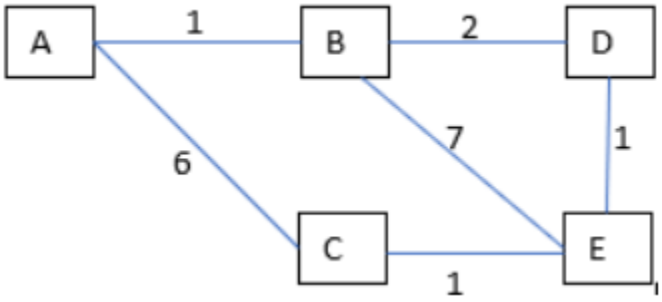
Balaji Paiyur Mohan
1001576836

Structure of code:

1. Firstly we are feeding the graph inputs in a text file “input.txt”
2. Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Adjacency Matrix is used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w. In our program, Adjacency matrix comes out to be

```
0 1 6 0 0
1 0 0 2 7
6 0 0 0 1
0 2 0 0 1
0 7 1 1 0
```



3. Data Structure of this Dijkstra Algorithm is done by initializing three values:

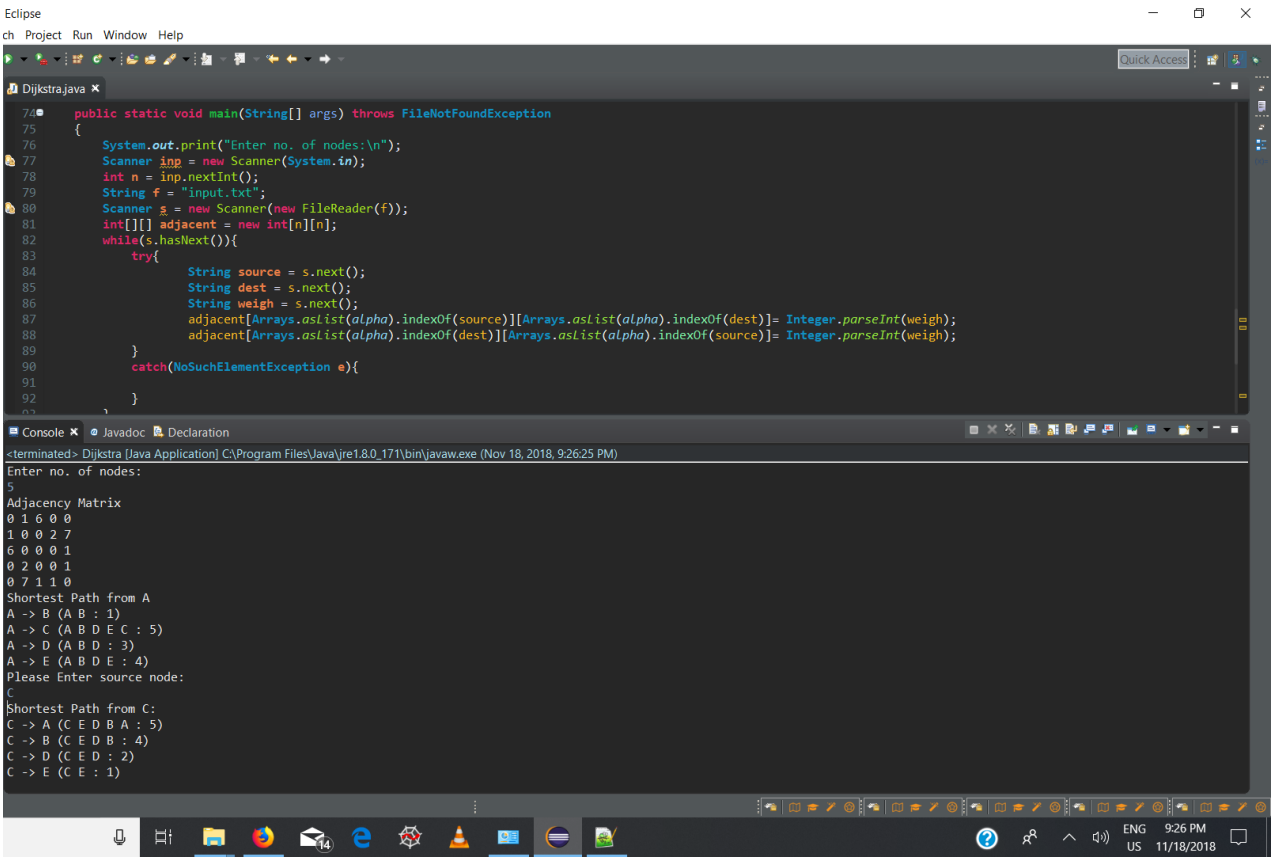
    - min_dist, an array of distances from the source node to each node in the graph, initialized the following way: min_dist(s) = 0; and for all other nodes v, min_dist(v) = infinity. This is done at the beginning because as the algorithm proceeds, the from the source to each node in the graph will be recalculated and finalized when the shortest distance to v is found.
    - Parents, an array of all parent nodes in the graph. At the end of the algorithm's progress, all parent nodes are stored while nodes without parents are initialized to -1.
    - Visited , an empty list, to indicate which nodes the algorithm has visited. At the end of the algorithm's run, will contain all the visited nodes of the graph.

4. The algorithm proceeds as follows:

    a. While is not empty, pop the node v, that is not already in Visited, from with the smallest min_dist(v). In the first run, source node s will be chosen because dist(s) was initialized to 0. In the next run, the next node with the smallest min_dist value is chosen.
    b. Add node v to Visited, to indicate that has been visited
    c. Update dist values of adjacent nodes of the current node v as follows: for each new adjacent node u.
        - if min_dist(v) + weight(u,v) < min_dist(u), there is a new minimal distance found for , so update min_dist(u) to the new minimal distance value;
        - otherwise, no updates are made to min_dist(u).

5. The algorithm has visited all nodes in the graph and found the smallest distance to each node now contains the shortest path tree from source s.
6. Output function retrieves the shortest distances from min_dist array and get_path function retrieves the traced path from the elements in parents array.

Screenshot of working code:



Time Complexity:

- Dijkstra Algorithm function runs with time complexity of ***O(n²)*** as we could see a nested for loop.
- Output function runs with time complexity of ***O(n)*** as perform search in an array
- Get_path function runs with time complexity of ***O(n)*** as it calls itself recursively n times.

So, Overall we are interested in the largest value which happens to be ***O(n²).***

Real life example:

Google Maps! It uses more complex and efficient algorithms but Dijkstras Algorithm is the basis. It's also used in finding a shortest communication path between two nodes connected in a network .

Example: https://www.google.com/maps

Balaji Paiyur Mohan
1001576836

References:

- https://brilliant.org/wiki/dijkstras-short-path-finder/