DSAD Assignment II

Bishwajit Prasad Gond
222CS3113

Master of Technology
222cs3113@nitrkl.ac.in

Department of Computer Science & Engineering
NIT, Rourkela

September 28, 2022

Contents

1 Question **1**

1.1 Define minimum spanning tree for a Graph. 1

1.2 Write down the objective function and a greedy algorithm to find out the minimum spanning tree for given a weighted connected graph. 1

1.3 Specify and explain the time complexity of your algorithm. Suggest some measures to reduce the time complexity. 2

1.4 Can we design a dynamic programming algorithm for finding minimum spanning-tree for a Graph? 4

1 Question

1.1 Define minimum spanning tree for a Graph.

A spanning tree of a graph is a collection of connected edges that include every vertex in the graph, but that do not form a cycle. Many such spanning trees may exist for a graph. The Minimum Spanning Tree is the one whose cumulative edge weights have the smallest value, however. Think of it as the least cost path that goes through the entire graph and touches every vertex.

Definition. Let $G = (V, E)$ be an undirected Connected graph. A subgraph $t = (V, E')$ of G is a spanning tree ig G if and only if t Is a tree.

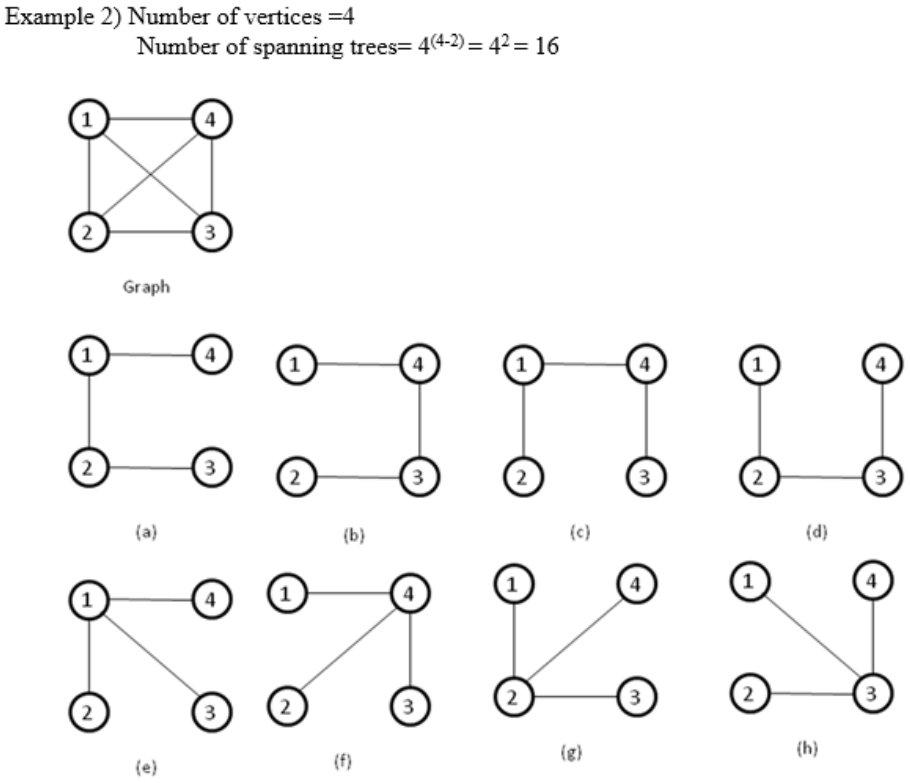

Figure 1: No of Spanning Tree

1.2 Write down the objective function and a greedy algorithm to find out the minimum spanning tree for given a weighted connected graph.

Begin with a graph $G = (V, E), |V| = n$. The conventional Minimum Spanning Tree (MST) approach assigns a cost to each feasible link (i, j) $\in$ E then seeks to minimize the sum of all link costs in the ST, that is to find the minimizing tree T such that

Objective Function:

$$min_T f_{MST}(T) = min_T = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} n_{ij} \tag{1}$$

or all trees T, where c_{ij} is the cost of the link (i, j) and $x_{ij}^T = 1$

Here T represent the tree and c is cost/weight associated with edges and n is number of node [1]

```
1  Algorithms Kruskal(E,cost,n,t)
2  //E is the set of edges in G. G has n vertices. cost[u,v] is the
      cost of edge(u,v). t is the set of edges in the minimum-cost
      spanning tree. The final cost is returned.
3  {
4      Construct a heap out of the edge costs using Heapify;
5      for i:= 1 to n do parent[i]:=-1;
6      //Each vertex is in a different set.
7      i:=0; mincost:=0.0;
8      while ((i<n-1)and (heap no empty)) do
9      {
10         Delete a minimum cost edge (u,v) from the heap and
              reheapify using Adjust;
11         j:=Find(u); k:=Find(v);
12         if(j =! k) then
13         {
14             i:=I + 1;
15             t[i,1]:= u;t[i,2]:=v;
16             mincost:= mincost + cost[u,v];
17             Union(j,k);
18
19         }
20     }
21     if (i=! n-1) then write("No spanning tree");
22     else return mincost;
23
24 }
```

we got the sum of the edges weight as 50.

1.3 Specify and explain the time complexity of your algorithm. Suggest some measures to reduce the time complexity.

Worst case time complexity of Kruskal's Algorithm = O(E log V) or O(E log E)
Analysis

- The edges are maintained as min heap.

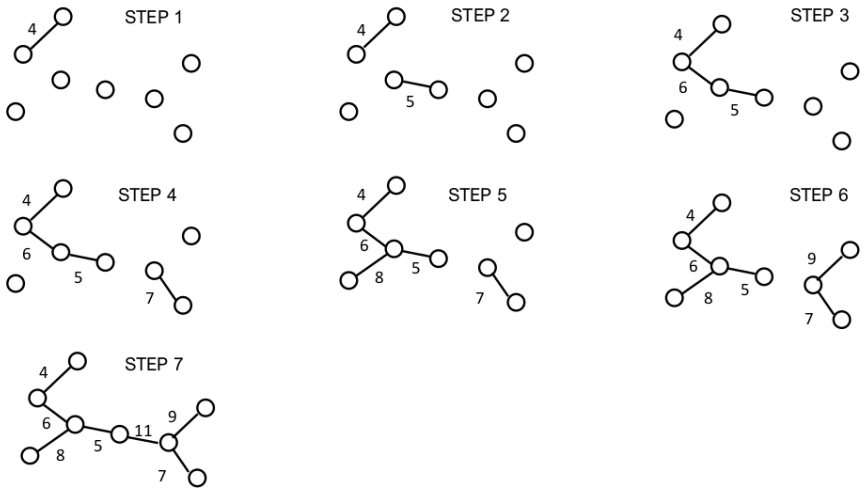

Figure 2: Step-wise edge selection with no cycle in the graph

- The next edge can be obtained in $O(logE)$ time if graph has E edges.
- Reconstruction of heap takes $O(E)$ time.
- So, Kruskal's Algorithm takes $O(ElogE)$ time.
- The value of E can be at most $O(V^2)$.
- So, $O(logV)$ and $O(logE)$ are same.

Special Case

- If the edges are already sorted, then there is no need to construct min heap.
- So, deletion from min heap time is saved.
- In this case, time complexity of Kruskal's Algorithm = O(E + V)

An Optimal Randomized Algorithm to reduce the time complexity [2]

Any algorithm for finding the minimum-cost spanning tree of a given graph G(V,E) will have to spend Ω(V+E) time in the worst case, since it has to examine each node and each edge at least once before determining the correct answer. A randomized Las Vegas algorithm that runs in time $\tilde{O}$(V+E) can be devised as follows:

1. Randomly sample m edges from G (for some suitable m).
2. Let G' be the induced subgraph; that is, G' has V as its node set and the sampled edges in its edge set. The subgraph G' need not be connected. Recursively find a minimum-cost spanning tree for each component of G'. Let F be the resultant minimum-cost spanning forest of G'.
3. Using F, eliminate certain edges (called the F-heavy edges) of G that cannot possibly be in a minimum-cost spanning tree. Let G" be the graph that results from G after elimination of the F-heavy edges.
4. Recursively find a minimum-cost spanning tree for G". This will also be a minimum-cost spanning tree for G.

Steps 1 to 3 are useful in reducing the number of edges in G. The algorithm can be speeded up further if we can reduce the number of nodes in the input graph as well. Such a node elimination can be effected using the Boruvka steps.

1.4 Can we design a dynamic programming algorithm for finding minimum spanning-tree for a Graph?

According to the author Sörensen, Kenneth, and Gerrit K. Janssens [3] it is possible to design a dynamic programming algorithms for finding minimum spanning-tree for a Graph. In their paper [3] they used dynamic programming by partitioning edges into subsets.

References

[1] M. Morgan and V. Grout, "Spanning tree objective functions and algorithms for wireless networks," in *2006 IEEE Sarnoff Symposium*, pp. 1–4, IEEE, 2006.

[2] E. Horowitz, *Fundamentals of computer algorithms*. Galgotia publications, 1978.

[3] K. Sörensen and G. K. Janssens, "An algorithm to generate all spanning trees of a graph in order of increasing cost," *Pesquisa Operacional*, vol. 25, pp. 219–229, 2005.