Most Central Tree in Graph

Jatin Goel [1], Rohit Raj[2], Mohd. Abdullah [3]

[1]IT Department,Indian Institute of Information Technology,Allahabad
Deoghat,Jhalwa-211015 India
icm2014503@iiita.ac.in
ism2014003@iiita.ac.in
ism2014004@iiita.ac.in

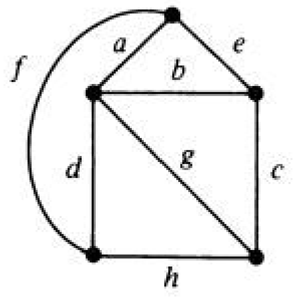
1. Introduction

We are given a simple graph G. We have to find the maximal central spanning tree which is considered to centre of the all the spanning trees generated by from the graph.

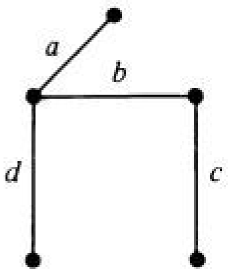
Branches : Set of all the edges of spanning tree
Chords : Set of all the edges in graph excluding the branches.

We will understand the transformation of new spanning tree from the given spanning and the set of chords.

Suppose We have a graph G shown in the figure below:
E = {a,b,c,d,e,f,g,h} // set of edges in graph



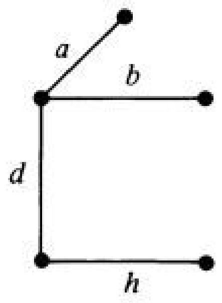
Now We find a basis spanning tree from which all other spanning tree will be generated;



T_1

Spanning Tree having :
Branches = {a,b,c,d}
Chords = {e,f,g,h}

Now to generate other spanning tree we will remove the branch c and add chord h to branches . Now we have new spanning tree :
Branches = {a,b,h,d}
Chords = {e,f,g,c}



T_2

In this way all the Spanning trees can be generated from the Graph.

2. Approach I

Suppose we have a graph G , we find a spanning tree in graph as a our basis spanning tree , we have sets of branches and chords.We can manipulate these two sets to find the all spanning trees then we can find central tree.

For finding one tree from another , we have to remove some branch from the tree and add chord to the set of branches to get new spanning tree.

For finding the most central spanning tree we have to find the maximum of d(ti,tj) for each spanning tree.
d(ti,tj) = No. of non common branches in these two trees.

Then the tree having Minimum of maximum of d(ti,tj) for all trees is the central tree.

First Step – Generating all spanning trees from graph.
In a Graph there are large number of spanning trees, To find the maximal central tree in graph we have to generate all the spanning trees from the graph. To generate all spanning trees we have to start with some spanning tree generated from graph.

We can generate some spanning tree from graph in this way.

```
Find_spanning_tree( )
{
        set = { } // set of spanning tree edges
        For each edge in graph
        {
                If (form_cycle(u,v) == false)
                {
                        set.insert(u,v);
                }
        }
}
```

Now we have to generate all others spanning trees from the graph.

```
Generate_all_spanning_trees()
{
        // set of edges of basis spanning tree
        branches = { }
        // set of all other edges in graph
        chords = { }
        Spanning_Tree = { }
        Spanning_Tree.add(branches);
        for each branch b in tree
        {
            branches.erase(b);
            for each chord c from chords
            {
                if(form_cycle(cu,cv)==false)
                {
                        branches.add(c);
                        Spanning_Tree.add(branches)

                        //We get new spanning tree
                }
            }
        }
}
```

Second Step : Finding the most central spanning tree from the set of trees.

```
find_central_tree()
{
        for each tree
                d[i] = -1;
        for each tree ti in Spanning tree:
                for each tree tj in Spanning tree:
                        d[i]=max(d[i] , d(ti,tj) );
        min = INT_MAX;
        for each tree
        {
                if (min > d[i])
                {
                        central tree = ti;
                        min = d[i];
                }
        }
}
```

3. Approach II

One important point to note is that the bridges in the graph will be present in all the spanning trees.
Hence we can keep them separate.

Now we find out all the bridges in the graph and store them.
After that all the bridges in the graph are removed.
Now the graph will be splitted into Q components.

M=#bridges

Q=M+1

Now for every component we find out all the spanning trees of that component.

Let's say we have M bridges stored in vector Bridges[]
Let's say we have Q components and edges in ith component is stored in vector Edge[i]
Now we form all possible spanning tree for every component.

We are generating spanning trees for ith component as follows :
Lets say there are X edges in ith component and Y nodes.
Therefore any spanning tree of ith component will contain exactly Y-1 edges.

Now we select every possible combination of |Y-1| edges and check if those |Y-1| selected edges will form a tree or not.

If they do not form a tree then discard them.

Else insert those edges in the spanning tree set (TreeSet[i]) for that component.

We repeat this procedure for all the components.
So in the end we will have all possible spanning tree for every component.
Now the goal is to generate all the spanning trees of the original graph

let's say TreeSet[i] is the spanning tree set of ith component , where TreeSet[i][j] denotes jth spanning tree of ith component.
So to form overall spanning tree we do as follows :
SpanningTree=Bridges U TreeSet[1][a] U TreeSet[2][b] U TreeSet[3][c] U ... TreeSet[Q][q]
//where 1<=a<=TreeSet[1].size() , 1<=b<=TreeSet[2].size() and so on.

overallSpanningTree.insert(spanningTree)

overallSpanningTree is the set which stores all spanning trees of the original graph.

After we have generated all the spanning trees of the original graph we can find out the most central spanning tree as follows:

pick two spanning trees T[i] and T[j] from overallSpanningTree

EB[i] = edge bitSet of ith spanning tree

EB[j] = edge bitset of jth spanning tree

ringSumEdgeBitset=EB[i]^EB[j]

dis(T[i],T[j]) = half number of set bits in "ringSumEdgeBitset".

And this way we can find out which is the spanning tree whose maximum distance from any other tree is minimum, and that tree will the most central tree.

Algorithm overview :

For the simplicity of explanation it has been assumed that few of the modules are self explanatory or self understandable.

findBridges(); // complexity O(V+E)

Edge[]=findComponents(); // complexity O(V+E)
//Edge[i] will store edges in ith component

Q = components.size() // number of components

TreeSet[][]=findSpanningTreeOfAllComponents();// O((E)c(V-1)*(V+E)) , a backtrack function.
//TreeSet[i] will store all spanning tree of ith component.Trees will be stored in form of edgeBitsets

overallSpanningTree={} // it stores edge bitsets of all spanning trees of original graph.
overallSpanningTree=generateAll(); // backtracking function to select one spannig tree from each spanning tree and merge them with bridges to get overall spanning tree.

total=overallSpanningTree.size()

```
find_central_tree()
for i : 1 to total
{
        for j : 1 to total
        {
                if(i!=j)
                {
                    EB[i]=overallSpanningTree[i]
                    EB[j]=overallSpanningTree[j]
                    ringSumEdgeBitset=EB[i]^EB[j]
                    distance=rangSumBitSet.count/2
                    maxi=max(maxi,distance)
                }
        }
        if(maxi<mini)
        {
                mini=maxi
                centralTree=i
        }
}
```

4. Conclusion

In this way we have generated all the spanning trees from the given simple graph and found the most central tree from all the spanning trees which can be treated as the centre of the all the spanning trees.