DATA STRUCTURES

Division	**CS(AIML) -A**
Batch	**2**
GR-no	**12311493**
Roll no	**54**
Name	**Atharva Kangralkar**

Assignment 14:

WAP to generate Single Source Shortest Path using Dijkstras Algorithm when graph is represented by A. Adjacency Matrix. B. Adjacency Lists.

Code:-

1. Adjacency Matrix

```
#include <stdio.h>
#include <limits.h>

#define MAX 100

void dijkstra_matrix(int graph[MAX][MAX], int n, int src) {
  int dist[MAX];
  int visited[MAX] = {0};

  for (int i = 0; i < n; i++)
    dist[i] = INT_MAX;

  dist[src] = 0;

  for (int count = 0; count < n - 1; count++) {
    int min = INT_MAX, u = -1;

    for (int v = 0; v < n; v++)
      if (!visited[v] && dist[v] <= min) {
        min = dist[v];
        u = v;
      }

    if (u == -1) break;  // No reachable vertex remaining

    visited[u] = 1;

    for (int v = 0; v < n; v++)
      if (!visited[v] && graph[u][v] && dist[u] != INT_MAX &&
        dist[u] + graph[u][v] < dist[v])
        dist[v] = dist[u] + graph[u][v];
  }

  printf("\nShortest distances from node %d:\n", src);
  for (int i = 0; i < n; i++)
    if (dist[i] == INT_MAX)
      printf("To %d = Unreachable\n", i);
```

```
            else
                printf("To %d = %d\n", i, dist[i]);
        }

        int main() {
            int n, graph[MAX][MAX], src;

            printf("Enter number of vertices: ");
            scanf("%d", &n);

            printf("Enter adjacency matrix (0 for no edge):\n");
            for (int i = 0; i < n; i++)
                for (int j = 0; j < n; j++)
                    scanf("%d", &graph[i][j]);

            printf("Enter source vertex: ");
            scanf("%d", &src);

            dijkstra_matrix(graph, n, src);

            return 0;
        }
```

2. Adjacency List

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

#define MAX 100

typedef struct Node {
    int vertex, weight;
    struct Node* next;
} Node;

Node* adjList[MAX];
```

```
void addEdge(int u, int v, int w) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->vertex = v;
   newNode->weight = w;
   newNode->next = adjList[u];
   adjList[u] = newNode;
}

void dijkstra_list(int n, int src) {
   int dist[MAX], visited[MAX] = {0};

   for (int i = 0; i < n; i++)
      dist[i] = INT_MAX;
   dist[src] = 0;

   for (int count = 0; count < n - 1; count++) {
      int min = INT_MAX, u = -1;

      for (int i = 0; i < n; i++)
         if (!visited[i] && dist[i] <= min) {
            min = dist[i];
            u = i;
         }

      if (u == -1) break;  // No reachable vertex left

      visited[u] = 1;
```

```
        Node* temp = adjList[u];
        while (temp != NULL) {
            int v = temp->vertex;
            if (!visited[v] && dist[u] != INT_MAX &&
                dist[u] + temp->weight < dist[v])
                dist[v] = dist[u] + temp->weight;
            temp = temp->next;
        }
    }

    printf("\nShortest distances from node %d:\n", src);
    for (int i = 0; i < n; i++)
        if (dist[i] == INT_MAX)
            printf("To %d = Unreachable\n", i);
        else
            printf("To %d = %d\n", i, dist[i]);
}

int main() {
    int n, e, u, v, w, src;

    printf("Enter number of vertices: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++)
        adjList[i] = NULL;
```

```
    printf("Enter number of edges: ");
    scanf("%d", &e);

    printf("Enter edges (u v weight):\n");
    for (int i = 0; i < e; i++) {
        scanf("%d %d %d", &u, &v, &w);
        addEdge(u, v, w); // directed
        // For undirected: addEdge(v, u, w);
    }

    printf("Enter source vertex: ");
    scanf("%d", &src);

    dijkstra_list(n, src);

    return 0;
}
```

Output:-

1. **Adjacency matrix:-**

```
Enter number of vertices: 4
Enter adjacency matrix (0 for no edge):
0 1 4 0
0 0 2 0
0 0 0 3
0 0 0 0
Enter source vertex: 0

Shortest distances from node 0:
To 0 = 0
To 1 = 1
To 2 = 3
To 3 = 6
```

2. **Adjacency list:-**

```
Enter number of vertices: 5
Enter number of edges: 3
Enter edges (u v weight):
0 1 2
1 2 3
2 3 1
Enter source vertex: 0

Shortest distances from node 0:
To 0 = 0
To 1 = 2
To 2 = 5
To 3 = 6
To 4 = Unreachable
```