Design & Analysis Of Algorithm

Lab Experiment -3

NAME : SURIYAPRAKASH.C

ROLL NO:CH.EN.U4CSE20170

SUBJECT: DESIGN & ANALYSIS OF ALGORITHM

SUBJECT CODE: 19CSE302

Submitted to – Mrs. Ashwini,

Department of CSE,

ASE Chennai campus.

EX NO: 3

Minimum Spanning Tree

1) **Prim's Algorithm**
2) **Kruskal Algorithm**

AIM:

To write an algorithm to implement prim's algorithm.

ALGORITHM:

1) Select a starting vertex
2) Repeat Steps 3 and 4 until there are fringe vertices
3) Select an edge 'e' connecting the tree vertex and fringe vertex that has minimum weight
4) Add the selected edge and the vertex to the minimum spanning treeT

[END OF LOOP]

5) EXIT

CODE SCREEN:

```
check = 9999999
N = 5
Graph = [[0, 2, 0, 6, 0],
         [0, 0, 3, 8, 5],
         [0, 3, 0, 0, 7],
         [6, 8, 0, 0, 9],
         [0, 5, 7, 9, 0]]
print("prims algorithm:")
selected_node = [0, 0, 0, 0, 0]
no_edge = 0
cost = 0
selected_node[0] = True
# printing for edge and weight
print("Edge : Weight\n")
while no_edge < N - 1:
    minimum = check
```

```
    a = 0
    b = 0
    for m in range(N):
        if selected_node[m]:
            for n in range(N):
                if (not selected_node[n]) and Graph[m][n]:
                    # not in selected and there is an edge
                    if minimum > Graph[m][n]:
                        minimum = Graph[m][n]
                        a = m
                        b = n
    print(str(a) + "-" + str(b) + ":" + " " + str(Graph[a][b]))
    selected_node[b] = True
    cost = cost + Graph[a][b]
    no_edge += 1
print("The cost is : ", cost)
```

OUTPUT SCREEN :

```
PS D:\python> & C:/Users/HP/AppData/Local/Programs/Python/Python310/python.exe d:/python/DAA/prims.py
prims algorithm:
Edge : Weight

0-1: 2
1-2: 3
1-4: 5
0-3: 6
The cost is :  16
PS D:\python>
```

TIME COMPLEXITY:

O(V*logV+E*logE), where V = No. of vertices, E = No. of edges Space

RESULT:

I have studied and understood the prim's algorithm in python language and executed the program successfully.

AIM:

To write an algorithm to implement Kruskal's algorithm .

ALGORITHM:

1) Sort all edges in increasing order of their edge weights.
2) Pick the smallest edge.
3) Check if the new edge creates a cycle or loop in a spanning tree.
4) If it doesn't form the cycle, then include that edge in MST. Otherwise, discard it.
5) Repeat from step 2 until it includes |V| - 1 edges in MST.

CODE SCREEN:

```
from collections import defaultdict

class Graph:

    def __init__(self, vertices):
        self.V = vertices
        self.graph = []
    def addEdge(self, u, v, w):
        self.graph.append([u, v, w])
    def find(self, parent, i):
        if parent[i] == i:
            return i
        return self.find(parent, parent[i])

    def union(self, parent, rank, x, y):
        xroot = self.find(parent, x)
        yroot = self.find(parent, y)
        if rank[xroot] < rank[yroot]:
            parent[xroot] = yroot
        elif rank[xroot] > rank[yroot]:
            parent[yroot] = xroot
        else:
            parent[yroot] = xroot
            rank[xroot] += 1
    def KruskalMST(self):

        result = []
```

```
        i = 0
        e = 0
        self.graph = sorted(self.graph,
                            key=lambda item: item[2])

        parent = []
        rank = []
        for node in range(self.V):
            parent.append(node)
            rank.append(0)
        while e < self.V - 1:
            u, v, w = self.graph[i]
            i = i + 1
            x = self.find(parent, u)
            y = self.find(parent, v)
            if x != y:
                e = e + 1
                result.append([u, v, w])
                self.union(parent, rank, x, y)

        minimumCost = 0
        print("Edges in the constructed MST")
        for u, v, weight in result:
            minimumCost += weight
            print("%d - %d ==> %d" % (u, v, weight))
        print("Minimum Spanning Tree =", minimumCost)

if __name__ == '__main__':
    g = Graph(4)
    g.addEdge(0, 1, 10)
    g.addEdge(0, 2, 6)
    g.addEdge(0, 3, 5)
    g.addEdge(1, 3, 15)
    g.addEdge(2, 3, 4)

    g.KruskalMST()
```

OUTPUT SCREEN :

TIME COMPLEXITY:

All cases - O(Nlog(E)+ Nlog(N))

RESULT:

I have studied and understood the Kruskal's algorithm in python language and executed the program successfully.

THANK YOU !!