****

**Algoritme dhe Struktura të të Dhënave**

**“Shortest Path”**

Studente: Prof.:

Edone Haziri Edmond Jahjaga

Ariola Cerkini

**Content**

Content 2

What is ‘Shortest Path’ Algorithm 3

Types of ‘Shortest Path’ Algorithm 3

Directed – Undirected Graph 3

Shortest path problem examples 3

Types of shortest path algorithms 4

Dijkstra’s 5

How Dijsktra works 6

Implementation of code 9

**SHORTEST PATH**

In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. We can categorize finding the shortest path in three ways.

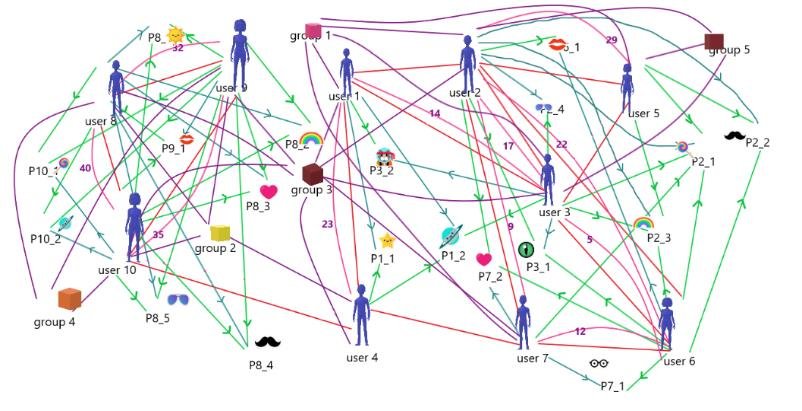
The single-source shortest path problem, in which we have to find shortest paths from a source vertex v to all other vertices in the graph.

The single-destination shortest path problem, in which we have to find shortest paths from all vertices in the directed graph to a single destination vertex v. This can be reduced to the single-source shortest path problem by reversing the arcs in the directed graph.

The all-pairs shortest path problem, in which we have to find shortest paths between every pair of vertices v, v' in the graph.

A directed graph is a type of graph that contains ordered pairs of vertices while an undirected graph is a type of graph that contains unordered pairs of vertices which also indicates a two-way relationship, in that each edge can be traversed in both directions.

Shortest path can be used to solve various different problems, such as in social networking applications where we can regularly see suggested lists of friends that we may know. Flighting agendas, where you can see all airports and flights, departures and arrivals. Digital mapping services, where you find the shortest route from one city to another, or from our location to the nearest desired. Or even robotic paths, where you can use robots or drones to deliver packages to a specific location by using the shortest path in a minimum amount of time.



Some of the most used algorithms for shortest path:

Dijkstra's algorithm - solves the single-source shortest path problem with non-negative edge weight.

Bellman–Ford algorithm - solves the single-source problem if edge weights may be negative.

Floyd–Warshall algorithm - solves all pairs shortest paths.

Today we will be using Dijkstra’s algorithm to find the shortest path between the current location and the destination of an example we took ourselves.

**DIJKSTRA’S ALGORITHM**

Dijkstra's algorithm (/ˈdaɪkstrəz/ DYKE-strəz) is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.

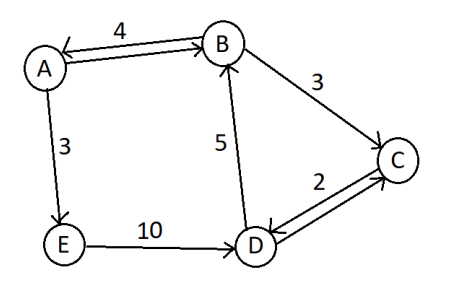


Cornerstone of Edsger W. Dijkstra’s fame: Dijkstra’s algorithm

That cornerstone is the algorithm for the shortest path, also known as Dijkstra's algorithm. According to Dijkstra, 'it was a twenty-minute invention' that he conceived while having a cup of coffee on a café terrace with his fiancée Ria, who he had met at Mathematisch Centrum.

**How Dijkstra works:**

We take a graph with weight and direction:

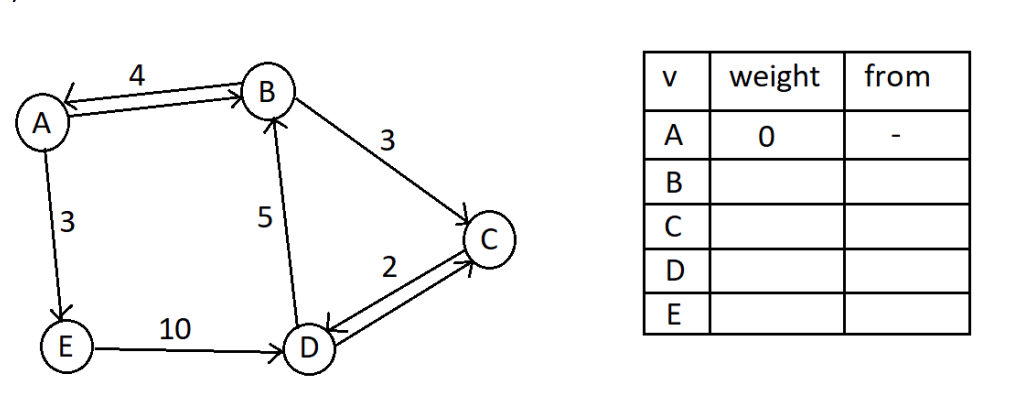


Select a starting vertex. - A

View all vertices starting from the starting vertex in ascending order.

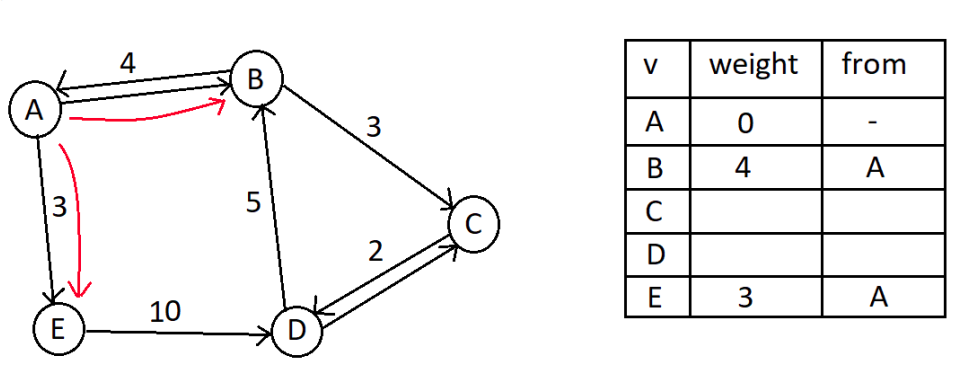
Add a vertex to the tree and relax all the edges coming from that vertex.

1)



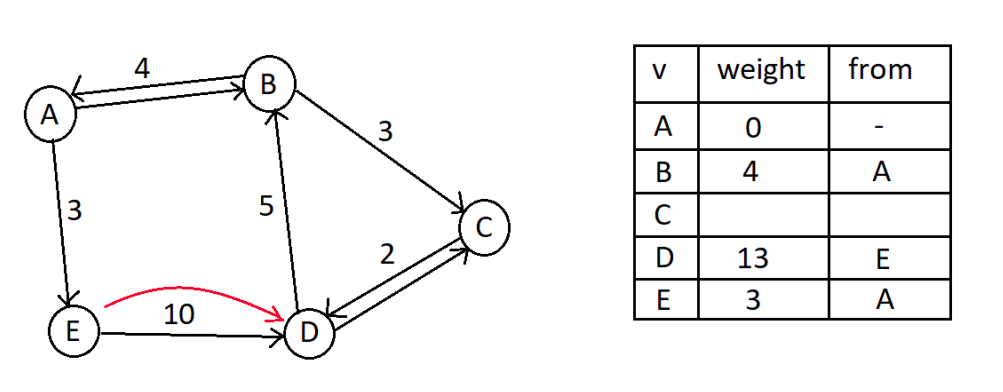
Visited: A

2)





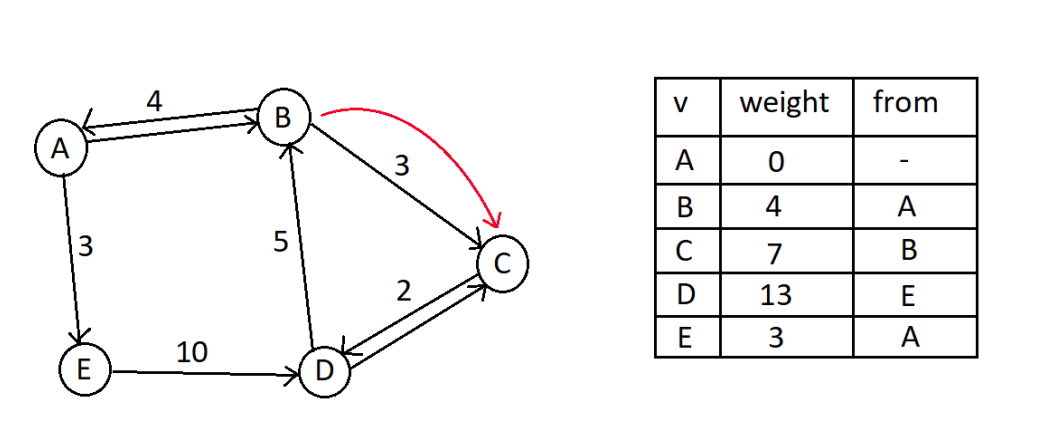
3) Choosing E:



Distance from A to D is 13.

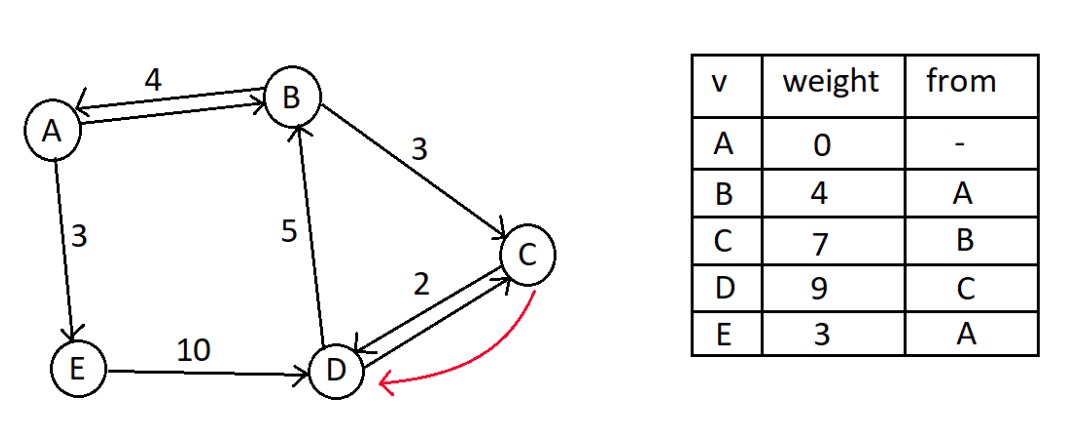
Visited: A,E.

4) Choosing B:



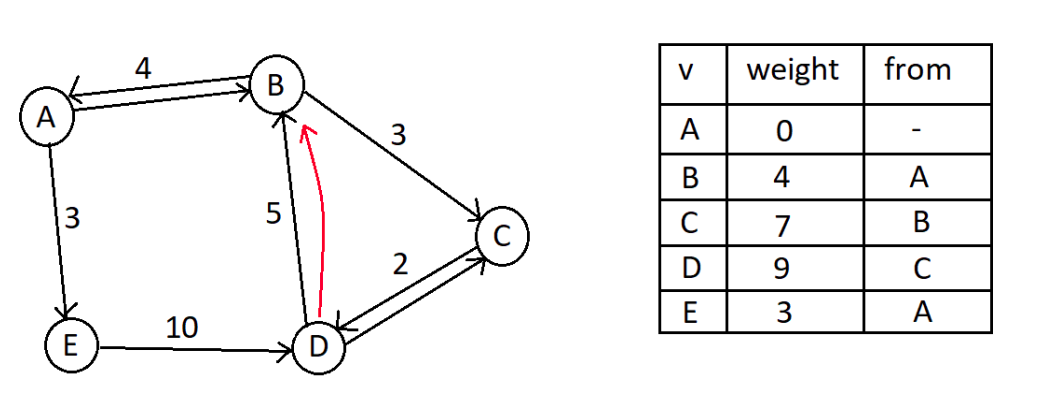
Visited: A,E,B.

5) Choosing C:

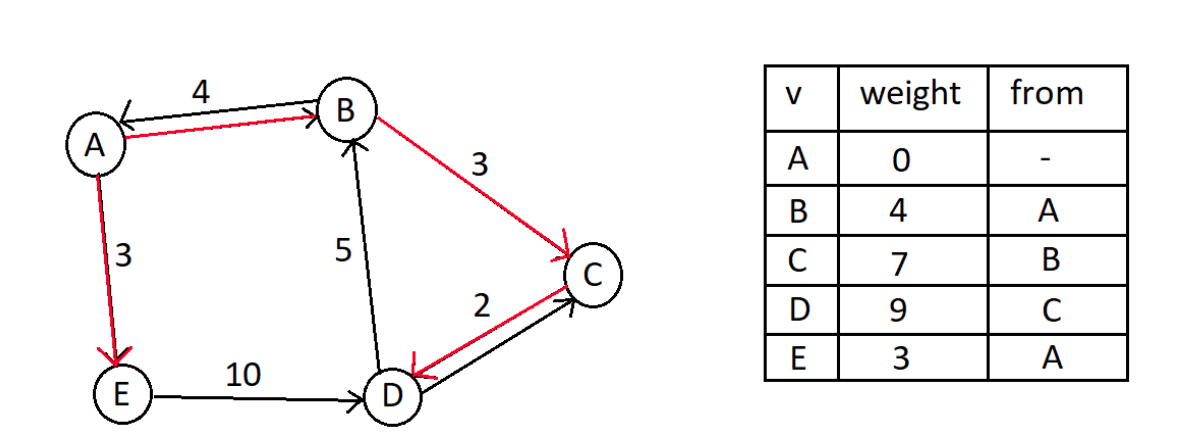


Visited: A,E,B,C – D is updated with new value.

6) Choosing D:

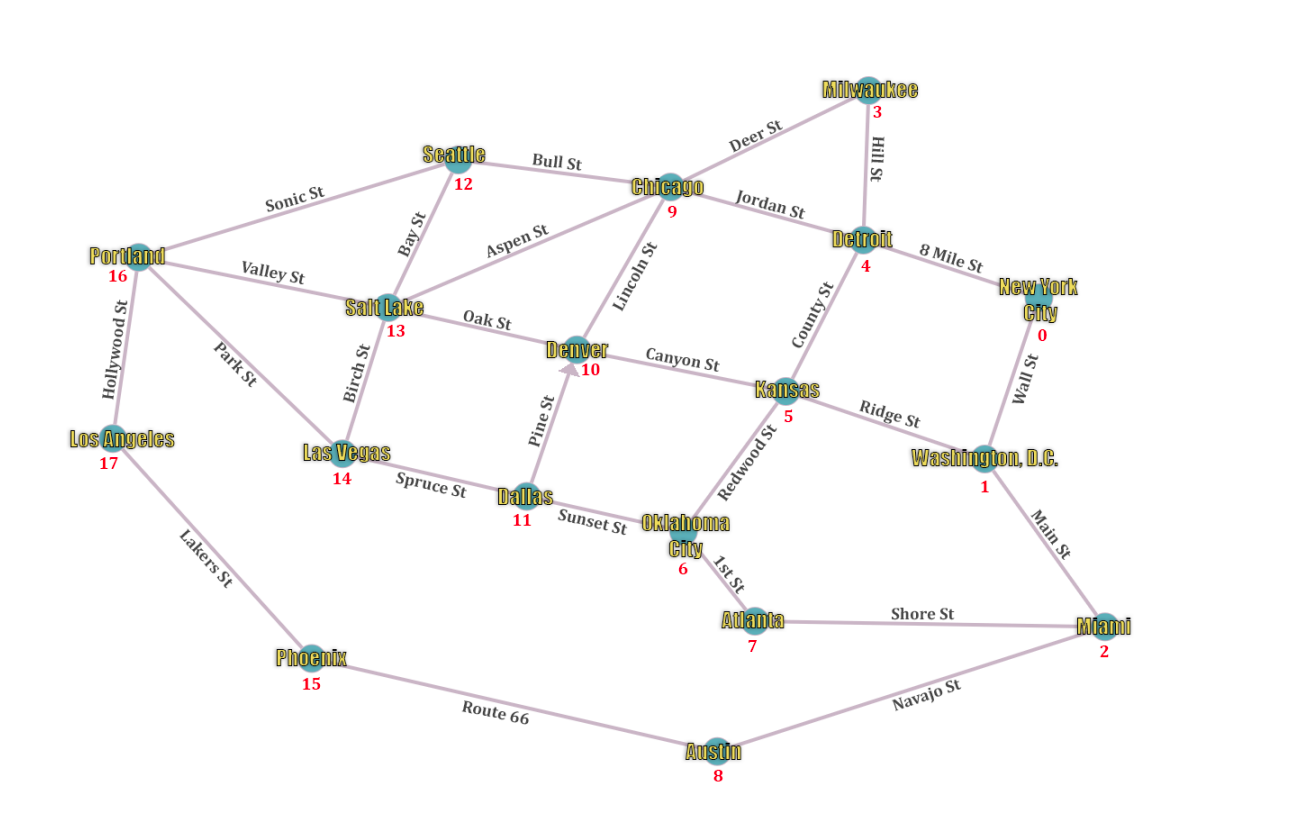


No changes. Visited: A,E,B,C,D

7) We now found the shortest path from A to the other vertices. 

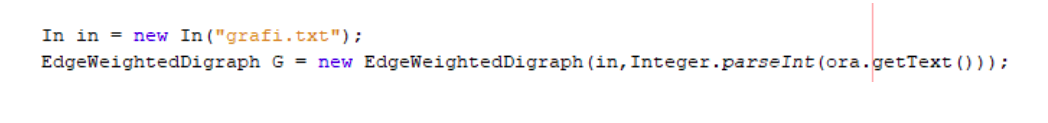
**IMPLEMENTATION**

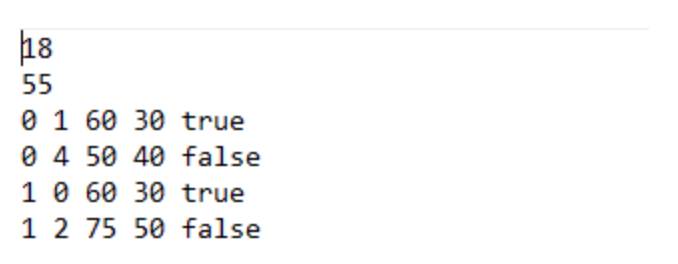
Finding the fastest route in a traffic network using Dijkstra's Algorithm. We did the implementation by using Java. We modified the code to find the fastest route to the desired destination. We have created a graph with several cities in the USA where: the vertices represent the locations, the nodes represent the roads, the weight represents the time and will be calculated from the average distance/speed of that route. Some of the roads will have rush hour at certain times which will increase the weight/time of that road.



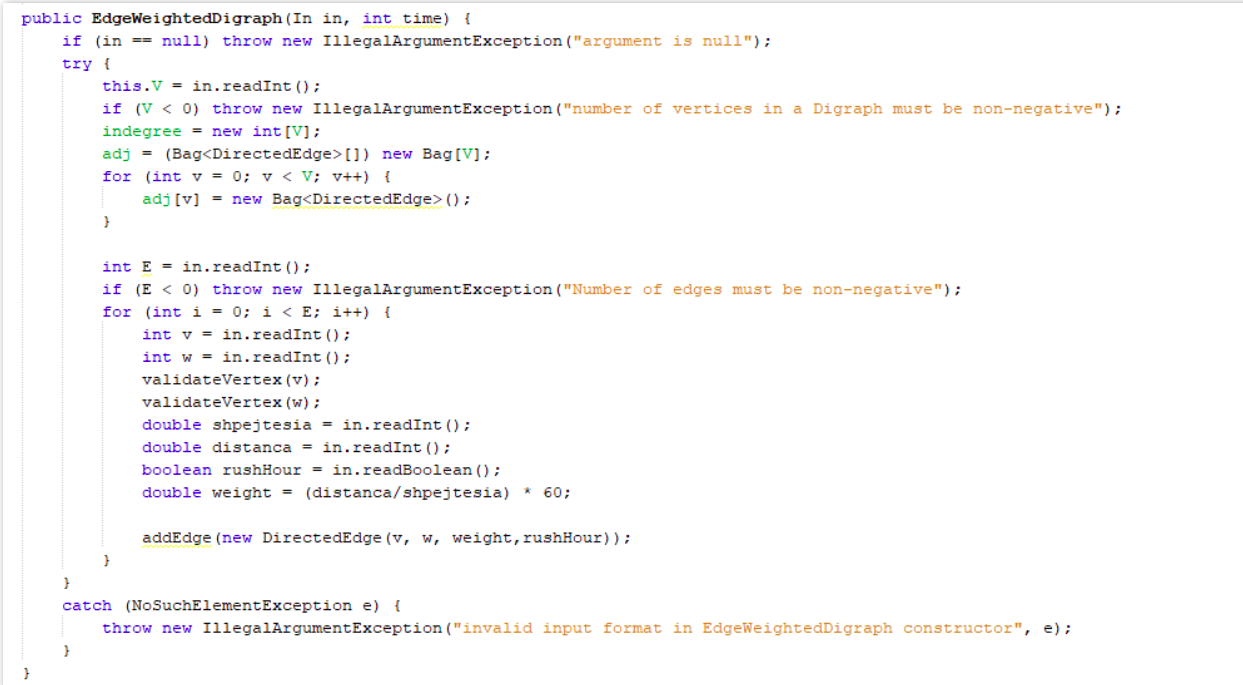
The graph will be read from the file in this format:

KulmiFrom, KulmiTo, Shpejtesia, Distanca, RushHour.

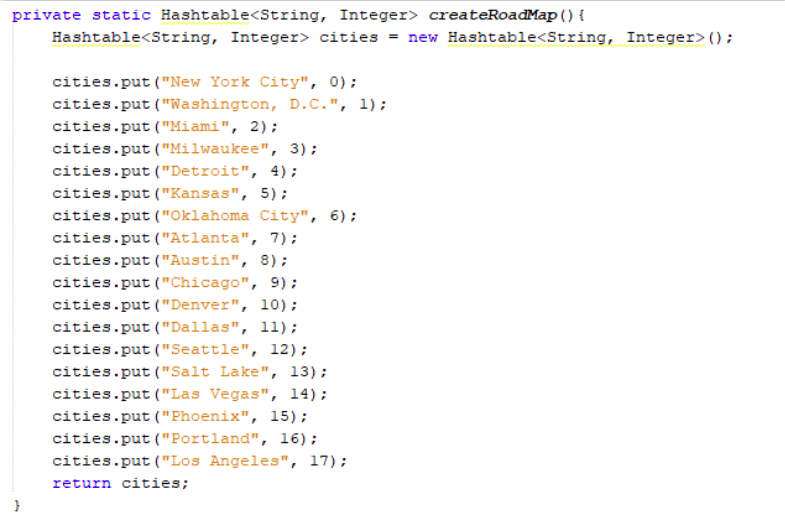




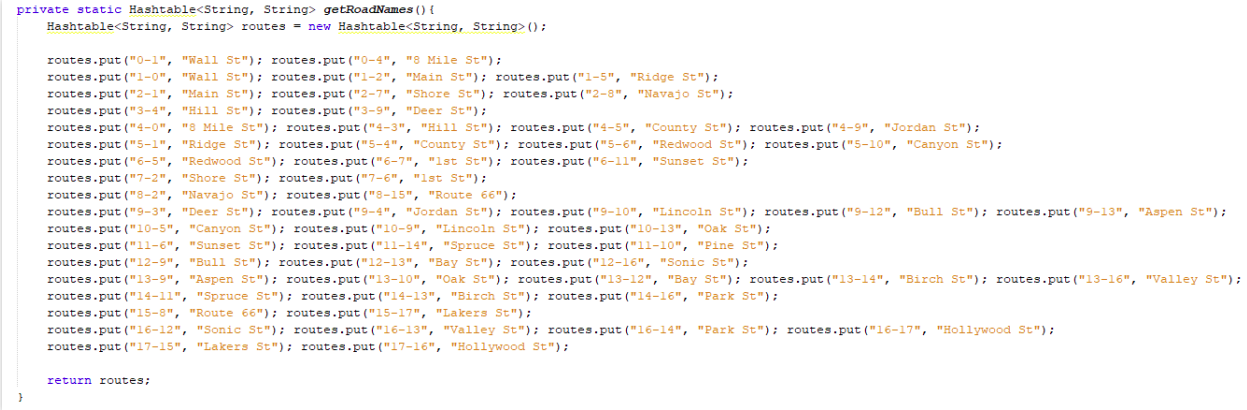
The coding of the graph:



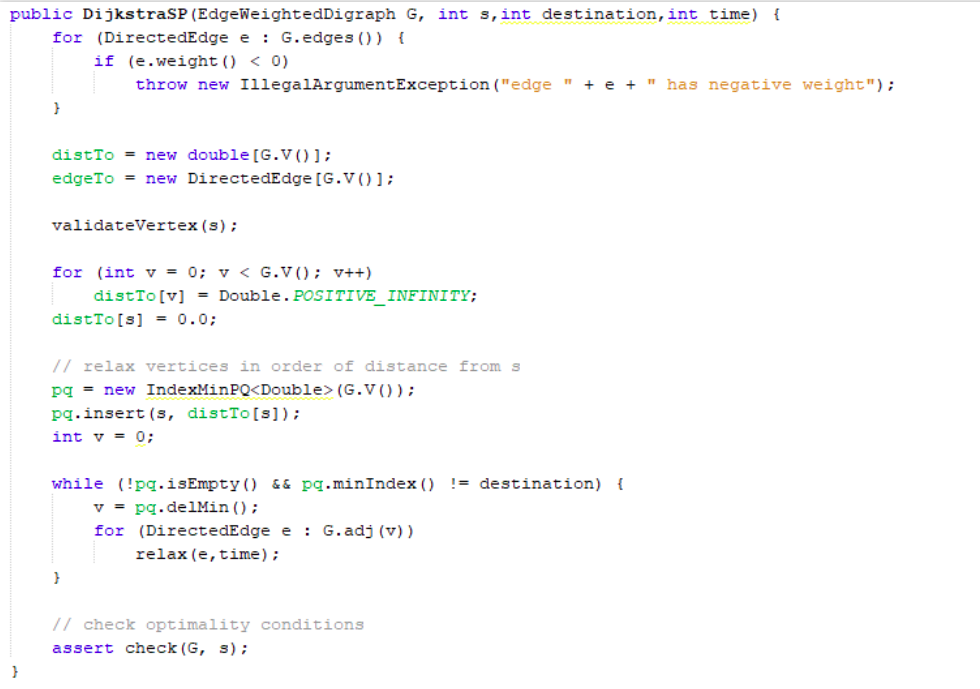
Naming of the vertices/locations:

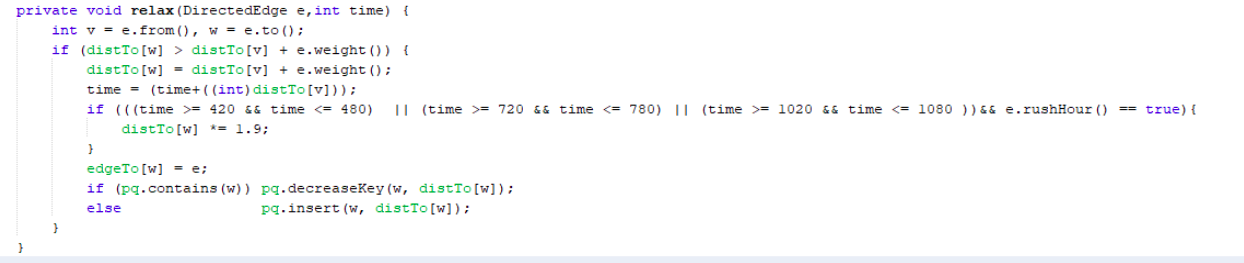


Naming of the roads:

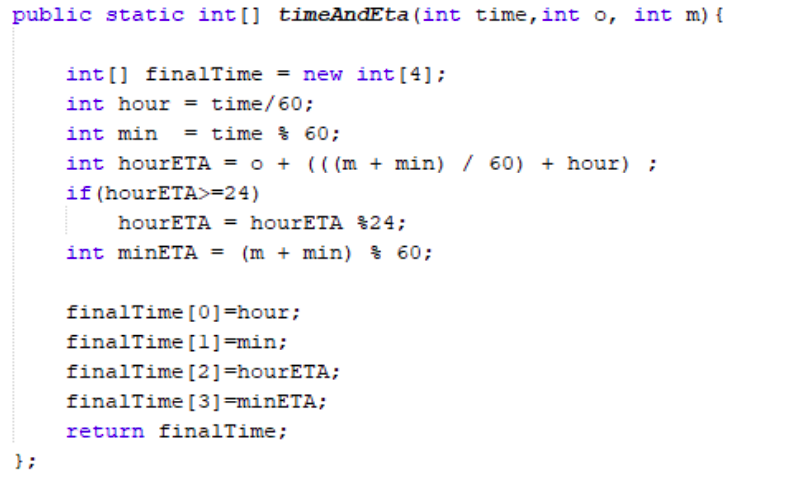


Llogaritja e rruges me te shpejte:





Llogaritja e kohes:



Output of the code:

