# ADS LAB ASSINGMENT 8

**Name-Sandesh Santosh Kadam**

**Class- SY IT-C Batch 1**

**PRN-12210695**

**Roll no-26**

Q. **Write C/C++ program for MST  Prim’s algorithm**

# Theory-

**Prim's Algorithm for Minimum Spanning Tree (MST)**

Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a connected, undirected, and weighted graph. The minimum spanning tree is a subset of the edges that connects all vertices in the graph with the minimum possible total edge weight.

**Overview:**

1. **Initialization:**
   * Start with an arbitrary vertex as the initial tree.
   * Initialize a priority queue (or a min-heap) to store edges based on their weights.
2. **Process:**
   * At each step, add the minimum weight edge that connects a vertex in the existing tree to a vertex outside the tree.
   * This ensures that the tree always remains connected and forms a spanning tree with minimum total weight.
3. **Algorithm Steps:**
   * Initialize an empty priority queue (or min-heap) and add all edges of the starting vertex to the priority queue.
   * While the priority queue is not empty:
     + Extract the edge with the minimum weight from the priority queue.
     + If the edge connects two different components of the growing minimum spanning tree, add it to the tree.
     + Add all edges of the newly added vertex to the priority queue.
4. **Termination:**
   * The algorithm terminates when all vertices are included in the minimum spanning tree.

**Programming Lang. Used- C**

**Compiler Used- VS Code**

# CODE-

#include <stdio.h>

#include <limits.h>

// Number of vertices in the graph

#define V 5

// Function to find the vertex with the minimum key value

int minKey(int key[], int mstSet[]) {

    int min = INT\_MAX, min\_index;

    for (int v = 0; v < V; v++) {

        if (mstSet[v] == 0 && key[v] < min) {

            min = key[v];

            min\_index = v;

        }

    }

    return min\_index;

}

// Function to print the constructed MST stored in parent[]

void printMST(int parent[], int graph[V][V]) {

    printf("Edge   Weight\n");

    for (int i = 1; i < V; i++) {

        printf("%d - %d    %d \n", parent[i], i, graph[i][parent[i]]);

    }

}

// Function to implement Prim's MST algorithm

void primMST(int graph[V][V]) {

    int parent[V];   // Array to store constructed MST

    int key[V];      // Key values used to pick minimum weight edge

    int mstSet[V];   // To represent set of vertices included in MST

    // Initialize all keys as INFINITE

    // and mstSet[] as 0

    for (int i = 0; i < V; i++) {

        key[i] = INT\_MAX;

        mstSet[i] = 0;

    }

    // Always include the first vertex in MST.

    key[0] = 0;       // Make key 0 so that this vertex is picked as the first vertex

    parent[0] = -1;   // First node is always the root of the MST

    // The MST will have V vertices

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

        // Pick the minimum key vertex from the set of vertices not yet included in MST

        int u = minKey(key, mstSet);

        // Add the picked vertex to the MST set

        mstSet[u] = 1;

        // Update key value and parent index of the adjacent vertices of the picked vertex

        for (int v = 0; v < V; v++) {

            // Update key[v] only if the graph[u][v] is smaller than key[v]

            if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v]) {

                parent[v] = u;

                key[v] = graph[u][v];

            }

        }

    }

    // Print the constructed MST

    printMST(parent, graph);

}

// Driver program to test above functions

int main() {

    // Example graph represented by an adjacency matrix

    int graph[V][V] = {{0, 2, 0, 6, 0},

                       {2, 0, 3, 8, 5},

                       {0, 3, 0, 0, 7},

                       {6, 8, 0, 0, 9},

                       {0, 5, 7, 9, 0}};

    // Print the MST using Prim's algorithm

    printf("Minimum Spanning Tree using Prim's algorithm:\n");

    primMST(graph);

    return 0;

}

# OUTPUT-

