DAA Assignment 10

Devam Desai IIT2022035

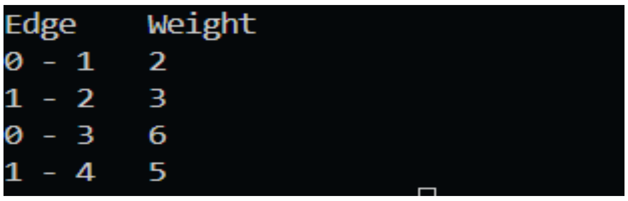
Q1) A university campus is planning to lay down a new fiber-optic network to

connect all its buildings. Use Prim's algorithm to determine the most cost-effective connection plan.

Source Code:

```
#include <cstring>
#include <iostream>
using namespace std;

#define INF 9999999

#define V 5
int G[V][V] = {
  {0, 9, 75, 0, 0},
  {9, 0, 95, 19, 42},
  {75, 95, 0, 51, 66},
  {0, 19, 51, 0, 31},
  {0, 42, 66, 31, 0}};

int main() {
  int no_edge;  // number of edge
  int selected[V];

  memset(selected, false, sizeof(selected));

  no_edge = 0;

  selected[0] = true;

  int x;  //  row number
  int y;  //  col number

  cout << "Edge"
     << " : "
     << "Weight";
  cout << endl;
  while (no_edge < V - 1) {

    int min = INF;
    x = 0;
    y = 0;

    for (int i = 0; i < V; i++) {
      if (selected[i]) {
        for (int j = 0; j < V; j++) {
          if (!selected[j] && G[i][j]) {
           if (min > G[i][j]) {
              min = G[i][j];
              x = i;
              y = j;
            }
          }
        }
      }
    }
    cout << x << " - " << y << " :  " << G[x][y];
    cout << endl;
    selected[y] = true;
    no_edge++;
  }

  return 0;
}
```

OUTPUT:



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

REPORT:

Introduction:

The task at hand involves planning a fiber-optic network to connect all buildings within a university campus. This report outlines the utilization of Prim's Algorithm for finding the Minimum Spanning Tree (MST), calculates the total cost for laying down the fiber-optic cables, and suggests a method to integrate a new building into the network with minimal additional cost.

1. Prim's Algorithm for MST:

Prim's Algorithm is a greedy algorithm used to find the Minimum Spanning Tree of a weighted undirected graph. The algorithm works by starting with an arbitrary vertex and repeatedly adding the shortest edge that connects a vertex in the MST to a vertex outside the MST until all vertices are included.

2. How MST is Made:

- Initialization: Start with an arbitrary vertex (Building) and mark it as visited.
- Iterative Steps:
- Add the shortest edge that connects a vertex in the MST to a vertex outside the MST.
- Continue adding edges until all vertices are included in the MST.
- Termination: Stop when all vertices are visited.

3. Total Cost Calculation:

- The total cost for laying down the fiber-optic cables is calculated by summing the weights of all edges in the Minimum Spanning Tree. This includes the cost of connecting each building in the network.

4. Integration of New Building:

When a new building is added to the network, the process involves:

- Determining Nearest Building: Identify the nearest existing building to the new one.
- Connecting the New Building: Establish a connection (edge) between the new building and the nearest existing building.
- Minimizing Additional Cost: This approach minimizes the additional cost by using the existing infrastructure efficiently.

Conclusion:

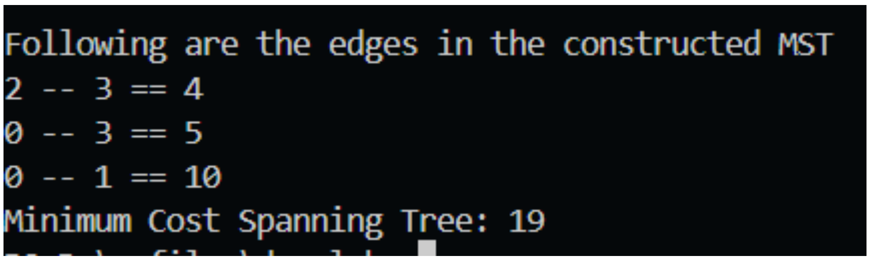
Prim's Algorithm offers an effective method for designing a cost-effective fiber-optic network within the university campus. By constructing the Minimum Spanning Tree and calculating the total cost, the campus can efficiently plan its network infrastructure. Integrating new buildings into the network can be seamlessly achieved by connecting them to the nearest existing buildings, thereby minimizing additional costs while ensuring optimal network connectivity.

Q2) Statement: A small town is developing its road network. Use Kruskal's algorithm to plan the network for connecting all junctions at a minimal cost.

Source Code:

```
vector<int> parent, rank;


void make_set(int v) {
    parent[v] = v;
    rank[v] = 0;
}

int find_set(int v) {
    if (v == parent[v])
        return v;
    return parent[v] = find_set(parent[v]);
}

void union_sets(int a, int b) {
    a = find_set(a);
    b = find_set(b);
    if (a != b) {
        if (rank[a] < rank[b])
            swap(a, b);
        parent[b] = a;
        if (rank[a] == rank[b])
            rank[a]++;
    }
}

struct Edge {
    int u, v, weight;
    bool operator<(Edge const& other) {
        return weight < other.weight;
    }
};

int n;
vector<Edge> edges;

int cost = 0;
vector<Edge> result;
parent.resize(n);
rank.resize(n);
for (int i = 0; i < n; i++)
    make_set(i);

sort(edges.begin(), edges.end());

for (Edge e : edges) {
    if (find_set(e.u) != find_set(e.v)) {
        cost += e.weight;
        result.push_back(e);
        union_sets(e.u, e.v);
    }
}
```

OUTPUT:



```
Following are the edges in the constructed MST
2 -- 3 == 4
0 -- 3 == 5
0 -- 1 == 10
Minimum Cost Spanning Tree: 19
```

REPORT

Introduction:

The task involves developing a road network in a small town to connect all junctions at a minimal cost. This report outlines the utilization of Kruskal's Algorithm for finding the Minimum Spanning Tree (MST) to plan the network infrastructure efficiently.

1. Kruskal's Algorithm for MST:

Kruskal's Algorithm is a greedy algorithm used to find the Minimum Spanning Tree of a weighted undirected graph. It works by sorting the edges of the graph by weight and then adding edges in ascending order while avoiding cycles until all vertices are included in the MST.

2. How MST is Made:

- Initialization: Start with all vertices as separate disjoint sets.
- Iterative Steps:
- Sort the edges of the graph by weight.
- Select the smallest edge that does not form a cycle in the MST.
- Merge the sets of vertices connected by the selected edge.
- Repeat until all vertices are included in the MST.
- Termination: Stop when all vertices are visited.

3. Total Cost Calculation:

The total cost for developing the road network is calculated by summing the weights of all edges in the Minimum Spanning Tree. This includes the cost of constructing roads to connect each junction in the network.

4. Integration of New Junctions:

When a new junction is added to the network, it can be seamlessly integrated by:

- Determining Nearest Junction: Identify the nearest existing junction to the new one.
- Connecting the New Junction: Establish a connection (edge) between the new junction and the nearest existing junction.
- Minimizing Additional Cost: This approach minimizes the additional cost by utilizing the existing road infrastructure efficiently.

Conclusion:

Kruskal's Algorithm offers an effective method for designing a cost-effective road network in the small town. By constructing the Minimum Spanning Tree and calculating the total cost, the town can efficiently plan its road infrastructure. Integrating new junctions into the network can be seamlessly achieved by connecting them to the nearest existing junctions, thereby minimizing additional costs while ensuring optimal connectivity throughout the town.