Code for Dijkstra's Algorithm

import java.util.\*;

public class EXPT\_04\_Dijkstra {

    static final int n = 6;

    int minimum\_distance\_from\_source(int distance[], Boolean visited[]) {

        int min = Integer.MAX\_VALUE, min\_index = -1;

        for (int v = 0; v < n; v++)

            if (visited[v] == false && distance[v] <= min) {

                min = distance[v];

                min\_index = v;

            }

        return min\_index;

    }

    void print\_solution(int distance[], int n) {

        System.out.print("Vertex    Distance from Source");

        for (int i = 0; i < n; i++)

            System.out.print("\n" + i + "\t\t  " + distance[i]);

    }

    void dijkstra(int graph[][], int source) {

        int distance[] = new int[n];

        // distance of n vertices from the source

        Boolean visited[] = new Boolean[n];

        // to check if each one of the n vertices has been visited or not

        for (int i = 0; i < n; i++) {

            distance[i] = Integer.MAX\_VALUE;

            // initializing the distance of n vertices from source to be infinity

            visited[i] = false;

            // initializing the truth value of visit of n vertices to be false

        }

        distance[source] = 0;

        // distance of source vertex from itself is 0

        for (int count = 0; count < n - 1; count++) {

            // finding shortest path for each vertex

            int u = minimum\_distance\_from\_source(distance, visited);

            // finding the minumum distance of uth vertex from source

            visited[u] = true;

            // uth vertex has been visited

            for (int v = 0; v < n; v++)

                if (!visited[v] && graph[u][v] != 0 && distance[u] != Integer.MAX\_VALUE

                        && distance[u] + graph[u][v] < distance[v])

                    distance[v] = distance[u] + graph[u][v];

        }

        print\_solution(distance, n);

    }

    public static void main(String args[]) {

        int graph[][] = new int[][] { { 0, 2, 4, 0, 0, 0 },

                { 2, 0, 1, 7, 0, 0 },

                { 4, 1, 0, 0, 3, 0 },

                { 0, 7, 0, 0, 2, 1 },

                { 0, 0, 3, 2, 0, 5 },

                { 0, 0, 0, 1, 5, 0 } };

        EXPT\_04\_Dijkstra obj = new EXPT\_04\_Dijkstra();

        int source\_index = 0;

        long start = System.currentTimeMillis();

        obj.dijkstra(graph, source\_index);

        long end = System.currentTimeMillis();

        long time = end - start;

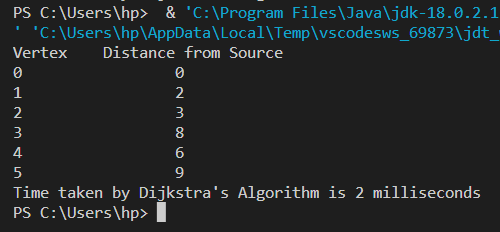
        System.out.print("\nTime taken by Dijkstra's Algorithm is " + time + " milliseconds");

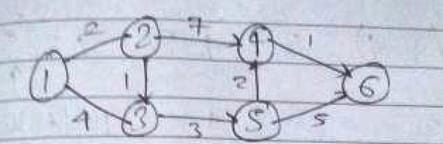
    }

}

Case 1: When the number of vertices is 6

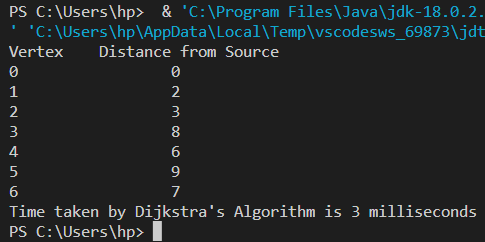
Time taken is 2 milliseconds.

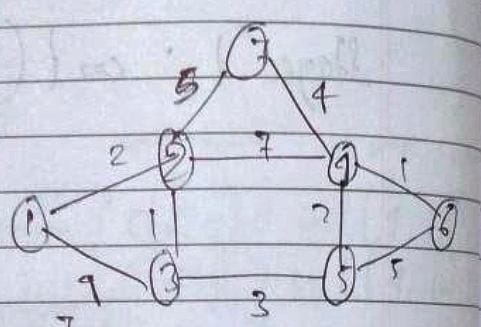




Case 2: When the number of vertices is 7

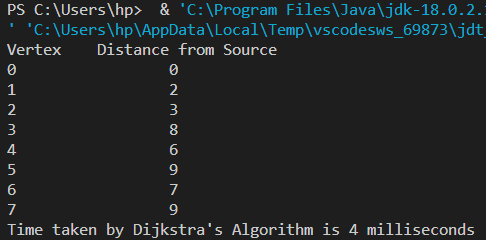
Time taken is 3 milliseconds.

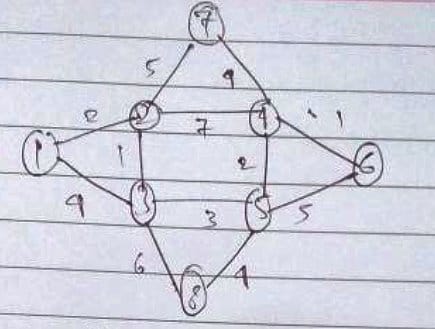




Case 3: When the number of vertices is 8

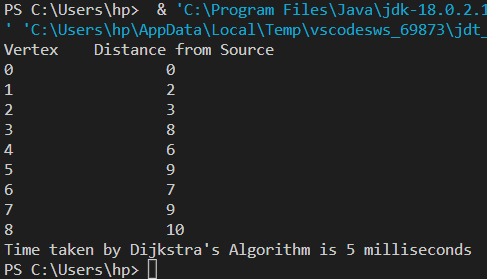
Time taken is 4 milliseconds.

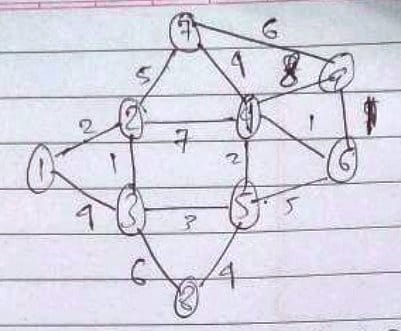




Case 4: When the number of vertices is 9

Time taken is 5 milliseconds.





Case 5: When the number of vertices is 10

Time taken is 6 milliseconds.

