Fundamental Algorithms, Graphs, HW - 11

Tulsi Jain

1. SCC

1.1 Contra-positive

If f(C) < f(C'), where C and C' are two strongly connected components of a graph G and there is no edge, E (u, v) that connects C and C' and u lies in C and v lies in C'.

1.2

Considering one SCC as one node then the given graph would be a DAG. where each node would represent a strongly connected component.
1. find DFS for graph G and computer finishing time i.e. u.f for every node.
2. Call DFS (G) on original graph but in increasing order of u.f
3. The generated forest would be SCC component then repeat step 2.

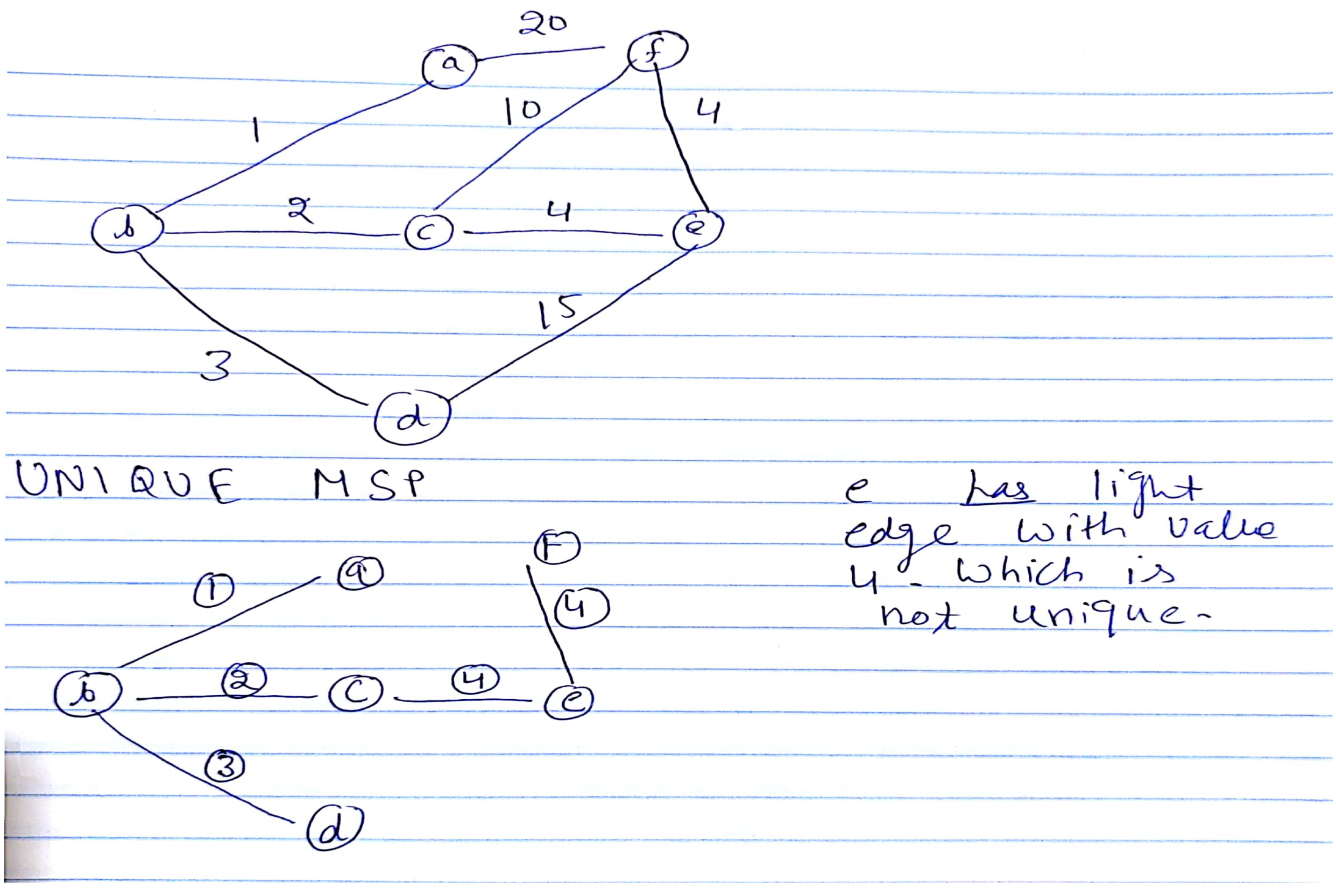
This algorithms looks very simple but it is not practical as in order to find all the vertices connected to X and to validate this forms one complete SCC would take a lot of computation. If this does not forms one SCC then we need to look at the sub problem and look for the same. Hence this algorithms is not practical.

2. MSP

Suppose graph G = (V, E). Cut anywhere in the graph has a unique light weight edge. If we make a cut keeping one vertex on one side and other $|V| - 1$ vertexes on other side. As there is only one light edge connecting the standalone vertex to rest of the graph. If, we do this for every vertex in the graph we would end up unique MSP as there is unique light edge for each cut.

Below given example would produce a unique minimum spanning path but across a cut light edge is not unique.

Figure 2.1: Unique MSP with multiple light edge

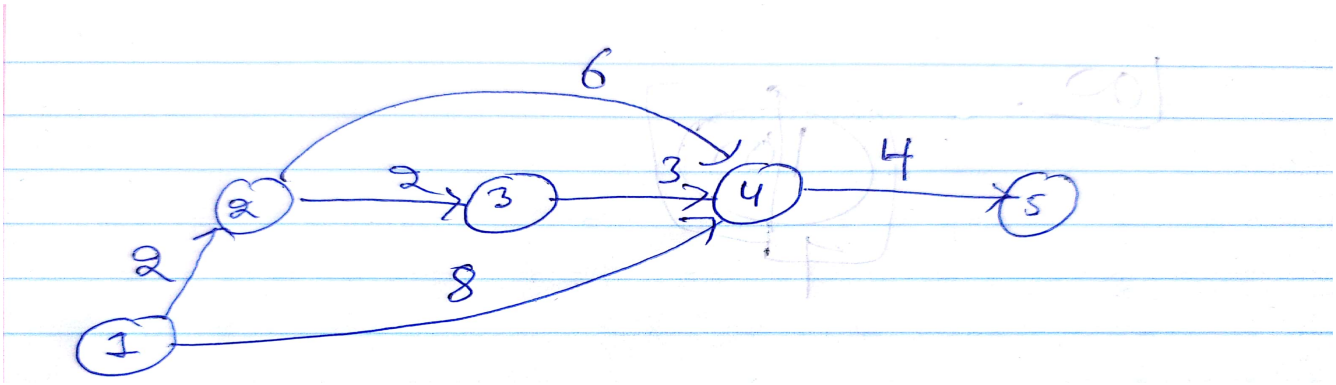


3. Third

3.1

MSP - It would be unique. This can be prove by same argument used in second question. For every cut there exists a unique light edge because all the edges are distinct.

Figure 3.1: Unique MSP with multiple light edge



In above mentioned example (1,2) (2,3) (3,4) and (4,5) is MSP, with weight 24 but different values of second best MSP is possible. It can be either (1,2) (2,3) (2,4) (4,5) with weight 26 or (1,2) (2,3) (1,4) (4,5) with weight 26. Edge(1,2) (2,3) has same weight. The questions says all edges needs to be different but it does not change the proof. If (1,2) is changed to something else solution is still valid.

3.2

This is true, Suppose there are $|V|$ edges, then we can remove the minimum weight edge and find the second MSP.

3.3

We have to computer this in polynomial in $|V|$ time. This can be done in following manner. For a vertex see through all the vertex and records the maximum value. In our solution max[u, v] ensured we visit every nodes only once.

Algorithm 1 MaxBFS

```
function COMPUTE-MAX(G)                                   ▷ where G graph
    tableMax = new int [V][V]
    for u in G.V do
        for v in G.V do
            tableMax[u][v] = NIL
            Queue Q;
            ENQUEUE(Q, u);
            while  Q is not empty do
                x = DEQUEUE(Q)
                for y in G.x do
                    if tableMax[u, v] = NIL and u ! = y  then
                        if x = u or w(x, y) > tableMax[u, x]  then
                            tableMax[u, y] = w(x, y)
                        else
                            tableMax[u, y] = tableMax[u, x]
                        end if
                        ENQUEUE(Q, y)
                    end if
                end for
            end while
        end for
    end for
    return tableMax;
end function
```

3.4

As shown in part b we can obtain a second-best minimum spanning tree by replacing exactly one edge of the minimum spanning tree T by some edge (u, v) that does not lies in T. If we create spanning tree T' by replacing edge (x, y) belongs to T with edge (u, v) does not belong T, then w(T') = w(T) w(x, y) + w(u, v). For a given edge (u, v), the edge (x, y) belongs to T that minimizes w(T') is the edge of maximum weight on the unique path between u and v in T.

By (c), that edge is max[u, v]. Thus, our algorithm needs to determine an edge (u, v) does not belong to T for which w(max[u, v]) w(u, v) is minimum: Algorithms is as follows:
1. Make a MST, T take , Time O(E + V log V).
2.$MakeaMaxtableasdescribeinpartctake, TimeO$(—V—^{2}).
3. Find an edge (u,v) does not belongs to T and max (u,v) - w(u,v) is minimum. Time O($|V|^2$). that minimizes
4. Having found an edge (u,v) in step 3, we replace a edge and return as a second-best minimum spanning tree, Time O($|V|^2$).