Experiment - 8

Student Name: Hemant Narain Jha **UID:** 23BCS10022
Branch: BE-CSE **Section/Group:** KRG-2B
Semester: 5th **Date of Performance:** 16/10/25
Subject Name: Design and Analysis of Algorithms
Subject Code: 23CSH-301

1. **Aim:** Develop a program and analyze complexity to find shortest paths in a graph with positive edgeweights using Dijkstra‟s algorithm.
2. **Objective:** Code and analyze to find shortest paths in a graph with positive edge weights using Dijkstra‟s
3. **Input/Apparatus Used:** Graph (G =(V,E)) is taken as input for this problem.

4. **Procedure:**

Follow the steps below to solve the problem:

- Create a set sptSet (shortest path tree set) that keeps track of vertices included in the shortest- path tree, i.e., whose minimum distance from the source is calculated and finalized. Initially, this set is empty.
- Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign the distance value as 0 for the source vertex so that it is picked first.
- While sptSet doesn‟t include all vertices
- Pick a vertex u which is not there in sptSet and has a minimum distance value.
- Include u to sptSet.
- Then update distance value of all adjacent vertices of u.
- To update the distance values, iterate through all adjacent vertices.

- For every adjacent vertex v, if the sum of the distance value of u (fromsource) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

5. Algorithm

- **Step 1:** SET STATUS = 1 (ready state) for each node in G
- **Step 2:** Push the starting node A on the stack and set its STATUS = 2 (waiting state)
- **Step 3:** Repeat Steps 4 and 5 until STACK is empty
- **Step 4:** Pop the top node N. Process it and set its STATUS = 3 (processed state)
- **Step 5:** Push on the stack all the neighbours of N that are in the ready state (whose STATUS = 1) and set their
  STATUS = 2 (waiting state)
  [END OF LOOP]
- **Step 6:** EXIT

6. Code and Output:

```
import java.util.*;
public class DijkstraAlgorithm {
static final int INF = Integer.MAX_VALUE;
// Find the vertex with the minimum distance value
public static int minDistance(int[] dist, boolean[] sptSet, int V) {
int minVal = INF, minIndex = -1;
for (int v = 0; v < V; v++) {
if (!sptSet[v] && dist[v] <= minVal) {
minVal = dist[v];
minIndex = v;
}
}
return minIndex;
}
// Print the shortest distance from source
public static void printSolution(int[] dist, int V) {
System.out.println("\nVertex\tDistance from Source");
for (int i = 0; i < V; i++) {
if (dist[i] == INF)
System.out.println(i + "\t\tINF");
else
```

```
System.out.println(i + "\t\t" + dist[i]);
}
}
// Dijkstra's Algorithm
public static void dijkstra(int[][] graph, int src, int V) {
int[] dist = new int[V]; // The output array: dist[i] holds the shortest distance from src to i
boolean[] sptSet = new boolean[V]; // True if vertex i is included in shortest path tree
Arrays.fill(dist, INF);
dist[src] = 0;
for (int count = 0; count < V - 1; count++) {
int u = minDistance(dist, sptSet, V);
sptSet[u] = true;
for (int v = 0; v < V; v++) {
if (!sptSet[v] && graph[u][v] != 0 && dist[u] != INF &&
dist[u] + graph[u][v] < dist[v]) {
dist[v] = dist[u] + graph[u][v];
}
}
}
printSolution(dist, V);
}
// Main method
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.println("DIJKSTRA'S ALGORITHM - SHORTEST PATH FINDER\n");
System.out.print("Enter number of vertices: ");
int V = sc.nextInt();
int[][] graph = new int[V][V];
System.out.println("\nEnter the adjacency matrix (0 for no edge):");
for (int i = 0; i < V; i++) {
for (int j = 0; j < V; j++) {
graph[i][j] = sc.nextInt();
}
}
System.out.print("\nEnter source vertex (0 to " + (V - 1) + "): ");
int src = sc.nextInt();
dijkstra(graph, src, V);
sc.close();
}
}
```

Output

```
DIJKSTRA'S ALGORITHM - SHORTEST PATH FINDER

Enter number of vertices: 5

Enter the adjacency matrix (0 for no edge):
0 10 0 5 0
0 0 1 2 0
0 0 0 0 4
0 3 9 0 2
7 0 6 0 0

Enter source vertex (0 to 4): 0

Vertex  Distance from Source
0       0
1       8
2       9
3       5
4       7
```