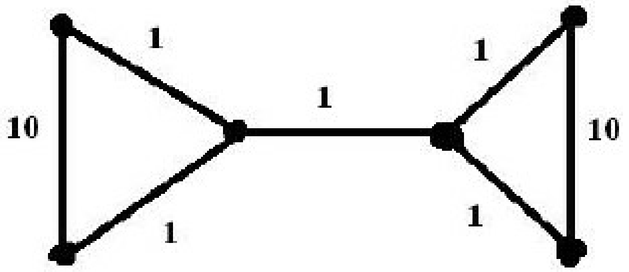
Theory Assignment on BFS, DFS, MST Algorithms

Name-Rohan Ghosh
ID: 181001001097
Batch: BCS2B
Subject-Design and analysis of algorithm
Techno India University, West Bengal

---

1. Give an example of a connected, weighted, undirected graph such that no matter at which vertex you start a DFS (depth first search) the resulting DFS- tree is not a MST (minimum spanning tree), regardless of how the adjacency lists are ordered. Justify the correctness of your example.

Solution 1:



No matter where you start DFS and independently on the order of adjacent vertices in the adjacency list, the DFS will have to enter one of the triangles through the vertex adjacent to the central edge. This means that the spanning DFS sub-tree for this triangle will have a weight of 11, while the minimum weight sub-tree spanning this triangle has the weight of 2.

---

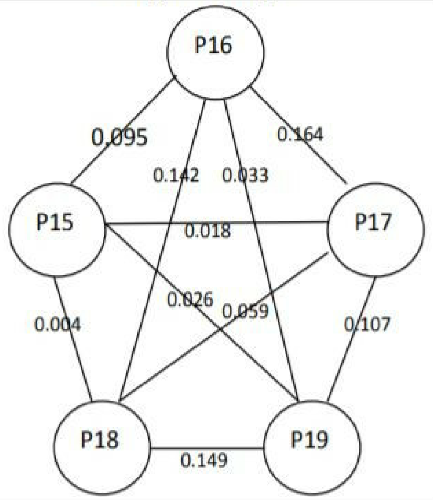
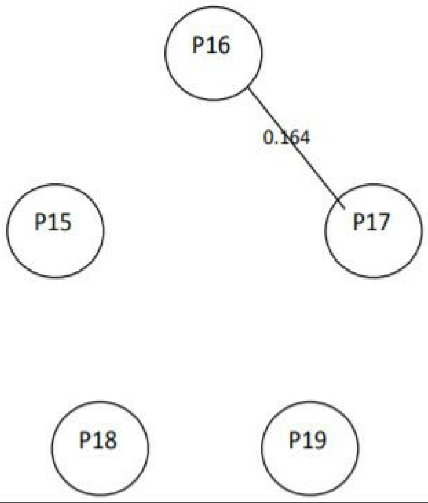
2. Show how to find the maximum spanning tree of a graph, that is, the spanning tree of largest total weight.

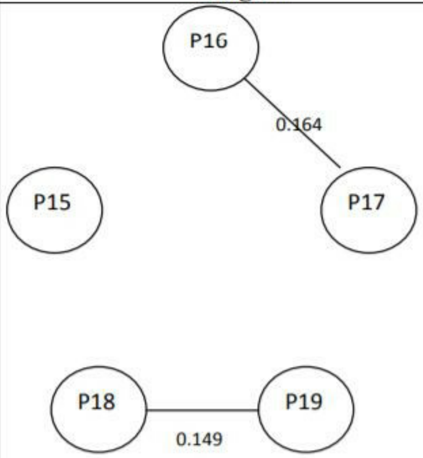
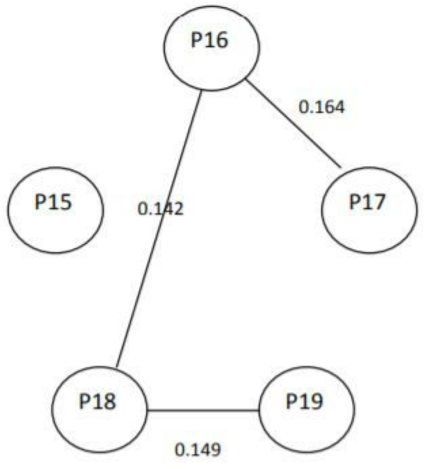
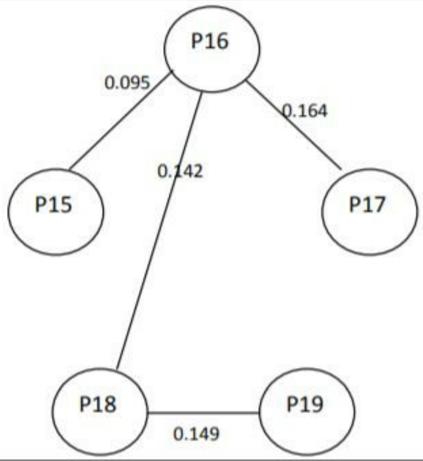
Solution 2:

The Maximum Spanning Tree can be obtained using Kruskal Algorithm which is a greedy approach. Initially all the n vertices without edges are considered so that each vertex is a component. All the edges are arranged in their descending order of costs. The edge with the biggest cost is added to T if and only if the edge connects two different components (i. e) it does not form a loop. The process of adding the edges is done for n-1 times. A single tree is the output obtained from a Kruskal's algorithm. The time complexity of the algorithm is O(mlogn) where m is the number of edges and n is the nodes. This algorithm works best if the number of edges is kept to a minimum.

Algorithm:

```
Let G = (V, E) be the given graph, with | V| = n
{
    T = (V, ϕ ) consisting of only the vertices of G and no edges; Arrange E in the
    order of decreasing costs;
    for (i = 1, i≤n - 1, i + +)
    {
        Select the next biggest cost edge;
        if (the edge connects two different connected components) add the
        edge to T;
     }
}
```

Original Graph	Stage 1
 P16 P15 P17 P18 P19 0.095 0.164 0.142 0.033 0.018 0.026 0.059 0.004 0.107 0.149	

Stage 2	Stage 3
	
Stage 4	
 P16 0.095 0.164 0.142 P15 P17 P18 P19 0.149	

----

3. Give an algorithm that determines whether or not a given undirected graph G = (V, E) contains a cycle. Your algorithm should run in O(|V|) time, independent of |E|.

Solution 3:

An undirected graph is acyclic (i.e., a forest) if and only if a DFS yields no back edges.

- If there's a back edge, there's a cycle.
- If there's no back edge, then by Lemma: An undirected graph is acyclic if and only if a DFS yields no back edges. There are only tree edges, hence, the graph is acyclic.

Thus, we can run DFS: if we find a back edge, there's a cycle.

- Time: O(V). (Not O(V + E)!) If we ever see |V | distinct edges, we must have seen a back edge because in an acyclic (undirected) forest, |E| ≤ |V | − 1.

---X---