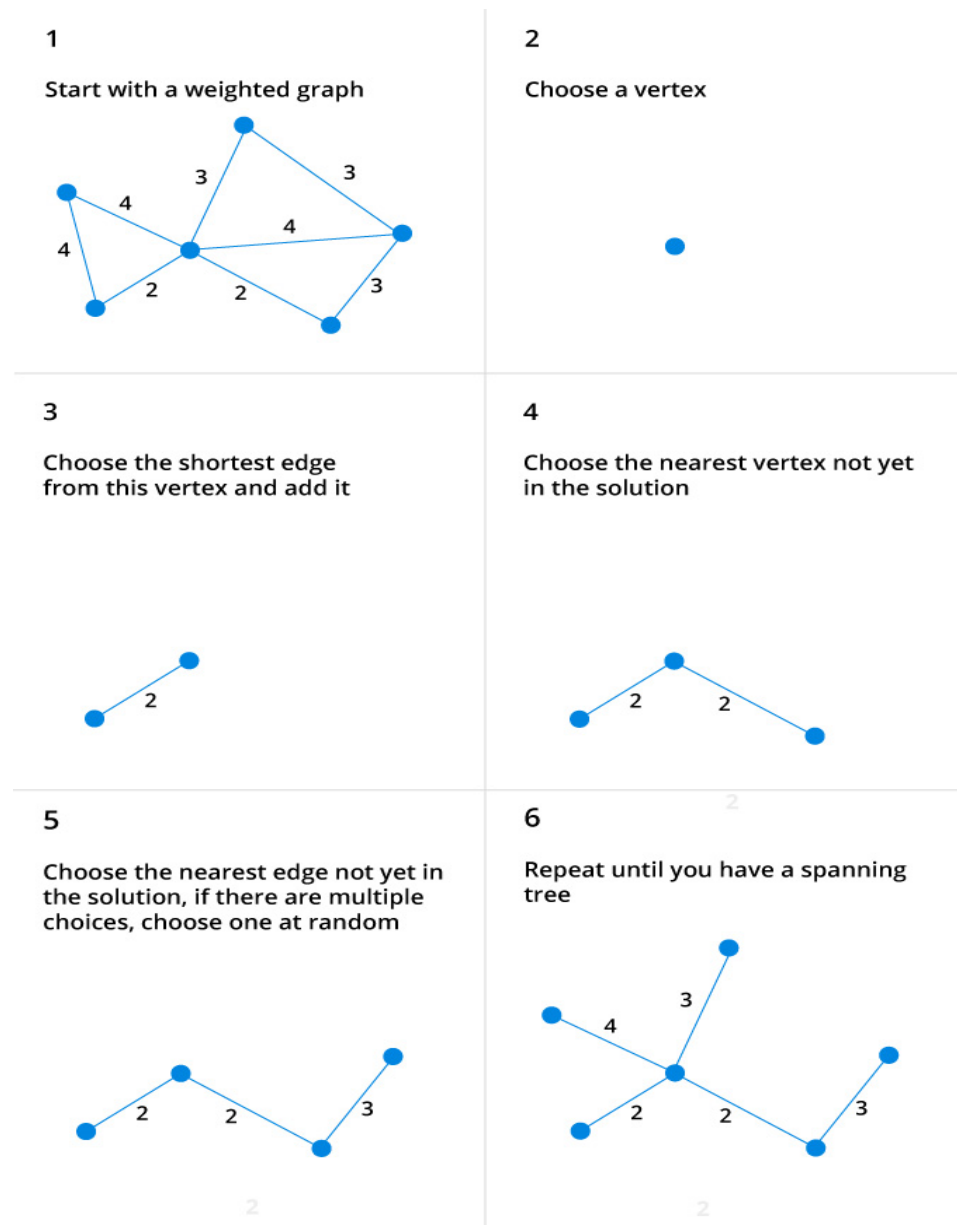# SKILL DEVELOPMENT LAB-II(2018-19)

ASSIGNMENT 4

- **Aim:** For a weighted graph G, find the minimum spanning tree using Prims Algorithm.

- **Objective:** To find the minimum cost by weight using Prims algorithm.

- **Theory:** **Prim's algorithm** is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. The algorithm operates by building this tree one vertex at a time, from an arbitrary starting vertex, at each step adding the cheapest possible connection from the tree to another vertex. Prim's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which form a tree that includes every vertex and has the minimum sum of weights among all the trees that can be formed from the graph.

Example of Prim's algorithm

1

Start with a weighted graph

2

Choose a vertex

3

Choose the shortest edge
from this vertex and add it

4

Choose the nearest vertex not yet
in the solution

5

Choose the nearest edge not yet in
the solution, if there are multiple
choices, choose one at random

6

Repeat until you have a spanning
tree

• Algorithm:

1. Maintain two disjoint sets of vertices. One containing vertices that are in the growing spanning tree and other that are not in the growing spanning tree.
2. Select the cheapest vertex that is connected to the growing spanning tree and is not in the growing spanning tree and add it into the growing spanning tree. This can be done using Priority Queues. Insert the vertices, that are connected to growing spanning tree, into the Priority Queue.
3. Check for cycles. To do that, mark the nodes which have been already selected and insert only those nodes in the Priority Queue that are not marked.

• Program code:

#include<iostream>

```cpp
using namespace std;

class operation
{
    int ad[20][20],visited[20],i,j,a,b,c=0,w,k,l,s=0;
public:

    void inser()
    {
        cout<<"Enter the no of vertices & edges\n";
        cin>>a>>b;
        if(a>b)
        {
            cout<<"Error\n";
        }
        else
        {
        for(i=0;i<a;i++)
        {
            for(j=0;j<a;j++)
            {
                ad[i][j]=0;
            }
        }
        for(i=0;i<b;i++)
        {
            cout<<"Enter edge\t Weight\n";
            cin>>k>>l>>w;
            ad[k][l]=w;
        }
```

```
   prims();
  }
}
void prims()
{
   visited[0]=1;
   for(i=1;i<a;i++)
     visited[i]=0;
while (c<a-1)
{
  int min=9999,x=0,y=0;
  //for(i=0;i<a;i++)
  //{visited[i]=0;}
 for (int i = 0; i<a; i++)
  {
  if (visited[i]==1)
  {
     for (int j = 0; j <a; j++)
    {
       if (visited[j]==0 && ad[i][j])
      {
       if (min > ad[i][j])
        {
         min = ad[i][j];
         x = i;
         y = j;
        }
      }
    }
  }
```

```
      }

    }

     s=s+ad[x][y];

    cout << x <<  " - " << y << " :  " << ad[x][y];

    cout << endl;

    visited[y]=1;

    c++;

   }

   cout<<"The total weight of Minimum spanning tree is "<<s<<endl;

   }

};

int main()

{

   operation op;

   op.inser();

}
```

- Output:

```
6
8
Enter edge        Weight
0
1
1
Enter edge        Weight
0
3
6
Enter edge        Weight
0
5
8
Enter edge        Weight
1
2
2
Enter edge        Weight
1
5
5
Enter edge        Weight
2
3
3
Enter edge        Weight
3
4
4
Enter edge        Weight
4
5
7
0 - 1 :   1
1 - 2 :   2
2 - 3 :   3
3 - 4 :   4
1 - 5 :   5
The total weight of Minimum spanning tree is 15
```

- **Conclusion:** The time complexity of the Prim's Algorithm
  is O((V+E)logV) because each vertex is inserted in the priority queue only once and insertion in priority queue take logarithmic time.