Minimum Cost Spanning Tree Using Prim’s Algorithm

Abhilasha R.
M.C.A. Department
Saurashtra University
IGNOU Rajkot Regional
Gujarat – India

Abstract: A spanning tree of a graph is just a sub-graph that contains all the vertices and is a tree (with no cycle). A graph may have many spanning trees.

If the graph is a weighted graph (length associated with each edge). The weight of the tree is just the sum of weights of its edges.

Obviously, different spanning trees have different weights or lengths. Our objective is to find the minimum cost (weight) spanning tree.

Keywords: Graph, Minimum cost spanning tree, Prim’s Algorithm.

I. Introduction

A minimum cost of the spanning tree is spanning tree but it has weights or length associated with the edges and total weight of the tree is a minimum.

Suppose, we have a group of islands that we wish to link with bridges so that it is possible to travel from one island to any other in the group, the set of bridges which will enable one to travel from any island to any other at minimum capital cost to the government is the minimum cost spanning tree.

The minimum cost spanning tree has wide applications in different fields. It represents many complicated real world problems like:

- Minimum distance for travelling all cities at most one (Travelling salesman problem).
- In electronic circuit design, to connect n pins by using n-1 wires, using least wire.
- Spanning tree also finds their application in obtaining an independent set of circuit equations for an electrical network.

The minimum cost of spanning tree can be implemented using two methods:

1. Prim’s Algorithm
2. Kruskal’s Algorithm

In this paper our objective is to find the minimum cost spanning tree using a Prim’s algorithm.

II. Prim’s Algorithm

A Prim’s algorithm is a greedy method which helps us to obtain minimum spanning tree. The Prim’s algorithm uses the concept of sets. Instead of processing the graph by sorting order of edges, this algorithm processes the edges in the graph randomly by building up disjoint sets.
The steps in Prim’s algorithm are as follows:

Prim’s (E, cost, n, t)
{
    Let (K, L) is being an edge of minimum cost in E;
    mincost:=cost[K,L];
    t[1,1]:=K;
    t[1,2]:=L;
    for i:=1 to n do
        if(cost[i,L])<(cost[I,K]) then
            near[i]:=L;
        else
            near[i]:=K;
    near[K]:=near[L]:=0;
    for i:=2 to n-1 do
        { 
            Let j be an index such that near [J]≠ 0 and cost [j, near[j]] is minimum;
            t[i,1]:=j;
            t[i,2]:=near[j];
            mincost:=mincost+cost[j,near[j]];
            near[j]:=0;
            for K:=1 to n do
                if(near[K] ≠) and (cost[K,near[K]]>cost[K,j]) then
                    near[K]:=j
        }
    return mincost;
}
III. Prim’s Algorithm Example

Consider the following graph and shows the various steps involved in the construction of the Minimum Cost Spanning Tree.

**Step: 0**
Connected Graph

![Graph Image]

**Step: 1**
S= {1}
V/S= {2, 3, 4, 5, 6, 7}
Lightest edge = {1, 6}

![Graph Image]

**Step: 2**
S= {1, 6}
V/S= {2, 3, 4, 5, 7}
Lightest edge = {1, 6}

![Graph Image]
Step: 3
S = {1, 6, 5}
V/S = {2, 3, 4, 7}
Lightest edge = {1, 6}, {6, 5}

Step: 4
S = {1, 6, 5, 4}
V/S = {2, 3, 7}
Lightest edge = {1, 6}, {6, 5}, {5, 4}
Step: 5
S = {1, 6, 5, 4, 3}
V/S = {2, 7}
Lightest edge = {1, 6}, {6, 5}, {5, 4}, {4, 3}

Step: 6
S = {1, 6, 5, 4, 3, 2}
V/S = {7}
Lightest edge = {1, 6}, {6, 5}, {5, 4}, {4, 3}, {3, 2}
Step: 7
S= {1, 6, 5, 4, 3, 2, 7}
V/S= {}
Lightest edge = {1, 6}, {6, 5}, {5, 4}, {4, 3}, {3, 2}, {2, 7}

IV. Conclusion

We have to study the minimum cost spanning tree using the Prim’s algorithm and find the minimum cost is 99 so the final path of minimum cost of spanning is {1, 6}, {6, 5}, {5, 4}, {4, 3}, {3, 2}, {2, 7}. The Prim’s algorithm operates on two disjoint sets of edges in the graph. Prim’s has a better running time if both the number of edges and the number of nodes are low.

Prim’s algorithm Operates like Dijkstra’s algorithm for finding the shortest path in a graph.

References

Books:

Web sites:
1. http://delab.csd.auth.gr/~manolopo/graph/Lec9_MST.ppt