. It takes its name from its inventor, Edsger Dijkstra, one of the pioneering founders of modern computing.

A walk destined to change history. It was a morning in 1956 and Dijkstra, who at the time worked as a programmer at the Centrum Wiskunde & Informatica (CWI) in Amsterdam, was walking with his girlfriend to do some shopping. When they got tired of walking they sat at a coffee shop, the Dutch scientist had an illumination: in just 20 minutes, in front of a cup of coffee, he designed the algorithm that would make him enter the history of computer science.

To understand how it works we need to introduce the concept of graphs. A graph, like the one represented in the figure, is a structure formed by nodes, represented by dots, and arcs, lines that connect the nodes. Although simple, this model can be used to describe many problems. For example, nodes can represent people and arcs connect those who know each other; in anatomy, the nodes can represent the organs and the arches the blood vessels that connect them, in astronomy, the stars are nodes that, joined together by the arches, form the constellations. In our case, instead, the arches represent the roads, while the nodes are the intersections, that is all the points where it is possible to choose which road to take.

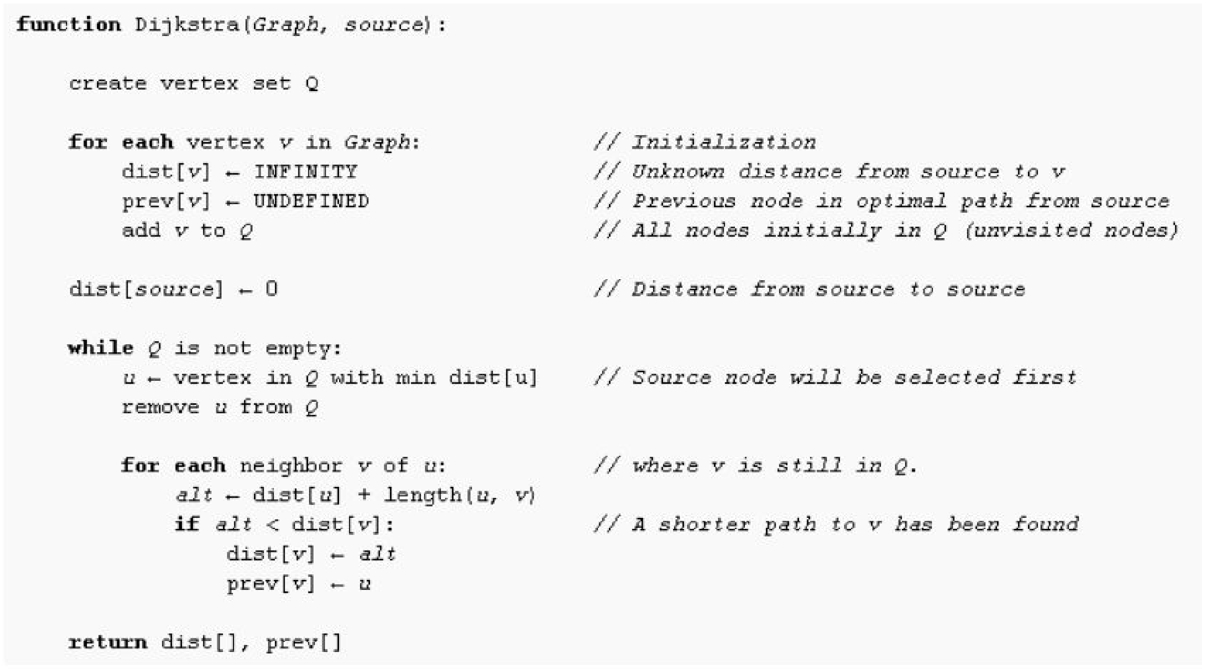
What does Dijkstra's algorithm do? Given a weighted graph, a starting point and an endpoint within the graph itself, the algorithm finds the "minimum path" that connects the two points, that is the sequence of arcs that minimizes the sum of the weights and therefore, in the case of Maps, minimizes the estimated travel time.
Dijkstra's algorithm always finds the fastest route. It has been shown mathematically that Dijkstra always finds the shortest path, as long as there is at least one possible route.

PROPOSAL

This project will tackle the common problem of modeling maps. When we use maps we always want to choose the fastest route or sometimes safest route. The entire premise of Google Maps is using a big giant graph with nodes and edges to figure out the fastest or shortest way to travel. That's all Google Maps is–a big graph with lots of nodes and edges.
So, we applied graph theory to solve these following common cases on our project:

- **The shortest path from city A to city B(using Dijkstra's algorithm)**

The Dijkstra's Algorithm pseudocode



```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:             // Initialization
        dist[v] ← INFINITY                  // Unknown distance from source to v
        prev[v] ← UNDEFINED                 // Previous node in optimal path from source
        add v to Q                          // All nodes initially in Q (unvisited nodes)

    dist[source] ← 0                        // Distance from source to source

    while Q is not empty:
        u ← vertex in Q with min dist[u]    // Source node will be selected first
        remove u from Q

        for each neighbor v of u:           // where v is still in Q.
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:               // A shorter path to v has been found
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

OBJECTIVE

In this short project, we implement Dijkstra’s algorithm. We have been provided with data and some code for a GUI, which generates a map of cities in the US and the different connections between them. Our goal is to compute for the user, the shortest path between any two cities they select on the map.

Introduction

Given two points on a graph, the shortest path between them is the straight line that joins them. However, drawing a straight line may not always be possible. For example, in a road network, straight roads between two cities seldom exist. In such a case, there might be multiple routes between the two cities, and the challenge then is to find the shortest one.

Dijkstra’s algorithm helps us find the shortest path between a source node and every other node in a graph.

The algorithm is ubiquitous. For example, it is used in computer networking where it ascertains the shortest path between the source router and other routers in the network. Also, each time we are navigating via google maps, complex algorithms based on Dijkstra are being used to provide us with the best route.

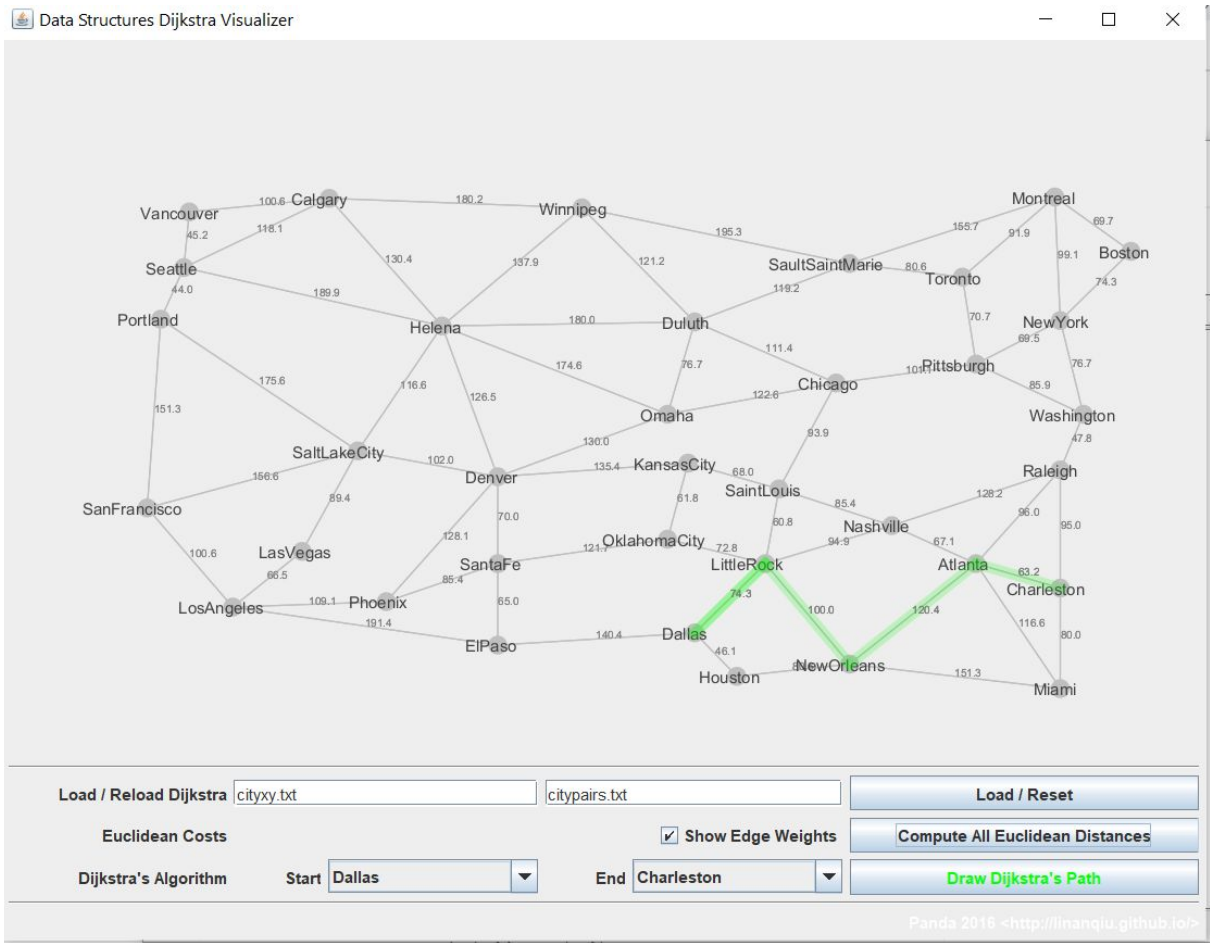
CODE IMPLEMENTATION

We implemented Dijkstra’s algorithm to find shortest paths between pairs of cities on a map using java.We are providing a GUI that lets you visualize the map and the path that the algorithm finds between two cities.

We have two classes Vertex.java and Edge.java provided in the code,, which represent the vertices and edges of a graph. we will not have to modify these classes.

we only have to modify Dijkstra.java, which represents a graph and will contain the implementation of Dijkstra’s algorithm. we need to use the instance variable vertexNames, which contains a mapping from city names to Vertex objects after the graph is read from the data files. The main method of Dijkstra illustrates how the class is used by the GUI and might be useful for testing the implementation on the command line.

RESULT



cityxy.txt

citypairs.txt

✓ Show Edge Weights

Start Dallas

End Charleston

The display shows the dijkstra's path between two cities i.eDallas(start) and Charleston(end/destination).

References

https://en.wikipedia.org/wiki/Graph_theory
https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
https://medium.com/@sukhrobgolibboev/modeling-google-maps-using-graph-theory-b7e90a6cf3e0