Analysis of Algorithms - Home Work 2

CSE 548 Fall '19

Prof. Jie Gao

Submission By:

Narayan Acharya

112734365

Contents

Narayan Acharya
112734365

1 Question 1

Textbook [Kleinberg & Tardos] Chapter 3, page 107, problem #6.
Solution:

We know that tree T is both a DFS tree and a BFS tree. So tree T should exhibit properties of both DFS and BFS trees.

For a DFS tree for two nodes **not** to be connected by an existing edge, one must be an ancestor of the other. [See Kleinberg & Tardos, proof 3.7, page 85]

For a BFS tree for two nodes to be connected, their distance from another node w in T can differ by at most 1. [See Kleinberg & Tardos, proof 3.4, page 81]

Suppose, there exists an edge e that connects nodes u and v in our graph G but does **NOT** belong to edges E of tree T, i.e. $e = (u, v)$ and $e \notin E$.

As u and v are connected by an edge **not** in T, one of them is an ancestor to the other. Without loss of generality, assume node u is an ancestor of v. Given T is also BFS tree, distance from random node w in T to u and v can differ by a maximum of 1. Thus u has to be a direct parent of v. This implies that edge e connecting them must be a part of T.

This contradicts our initial assumption of $e \notin E$. Hence $e \in E$.

Narayan Acharya
112734365

2 Question 2

Textbook [Kleinberg & Tardos] Chapter 3, page 107, problem #8.
Solution:

We need to prove that for a connected graph G there exists a positive natural number that c, for which

$$\frac{diam(G)}{apd(G)} \leq c$$

In order to maximize the quantity on LHS we need to increase the diameter($diam$) of graph G without increasing the average pairwise distance(apd) of G. As we look to maximize LHS we might encounter a bound on it.

Notice 2 things:

1. $diam(G)$ is high when it has minimum number of edges i.e. for a graph with n nodes, the maximum number of edges in G are $(n-1)$. This is the max possible diameter of G.
2. $apd(G)$ is less when it has a lot of edges and getting from one node to another is less costly. For a connected graph, where each node has an edge to every other node, the apd to 1.

Consider the following case where we start with a connected graph, G, with n nodes. The $apd(G)$ is 1 now. Next we look to add a path with k nodes to one of the nodes in G. We deliberately connect it to just one node so as to maximize $diam(G)$. The $apd(G)$ will increase as well but we are looking to find a relation between n and k such that the rate at which $diam(G)$ increases is faster than the rate at which $apd(G)$ increases and eventually check if there is a bound on the difference at which there 2 functions grow.
New diameter of G with $(n+k)$ nodes:

$$diam(G) = k + 1 \tag{1}$$

New average pairwise distance of G with $(n+k)$ nodes:

$$apd(G) = \frac{\frac{n(n-1)}{2} + \frac{k(k+1)}{2} + (n-1) \times \left[\frac{(k+1)(k+2)}{2} - 1\right] + \frac{(k-1)(k)(k+1)}{6}}{\frac{(n+k)(n+k-1)}{2}} \tag{2}$$

The first term in the numerator of equation 2 is the addition of distance for n nodes in the connected graph part of G. The second term is addition of the distances for k nodes with the node in G where we connected the path. Third term is addition of distances for k nodes with all the other $n-1$ nodes. Finally we have the fourth term which is for distances of k nodes in the path with other nodes in the path.

Simplifying 2 and dividing 1 by 2 we get:

$$\frac{diam(G)}{apd(G)} = \frac{6(k+1)(n+k)(n+k-1)}{6(n-1)(n+k(k+3)) + k(k+1)(k+5)}$$

Notice that the RHS has 2 variables n and k, independent of each other. For a particular n we can come up with a k such that we exceed any given c, as desired. Thus there is **no** bound on $\frac{diam(G)}{apd(G)}$.

Narayan Acharya
112734365

3 Question 3

Textbook [Kleinberg & Tardos] Chapter 3, page 107, problem #12.
Solution:

We can use directed graphs with topological ordering to judge if the information presented to the ethnographers is internally consistent or not.

For every person P_i, consider their date of birth and the date of death be denoted by b_i and d_i.

The below algorithm returns $True$ if the information presented to the ethnographers is internally consistent else returns $False$. First we construct a directed graph starting with an empty graph and adding nodes along with their edges based on information received for each person P_i. For each fact we add additional edges to our graph. If a fact is of type 1 i.e. person P_i died before person P_j, we add an edge (d_i, b_j) signifying that death of P_i precedes death of P_j. On similar lines, if fact is of type 2 i.e. lifespans of persons P_i and P_j overlapped we add edges (b_i, d_j) and (b_j, d_i) to our graph.

Finally we check if our graphs has any cycles. If our graph does have at least one cycle then there is no date that can be put first on our timeline, indicating that some information presented to us is inconsistent. If our graph does **NOT** have a cycle that means our graphs is a DAG and has topological ordering and can be considered internally consistent.

Algorithm 1 Algorithm to check consistency of time series information

```
1:  procedure CHECK(List People, List Facts)        ▷ Given information of of people and facts.
2:      Graph G ← φ                  ▷ Initialize empty directed graph to hold facts about people
3:      for P_i in People do                                  ▷ Store birth and death dates of People
4:          b_i ← birth of P_i
5:          d_i ← death of P_i
6:          Add nodes b_i and d_i to G
7:          Add edge e ← (b_i, d_i) to G
8:      end for
9:      for F_i in Facts do                                   ▷ Extract information from facts
10:         if F_i about death of P_i before birth of P_j then
11:             Add edge e ← (d_i, b_j) to G
12:         else
13:             if F_i life of P_i overlapped with life P_j then
14:                 Add edge e_1 ← (b_i, d_j) to G
15:                 Add edge e_2 ← (b_j, d_i) to G
16:             end if
17:         end if
18:     end for
19:     if G is a Directed Acyclic Graph then return True
20:     end if
         return False
21: end procedure
```

Narayan Acharya
112734365

4 Question 4

Textbook [Kleinberg & Tardos] Chapter 4, page 190, problem #8.
Solution:

We prove this by way of contradiction. Let us suppose that there are two distinct minimum spanning trees T_1 and T_2 for a graph G. Given that these trees are distinct there has to be at least one edge, say e, that exists in one tree but not the other. Without loss of generality, assume that edge e exists in T_1 but not in T_2.

Now, adding edge e to T_2 will create a cycle in T_2. Given that there is no restriction as such on which edge from G e can be, we choose the one with maximum edge weight. Thus we have an edge with maximum weight in a cycle in a minimum spanning tree. This contradicts Cycle Property of minimum spanning tree which states that such an edge cannot be on any mimimum spanning tree. [See Kleinberg & Tardos, proof 4.20, page 147]

This contradicts our assumption that edge e is in either T_1 or T_2 and hence we can confirm that T_1 and T_2 are the same tree and graph G has unique minimum spanning tree.

Narayan Acharya
112734365

5 Question 5

Textbook [Kleinberg & Tardos] Chapter 4, page 190, problem #21.
Solution:

A minimum spanning tree will have $(n-1)$ edges if there are n nodes in the tree. We are getting a *near* tree, with $(n+8)$ edges, as input and need to output a minimum spanning tree which has $(n-1)$ edges. That means we need to get rid of the extra 9 edges in the *near* tree. These 9 edges are the ones contributing to some cycle in the *near* tree.

In order to find a cycle we can use any of the graph traversal algorithms, BFS or DFS, and modify to keep track of cycles as we walk through the nodes[1]. We can keep detect cycles by keeping track of nodes that we encounter and are already marked as explored. Except for the parent of a node, if we encounter and explored node from any node we have hit a cycle. As soon as we detect a cycle, we drop the edge with maximum weight in that cycle. We can do this because of the Cycle Property [See Kleinberg & Tardos, proof 4.20, page 147] and we are given that all edge weights are distinct. If we do this 9 times, we should be left with the minimum spanning tree we require.

Graph traversal and max edge-weight detection has a complexity of $O(m+n)$ where m is the number of edges and n is the number of nodes in the graph. In this case we have m bound by n, i.e $m \leq (n+8)$ for our *near* tree. Implying that our algorithm is bound by n as $O(n)$. (We only need to apply the same thing a constant number of times, 9, and hence we disregard it.)

Narayan Acharya
112734365

6 Question 6

Textbook [Kleinberg & Tardos] Chapter 4, page 190, problem #27.
Solution:

For 2 minimum spanning tree, say T_1 and T_2, that are neighbors are connected by an edge in H as per definition of H in this problem.

Consider the following sequence of steps: We pick an edge in T_2 that does not exist in T_1 and add it to T_1. This introduces a cycle in T_1. We can delete any other edge on the cycle, barring the one we just added. Notice that when we remove this edge, the edge we removed cannot exist in T_2, else it would have been a cycle in T_2 in the first place. Thus we have removed the edge that did not exists in T_2 and added the missing edge in T_1 to it and given they were neighbors, differing by one edge, now both the trees are same. The sequence of steps can be imagined as traversing an edge in H, to go from one minimum spanning tree to its neighbor.

If two minimum spanning trees are, say, d distance apart. We will need to do the above sequence of steps d times. At each step, we will have a minimum spanning tree which is a neighbor of the minimum spanning tree we started with. Thus, there will exist an edge between these trees in the graph H.

Thus, we can conclude that 2 minimum spanning trees that are d distance apart will be connected in H by path of length d which confirms that H is a connected graph.

Narayan Acharya
112734365

References

[1] Detect cycle in an undirected graph,
`https://www.geeksforgeeks.org/detect-cycle-undirected-graph/`