EXPERIMENT-8

Program:

WAP to implement Dijkstra's algorithm using c/c++ and write the complexity.

Pseudo code:

```
DIJKSTRA (G, w, s) // G is the graph, w is the weight function, s is the source node
  for each node v in G
    d[v] = infinity // d[v] is the distance from s to v
    p[v] = null // p[v] is the previous node in the shortest path from s to v
  d[s] = 0
  Q = a priority queue of nodes ordered by d
  while Q is not empty
    u = Q.extract_min() // remove and return the node with the smallest distance
    for each neighbor v of u
      if d[v] > d[u] + w(u, v) // if the distance can be improved
        d[v] = d[u] + w(u, v) // update the distance
        p[v] = u // update the previous node
        Q.decrease_key(v, d[v]) // update the priority queue
  return d, p
```

Input:

```
#include <stdio.h>
#include <stdbool.h>
#include <limits.h>
#define MAX_VERTICES 10
int minDistance(int dist[], bool sptSet[], int vertices) {
    int min = INT_MAX, min_index;
    for (int v = 0; v < vertices; v++) {
        if (!sptSet[v] && dist[v] < min) {
            min = dist[v];
            min_index = v;
        }   }
    return min_index;
```

```
}
void printPathAndDistance(int parent[], int target) {
    if (parent[target] == -1) {
        printf("%d ", target);
        return;    }
    printPathAndDistance(parent, parent[target]);
    printf("-> %d ", target);
}
void printSolution(int dist[], int parent[], int src, int vertices) {
    printf("Vertex   Shortest Distance   Shortest Path\n");
    for (int v = 0; v < vertices; v++) {
        printf("%d \t\t %d \t\t\t ", v, dist[v]);
        printPathAndDistance(parent, v);
        printf("\n");
    }
}
void dijkstra(int graph[MAX_VERTICES][MAX_VERTICES], int src, int vertices) {
    int dist[MAX_VERTICES];
    bool sptSet[MAX_VERTICES];
    int parent[MAX_VERTICES];
    for (int i = 0; i < vertices; i++) {
        dist[i] = INT_MAX;
        sptSet[i] = false;
        parent[i] = -1;
    }
    dist[src] = 0;
    for (int count = 0; count < vertices - 1; count++) {
        int u = minDistance(dist, sptSet, vertices);
        sptSet[u] = true;
        for (int v = 0; v < vertices; v++) {
            if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX && (dist[u] + graph[u][v] < dist[v])) {
```

```
            dist[v] = dist[u] + graph[u][v];
            parent[v] = u;
        }      }   }
    printSolution(dist, parent, src, vertices);
}int main() {
    int vertices;
    printf("Boddu Asmitha Bhavya_A2305221386");
    printf("\nEnter the number of vertices: ");
    scanf("%d", &vertices);
    int graph[MAX_VERTICES][MAX_VERTICES];
    printf("\nEnter the adjacency matrix:\n");
    for (int i = 0; i < vertices; i++) {
        for (int j = 0; j < vertices; j++) {
            scanf("%d", &graph[i][j]);
        }  }
    int source;
    printf("\nEnter the source vertex: ");
    scanf("%d", &source);
    dijkstra(graph, source, vertices);
    return 0;
}
```

```
Boddu Asmitha Bhavya_A2305221386
Enter the number of vertices: 3

Enter the adjacency matrix:
2 0 3
0 6 4
1 8 0

Enter the source vertex: 1
Vertex   Shortest Distance   Shortest Path
0                5                       1 -> 2 -> 0
1                0                       1
2                4                       1 -> 2
```

Time complexity: O(E log V)