Name: Vinayak Madan Shete

Roll No.: TYCOC303

Div: C Batch: C4

Course Name: Design and Analysis of Algorithms Laboratory

Course Code: BCE5412

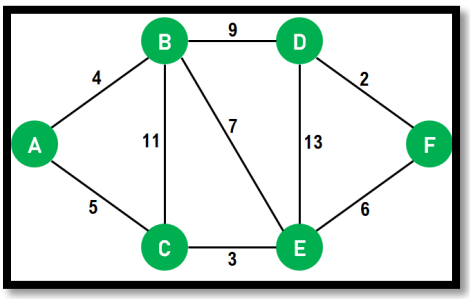
==========

Assignment 04:Implementing Dijikstra's shortest path algorithm.

==========

Input:

Graph:




```
import java.util.*;
dijkstra input
public class Dijkstra {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Total vertices : ");
        int vertices = sc.nextInt();
        System.out.print("Total edges : ");
        int edges = sc.nextInt();
        int[][] graph = new int[vertices][vertices];
        System.out.println("Enter edges : ");
        System.out.println("Node 1 | Node 2 | Weight");
        for(int i=0; i<edges; i++){
```

```
            int node1 =sc.nextInt();
            int node2 = sc.nextInt();
            int weight = sc.nextInt();
            graph[node1][node2] = weight;
        }
        System.out.println("Graph is : ");
        for(int i=0;i<vertices;i++){
            for(int j=0; j<vertices; j++){
                System.out.print(graph[i][j] + " ");
            }
            System.out.println();
        }
        System.out.print("Enter source vertex : ");
        int sourceVertex = sc.nextInt();
        dijkstraAlgorithm(graph,sourceVertex);
    }
    private static void dijkstraAlgorithm(int[][] graph, int sourceVertex) {
        int infinity = Integer.MAX_VALUE;
        int dist[] = new int[graph.length];
        Arrays.fill(dist,infinity);
        ArrayList <Integer> sequenceArray = new ArrayList<>();
        boolean[] visited = new boolean[graph.length];
        dist[sourceVertex] = 0;
        for(int i=0;i<graph.length;i++){
        int nextVertex = findNextMin(graph,dist,visited,sequenceArray);
        System.out.println("Next min vertex : "+nextVertex);
        visited[nextVertex] = true;
        sequenceArray.add(nextVertex);
        for (int j = 0; j < graph.length; j++) {
            if (!visited[j] && graph[nextVertex][j] != 0 && dist[nextVertex]
!= infinity &&
                    dist[nextVertex] + graph[nextVertex][j] < dist[j]) {
                dist[j] = dist[nextVertex] + graph[nextVertex][j];
            }
        }
```

```
    }
        System.out.println("Vertex | Dist from SourceVertex");
        for(int i=0;i<graph.length;i++){
            System.out.println(i+ " " +dist[i]);
        }
        System.out.println("Sequence : "+sequenceArray);
    }
    private static int findNextMin(int[][] graph, int[] dist, boolean[]
visited,ArrayList<Integer> sequenceArray) {
        int nextVertex = -1;
        for(int i=0;i<graph.length;i++){
            if(!visited[i] && (nextVertex==-1 || dist[i] <
dist[nextVertex])){
                nextVertex = i;
            }
        }
        return nextVertex;
    }
}
```

Output:

```
D:\PCCOE\Semester5\DAA\Practicals>java Dijikstra
Total vertices : 6
Total edges : 9
Enter edges :
Node 1 | Node 2 | Weight
0 1 4
0 2 5
1 2 11
1 3 9
1 4 7
3 4 13
2 4 3
3 5 2
4 5 6
Graph is :
0 4 5 0 0 0
0 0 11 9 7 0
0 0 0 0 3 0
0 0 0 0 13 2
0 0 0 0 0 6
0 0 0 0 0 0
Enter source vertex : 0
Next min vertex : 0
Next min vertex : 1
Next min vertex : 2
Next min vertex : 4
Next min vertex : 3
Next min vertex : 5
Vertex | Dist from SourceVertex
0 0
1 4
2 5
3 13
4 8
5 14
Sequence : [0, 1, 2, 4, 3, 5]
```

==========