Practical – 10

AIM: Implementat prim's algorithm.

```
// for adjacency matrix representation of the graph

#include <bits/stdc++.h>
using namespace std;

// Number of vertices in the graph
#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] == false && key[v] < min)
            min = key[v], min_index = v;

    return min_index;
}

void printMST(int parent[], int graph[V][V])
{
    cout << "Edge \tWeight\n";
    for (int i = 1; i < V; i++)
        cout << parent[i] << " - " << i << " \t"
             << graph[i][parent[i]] << " \n";
}

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] == false && graph[u][v] < key[v])
                parent[v] = u, key[v] = graph[u][v];
    }
```

```
    printMST(parent, graph);
}

int main()
{
    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 solution
    primMST(graph);

    return 0;
}
```

OUTPUT

```
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5
```

Time analysis

Operation	Time Complexity (Typical Cases)
Initialization	O(V + E)
Main Loop (V-1 iterations)	O(V + E) per iteration
- Finding minimum-weight edge	O(V) with a min-heap, O(E) with a simple priority queue
- Updating key values	O(log V) with a min-heap, O(1) with an array
- Relaxing adjacent edges	O(E)
Total Time Complexity	O((V + E) * log V) typical, O(V^2) worst-case with simple priority queue

Applications

1. **Network Design:**
   - Telecommunications: Designing cost-effective networks for connecting cities, buildings, or devices, such as laying fiber optic cables or establishing wireless links.
   - Transportation: Planning routes for roads, railways, or airline networks to minimize construction or travel costs while connecting all destinations.
   - Electrical Grids: Designing power distribution networks to minimize the cost of wiring and ensure efficient energy delivery.

2. **Clustering:**
   - Image Segmentation: Dividing images into regions with similar characteristics (e.g., color, texture) for object recognition or analysis.
   - Data Analysis: Grouping similar data points in various domains, such as customer segmentation for targeted marketing or identifying patterns in scientific data.

3. **Maze Generation:**
   - Game Design: Creating random mazes with guaranteed solvability for puzzle games or challenges.

4. **Approximation Algorithms**:
   - Traveling Salesman Problem: Finding near-optimal solutions for the TSP by using a minimum spanning tree as a starting point.

5. Other Applications:
   - Bioinformatics: Analyzing protein structures and genetic sequences.
   - VLSI Design: Designing integrated circuits.
   - Social Network Analysis: Identifying communities and influential nodes in social networks.