DATA STRUCTURES AND ALGORITHMS LABORATORY

Group C

Assignment No. 2

Name :- Ojus Pravin Jaiswal
Roll No. :- SACO19108
Division :- A

Group C

Assignment 2

Title: You have a business with several offices; you want to lease phone lines to connect them up with each other; and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve the problem by suggesting appropriate data structures.

Objectives:

1. To understand concept of Spanning Tree and Greedy Algorithm.
2. To identify the minimum distance between the vertices of Graph in Data structure.

Outcome:

1. Identify the minimum distance between the vertices of Graph using Prims Algorithm

Theory:

Properties of a Greedy Algorithm:

1. At each step, the best possible choice is taken and after that only the sub-problem is solved.

2. Greedy algorithm might be depending on many choices. But, it cannot ever be depending upon any choices of future and neither on sub-problems solutions.

3. The method of greedy algorithm starts with a top and goes down, creating greedy choices in a series and then reduce each of the given problem to even smaller ones.

Minimum Spanning Tree:

A Minimum Spanning Tree (MST) is a kind of a sub graph of an undirected graph in which, the sub graph spans or includes all the nodes has a minimum total edge weight.

To solve the problem by a prim's algorithm, all we need is to find a spanning tree of minimum length, where a spanning tree is a tree that connects all the vertices together and a minimum spanning tree is a spanning tree of minimum length.

Properties of Prim's Algorithm:

Prim's Algorithm has the following properties:

1. The edges in the subset of some minimum spanning tree always form a single tree.

2. It grows the tree until it spans all the vertices of the graph.

3. An edge is added to the tree, at every step, that crosses a cut if its weight is the minimum of any edge crossing the cut, connecting it to a vertex of the graph.

Algorithm:

1. Begin with any vertex which you think would be suitable and add it to the tree.

2. Find an edge that connects any vertex in the tree to any vertex that is not in the tree. Note that, we don't have to form cycles.

3. Stop when n - 1 edges have been added to the tree

Software Required: g++ / gcc compiler- / 64 bit Fedora, eclipse IDE

Input: No of branches, No of Connections, Cost

Program:

```
//============================================================================
// Name        : GraphRepresentation.cpp
// Author      :
// Version     :
// Copyright   : Your copyright notice
// Description : Prims Algorithm
//============================================================================

#include <iostream>
#include <iomanip>
using namespace std;

class tree
{
        int a[20][20], l, u, w, i, j, v, e, visited[20];

  public:
        void input();
        void display();
```

```
        void minimum();
};

void tree::input()
{
        cout << "\nEnter the no. of branches : ";
        cin >> v;

        for (i = 0; i < v; i++)
        {
                visited[i] = 0;
                for (j = 0; j < v; j++)
                {
                        a[i][j] = 999;
                }
        }

        cout << "\nEnter the no. of connections : ";
        cin >> e;

        for (i = 0; i < e; i++)
        {
                cout << "\nEnter the end branches of connections : ";
                cin >> l >> u;
                cout << "Enter the phone company charges for this connection : ";
                cin >> w;
                a[l - 1][u - 1] = a[u - 1][l - 1] = w;
        }
```

```
}

void tree::display()
{
	cout << "\nAdjacency matrix : \n";
	for (i = 0; i < v; i++)
	{
		cout << endl;
		for (j = 0; j < v; j++)
		{
			cout << setw(3) << a[i][j] << "   ";
		}
		cout << endl;
	}
}

void tree::minimum()
{
	int p = 0, q = 0, total = 0, min;
	visited[0] = 1;
	for (int count = 0; count < (v - 1); count++)
	{
		min = 999;
		for (i = 0; i < v; i++)
		{
			if (visited[i] == 1)
			{
				for (j = 0; j < v; j++)
```

```
                {
                    if (visited[j] != 1)
                    {
                        if (min > a[i][j])
                        {
                            min = a[i][j];
                            p = i;
                            q = j;
                        }
                    }
                }
            }
        }
        visited[p] = 1;
        visited[q] = 1;
        total = total + min;
        cout << "\nMinimum cost connection is " << (p + 1) << " -> " << (q + 1) << "  with charge : "
<< min;
    }
    cout << "\n\nThe minimum total cost of connections of all branches is : " << total << endl;
}

int main()
{
    int ch;
    tree t;
    cout << "\n==========PRIM'S ALGORITHM==========" << endl;
    while (1)
```

```
	{
		cout << "\n-----Menu-----\n1. Input\n2. Display\n3. Minimum\n4. Exit Program\n";
		cout << "\nEnter your choice : ";
		cin >> ch;
		switch (ch)
		{
		case 1:
			cout << "\n***Input Your Values***" << endl;
			t.input();
			break;
		case 2:
			cout << "\n***Display The Contents***" << endl;
			t.display();
			break;
		case 3:
			cout << "\n***Minimum***" << endl;
			t.minimum();
			break;
		case 4:
			cout << "\nExitting Program!!!\n";
			exit(0);
		default:
			cout << "\nWrong choice entered!!!\n";
		}
	}
	return 0;
}
```

Output:

```
==========PRIM'S ALGORITHM==========

-----Menu-----
1. Input
2. Display
3. Minimum
4. Exit Program

Enter your choice : 1

***Input Your Values***

Enter the no. of branches : 6

Enter the no. of connections : 8

Enter the end branches of connections : 1 2
Enter the phone company charges for this connection : 4

Enter the end branches of connections : 2 3
Enter the phone company charges for this connection : 2

Enter the end branches of connections : 1 3
Enter the phone company charges for this connection : 4

Enter the end branches of connections : 3 4
Enter the phone company charges for this connection : 3

Enter the end branches of connections : 3 5
Enter the phone company charges for this connection : 2

Enter the end branches of connections : 3 6
Enter the phone company charges for this connection : 4

Enter the end branches of connections : 4 6
Enter the phone company charges for this connection : 3

Enter the end branches of connections : 5 6
Enter the phone company charges for this connection : 3

-----Menu-----
1. Input
2. Display
3. Minimum
4. Exit Program

Enter your choice : 2
```

```
***Display The Contents***

Adjacency matrix :

999     4     4   999   999   999

  4   999     2   999   999   999

  4     2   999     3     2     4

999   999     3   999   999     3

999   999     2   999   999     3

999   999     4     3     3   999

-----Menu-----
1. Input
2. Display
3. Minimum
4. Exit Program

Enter your choice : 3

***Minimum***

Minimum cost connection is 1 -> 2  with charge : 4
Minimum cost connection is 2 -> 3  with charge : 2
Minimum cost connection is 3 -> 5  with charge : 2
Minimum cost connection is 3 -> 4  with charge : 3
Minimum cost connection is 4 -> 6  with charge : 3

The minimum total cost of connections of all branches is : 14

-----Menu-----
1. Input
2. Display
3. Minimum
4. Exit Program

Enter your choice : 5

Wrong choice entered!!!

-----Menu-----
1. Input
2. Display
3. Minimum
4. Exit Program

Enter your choice : 4

Exitting Program!!!

[Program finished]
```

Conclusion: This program implements graph data structure