S. B. JAIN INSTITUTE OF TECHNOLOGY,

MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 3

Aim: Create a program that finds the Minimum Cost Spanning Tree of a given undirected graph using Prim’s algorithm.

Name of Student: Shrutika Pradeep Bagdi

Roll No: CS22130

Semester/Year: V/III

Academic Session:2024-2025

Date of Performance:

Date of Submission:

AIM: Create a program that finds the Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

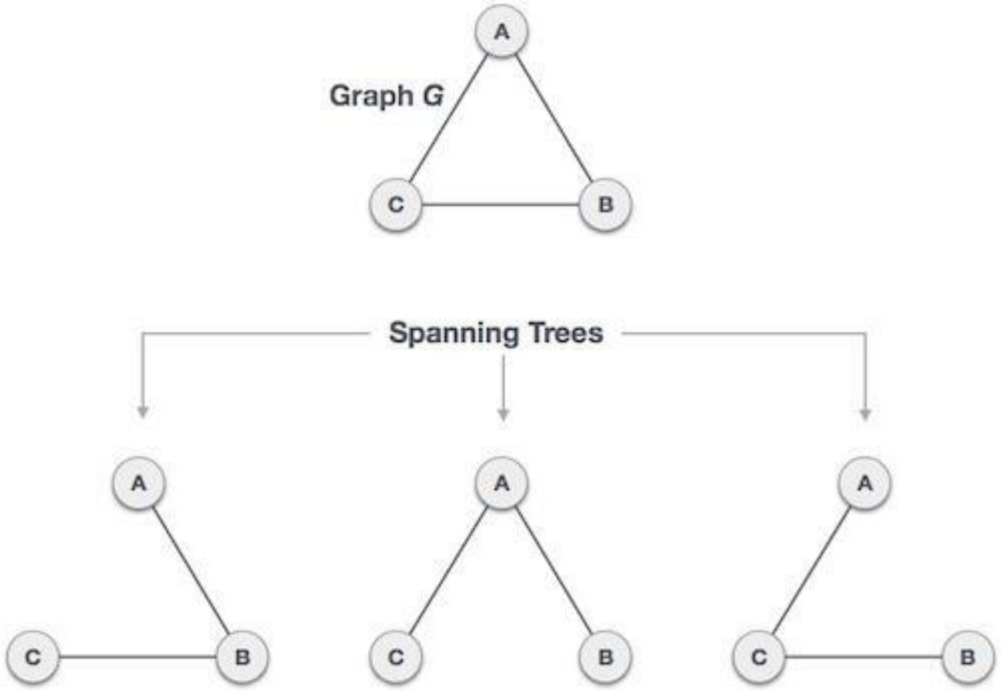
The objectives and expected learning outcome of this practical are:

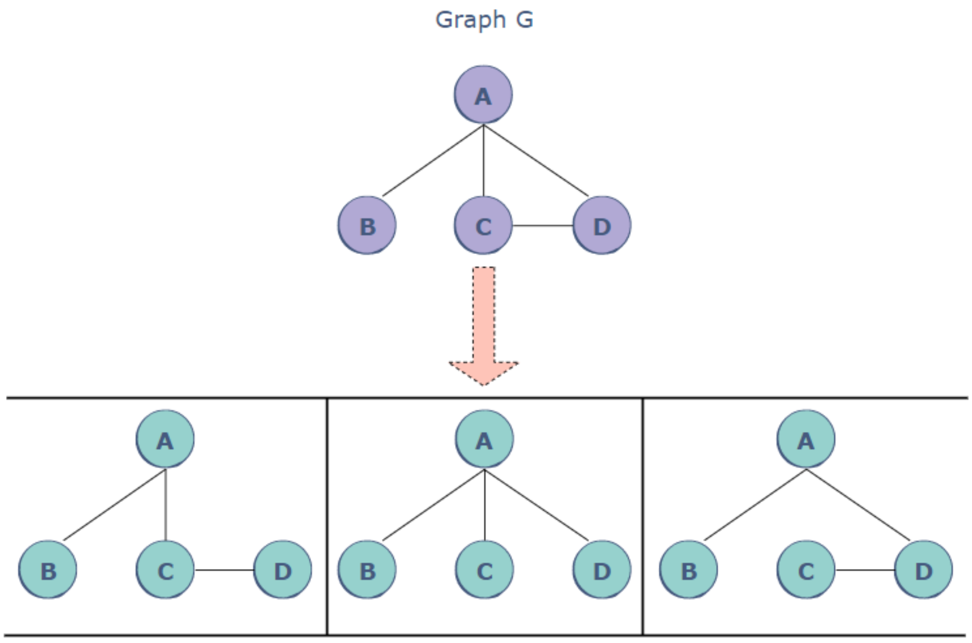
- To understand the concepts of spanning trees of the graph.
- To understand and implement the minimum cost spanning trees using greedy method.

THEORY:

A spanning tree is a subset of Graph G, which has all the vertices covered with minimum possible number of edges. Hence, a spanning tree does not have cycles and it cannot be disconnected.

By this definition, we can draw a conclusion that every connected and undirected Graph G has at least one spanning tree. A disconnected graph does not have any spanning tree, as it cannot be spanned to all its vertices.





Different possible spanning trees

General Properties of Spanning Tree:

- A connected graph G can have more than one spanning tree.
- All possible spanning trees of graph G, have the same number of edges and vertices.
- The spanning tree does not have any cycle (loops).
- Removing one edge from the spanning tree will make the graph disconnected, i.e. the spanning tree is **minimally connected**.
- Adding one edge to the spanning tree will create a circuit or loop, i.e. the spanning tree is **maximally acyclic**.

Mathematical Properties of Spanning Tree:

- Spanning tree has **n-1** edges, where **n** is the number of nodes (vertices).
- A complete graph can have maximum $\mathbf{n^{n-2}}$ number of spanning trees.

Therefore, spanning trees are a subset of connected Graph G and disconnected graphs do not have spanning tree.

Application of Spanning Tree:

Spanning tree is basically used to find a minimum path to connect all nodes in a graph. Common application of spanning trees are,

- Civil Network Planning
- Computer Network Routing Protocol
- Cluster Analysis

Minimum Cost Spanning Tree (MCST):

In a weighted graph, a minimum spanning tree is a spanning tree that has minimum weight than all other spanning trees of the same graph.

Minimum Spanning-Tree Algorithm:

- Kruskal's Algorithm
- Prim's Algorithm

How does Prim's Algorithm Work?

The working of Prim's algorithm can be described by using the following steps:

Step 1: *Determine an arbitrary vertex as the starting vertex of the MST.*

Step 2: *Follow steps 3 to 5 till there are vertices that are not included in the MST (known as fringe vertex).*

Step 3: *Find edges connecting any tree vertex with the fringe vertices.*

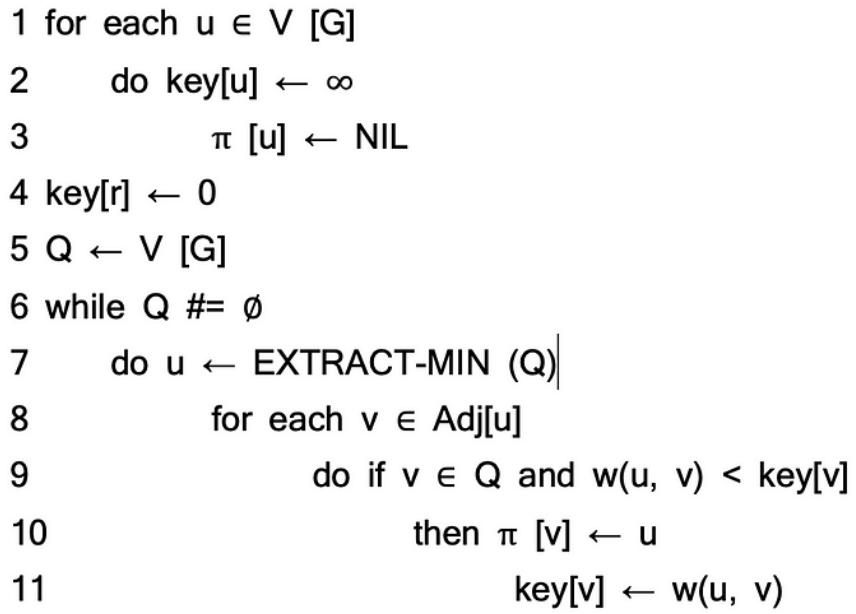
Step 4: *Find the minimum among these edges.*

Step 5: *Add the chosen edge to the MST if it does not form any cycle.*

Step 6: *Return the MST and exit*

ALGORITHM:

MST-PRIM (G, w, r)

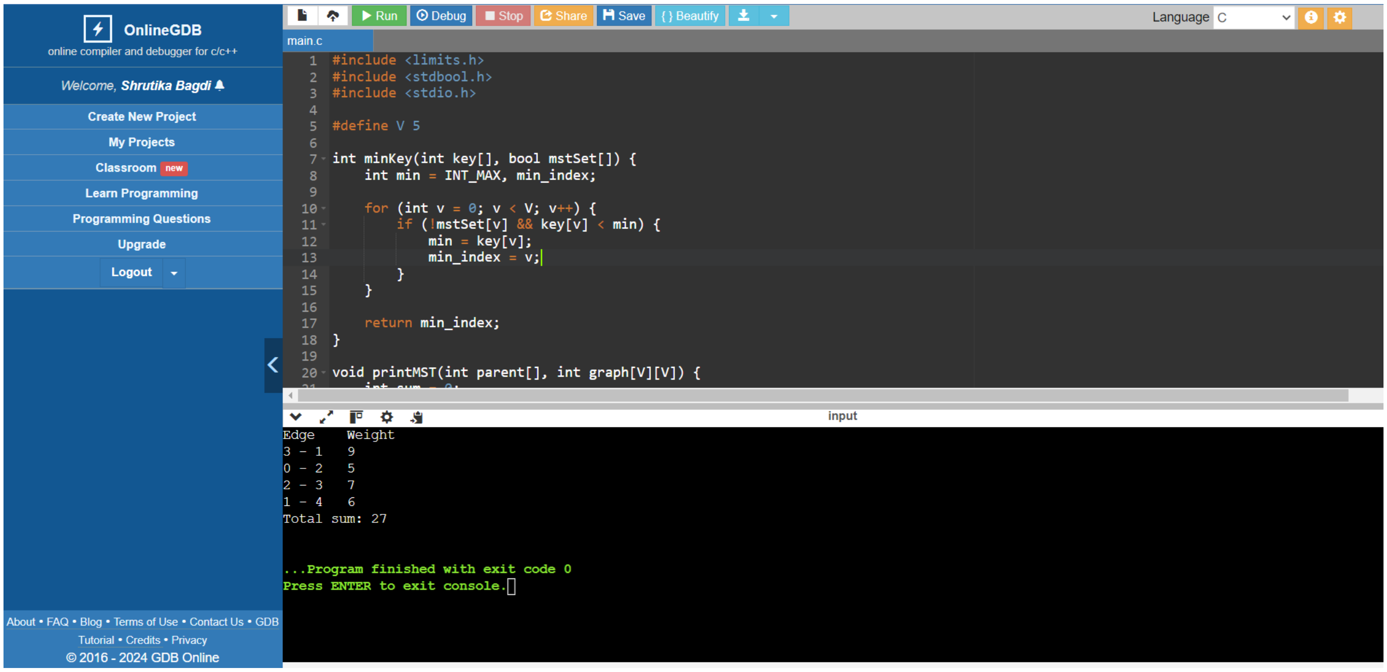


```
for each u ∈ V [G]
    do key[u] ← ∞
        π [u] ← NIL
key[r] ← 0
Q ← V [G]
while Q #= ∅
    do u ← EXTRACT-MIN (Q)
        for each v ∈ Adj[u]
            do if v ∈ Q and w(u, v) < key[v]
                then π [v] ← u
                    key[v] ← w(u, v)
```

CODE:

```
#include <limits.h>
#include <stdbool.h>
#include <stdio.h>
#define V 5

int minKey(int key[], bool mstSet[]) {
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++) {
        if (!mstSet[v] && key[v] < min) {
            min = key[v];
            min_index = v;
        }
    }
    return min_index;
}
void printMST(int parent[], int graph[V][V]) {
    int sum = 0;
    printf("Edge \tWeight\n");
    for (int i = 1; i < V; i++) {
        printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);
        sum += graph[i][parent[i]];
    }
    printf("Total sum: %d\n", sum);
}

void primMST(int graph[V][V]) {
    int parent[V];
    int key[V];
    bool mstSet[V];

    for (int i = 0; i < V; i++) {
        key[i] = INT_MAX;
        mstSet[i] = false;
    }

    key[0] = 0;
    parent[0] = -1;

    for (int count = 0; count < V - 1; count++) {
        int u = minKey(key, mstSet);
        mstSet[u] = true;

        for (int v = 0; v < V; v++) {
            if (graph[u][v] && !mstSet[v] && graph[u][v] < key[v]) {
                parent[v] = u;
                key[v] = graph[u][v];
            }
        }
    }
    printMST(parent, graph);
}

int main() {
    int graph[V][V] = { { 0, 10, 5, 0, 0 },
                        { 10, 0, 0, 9, 6 },
                        { 5, 0, 0, 7, 11 },
                        { 0, 9, 7, 0, 21 },
                        { 0, 6, 11, 21, 0 } };

    primMST(graph);

    return 0;
}
```

INPUT & OUTPUT WITH DIFFERENT TEST CASES:



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

#define V 5

int minKey(int key[], bool mstSet[]) {
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++) {
        if (!mstSet[v] && key[v] < min) {
            min = key[v];
            min_index = v;
        }
    }

    return min_index;
}

void printMST(int parent[], int graph[V][V]) {
    int sum = 0;
```

```
Edge    Weight
3 - 1    9
0 - 2    5
2 - 3    7
1 - 4    6
Total sum: 27


...Program finished with exit code 0
Press ENTER to exit console.
```

CONCLUSION:

Thus, create a program that finds the Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.

DISCUSSION AND VIVA VOCE:

- What is a minimum cost spanning tree?
- Discuss Prim's algorithm.
- Discuss the Kruskal's algorithm.
- Explain the properties of minimum spanning tree.

REFERENCES:

- *https://www.javatpoint.com/prim-algorithm*
- *https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/*
- *https://www.simplilearn.com/tutorials/data-structure-tutorial/prims-algorithm*
- *https://www.tutorialspoint.com/data_structures_algorithms/prims_spanning_tree_algorithm.htm*