Assignment 03 - Chapter 03 - Solutions

1 Problem 1

Consider the directed acyclic graph G in Figure 3.10. How many topological orderings does it have?

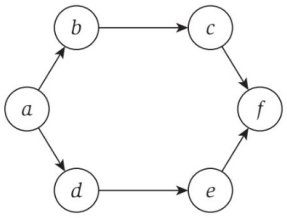

Figure 3.10 How many topological orderings does this graph have?

Solution: There are 6 topographical orderings. They are as follows:

1. a-b-c-d-e-f
2. a-b-d-c-e-f
3. a-b-d-e-c-f
4. a-d-b-c-e-f
5. a-d-b-e-c-f
6. a-d-e-b-c-f

2 Problem 2

Question: Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. (It should not output all cycles in the graph, just one of them.) The running time of your algorithm should be $O(m+n)$ for a graph with n nodes and m edges.

Solution: First, we modify the DFS algorithm so that it records the number of hops away from the starting node for each node as it builds the DFS tree, T. This can be done in $O(1)$ time since this involves simply incrementing a variable and storing that value in a separate array, call it $Distance$. That is, $Distance[u]$ stores the distance of node u from s in the DFS tree. Since recording/storing this value takes $O(1)$, the algorithm still runs in $O(m+n)$.

Now, the algorithm for finding a cycle is as follows:

1. Find all of the connected components within G by repeatedly running the DFS algorithm until all nodes have been covered. Due to the modification to the DFS algorithm, each node has a record of the distance from the starting node, s. This takes $O(m+n)$.
2. Traverse the edges and find an edge, $e=(u,v)$, that exists in G that does not exist in any of the DFS trees of the connected components. $O(m)$.
3. Assume v is farther from the start node than u. Then, traverse backward to v's parent, and then to the parent of that node, etc. Do this repeatedly until we discover node u. Then, append v for the edge, e. This consititutes the cycle. This procedure runs in $O(n)$.

Total runtime for this algorithm is $O(m+n)$.

3 Problem 5

Question: A binary tree is a rooted tree in which each node has at most two children. Show by induction that in any binary tree the number of nodes with two children is exactly one less than the number of leaves.

Solution: By induction. We wish to show that for a tree of n nodes, the number of nodes with two children is exactly one less than the number of leaves. Let's

define the following quantities:

Let n_{0_i} = the number of nodes with 0 children in a binary tree of size i nodes, $i > 0$.

n_{1_i} = the number of nodes with 1 child in a binary tree with i nodes

n_{2_i} = the number of nodes with 2 children in a binary tree with i nodes

Therefore, we have the following:

$$i = n_{0_i} + n_{1_i} + n_{2_i}$$

We want to show that $\forall i > 0, n_{2_i} = n_{0_i} - 1$.

Base Case: First, we will show the proposition is true for a binary tree when $i = 1$ node. Then, we will show that under the assumption the proposition is true for a binary tree with $i = k$ nodes, for some integer $k > 1$, then it must also be true for a binary tree with $i = k + 1$ nodes.

Suppose we have a binary tree with $i = 1$ node. This singe node is a leaf (since it has no children), and thus, we have $n_{0_1} = 1$. The number of nodes with two children is 0, i.e., $n_{2_1} = 0$. Thus, it is true that $n_{2_1} = n_{0_1} - 1 = 1 - 1 = 0$.

Inductive Hypothesis: Now, let's assume we have a binary tree with $i = k$ nodes and that $n_{2_k} = n_{0_k} - 1$ for some $k \geq 1$.

Inductive Step: We need to show that if the assertion is true for some $k \geq 1$ then it *must also be also true* for a tree with $k + 1$ nodes. That is, we must show given our inductive hypothesis, it must also be the case that for a tree of $k + 1$ nodes, $n_{2_{k+1}} = n_{0_{k+1}} - 1$.

So, let's suppose we take the binary tree of k nodes for which our assertion holds and we add one node so that we now have a tree with $k + 1$ nodes. There are two cases to consider:

Case 1 - We add the new node to a leaf node. In this scenario, the number of leaves has not changed since by adding this new node to a leaf, we've simply transformed a former leaf node into a node with 1 child (thereby adding 1 to the count of nodes with 1 child), and added a new leaf node, for a net change of 0 for the number of leaves. Also, the number of nodes with two children remains unchanged. Thus we have the following relationships:

$$\begin{aligned} n_{0_{k+1}} &= n_{0_k} \\ n_{1_{k+1}} &= n_{1_k} + 1 \\ n_{2_{k+1}} &= n_{2_k} \end{aligned}$$

Thus, we have the following:

$$\begin{aligned} n_{2_{k+1}} &= n_{2_k} \\ &= n_{0_k} - 1 && \text{since } n_{2_k} = n_{0_k} - 1 \text{ by the inductive hypothesis.} \\ &= n_{0_{k+1}} - 1 && \text{since } n_{0_k} = n_{0_{k+1}} \end{aligned}$$

Therefore, the relationship holds in Case 1.

Case 2 - We add the new node to a node with 1 child. Under this scenario, the addition of the new node has the following efrects:

1. We've increased the number of leaf nodes by 1.
2. We've reduced the number of nodes with 1 child by 1.
3. We've increased the number of nodes with 2 children by 1.

In mathematical terms, we have the following:

$$\begin{aligned} n_{0_{k+1}} &= n_{0_k} + 1 \\ n_{1_{k+1}} &= n_{1_k} - 1 \\ n_{2_{k+1}} &= n_{2_k} + 1 \end{aligned}$$

Thus, we have the following:

$$\begin{aligned} n_{2_{k+1}} &= n_{2_k} + 1 \\ &= (n_{0_k} - 1) + 1 && \text{since } n_{2_k} = n_{0_k} - 1 \text{ by the inductive hypothesis.} \\ &= (n_{0_k} + 1) - 1 && \text{by re-arranging terms.} \\ &= n_{0_{k+1}} - 1 && \text{by substitution since } n_{0_{k+1}} = n_{0_k} + 1 \end{aligned}$$

Therefore, the relationship also holds in Case 2.

So, in either case, when we add a single node to our binary tree of k nodes so that it subsequently contains $k+1$ nodes, the number of nodes with two children is equal to one less than the number of leaves.

Thus, we have shown by induction that for all binary trees of size $k \geq 1$, the number of nodes with 2 children is one less than the number of leaves.

4 Problem 6

Question: We have a connected graph $G = (V, E)$ and a specfic vertex $u \in V$. Suppose we compute a depth-first search tree rooted at u, and obtain a tree T that includes all nodes of G. Suppose we then compute a breadth-first search tree rooted at u, and obtain the same tree T. Prove that $G = T$. (In other words, if T is both a depth-first search tree and a breadth-first search tree rooted at u, then G cannot contain any edges that do not belong to T.)

Proof: By contradiction. Assume T is both the DFS tree and the BFS tree for G *and* that G contains an edge (x, y) between nodes $x, y \in V$ that does *not* exist in T.

Recall Proposition (3.7) which states the following: Let T be a depth-first search tree, let x and y be nodes in T, and let (x, y) be an edge of G that is not an edge of T. Then, one of x and y is an ancestor of the other.

So, by proposition (3.7), one of the nodes x and y is an ancestor of the other.

Now, recall Proposition (3.4) which states the following: Let T be a breadth-first search tree, let x and y be nodes in T belongining to layers L_i and L_j respectively, and let (x, y) be an edge of G. Then, i and j differ by at most 1.

Now let's consider the fact that T is the BFS tree for G. Since this is true, we know by Proposition (3.4) that since there's an edge, (x, y) between nodes x and y, their respective layers, i and j differ by *at most* 1.

In order for both of these statements to be true at the same time, it must be the case that nodes x and y occupy layers i and j, respectively such that i and j differ by *exactly* 1, (since if they differed by 0, then it would be the case that these two nodes were *siblings*, and thus, *not* ancestor/descendant, as required for the DFS tree).

With a difference in layers of exactly 1, this implies that x and y are parent/child. But if x and y are parent/child, then the edge (x, y) must exist in the tree, T. But our original assumption was that edge (x, y) does *not* exist in T. Thus, we have a contradition. And thus, it must be the case that there can be no edges in G that are not in T.

5 Problem 7

Question: Some friends of yours work on wireless networks, and they're currently studying the properties of a network with n mobile devices. As the devices move around (actually, as their human owners move around), they define a graph at any point in time as follows: there is a node representing each of the n devices, and there is an edge between device i and device j if the physical locations of i and j are no more than 500 meters apart. (If so, we say that i and j are "in range" of each other.)

They'd like it to be the case that the network of devices is connected at all times, and so they've constrained the motion of devices to satisfy the following property: at all times, each device i is within 500 meters of at least $n/2$ of the other devices. (We'll assume n is an even number.) What they'd like to know

is: Does the property by itself guarantee that the network remains connected?

Here's a concrete way to formulate the question as a claim about graphs:

Claim: Let G be a graph on n nodes, where n is an even number. If every node in G has degree at least $n/2$, then G is connected.

Decide whether you think the claim is true or false, and give a proof of either the claim or its negation.

Proof: The claim is true. We will prove this claim by contradiction. Suppose G is a graph consisting of n nodes, (n is even) such that every node has degree at least $n/2$, and suppose that G is *not* connected.

Since G is not connected, there must have *at least* two connected components, R and S, where R and S are disjoint. That is, $G = (V, E)$, $R \subseteq V$, $S \subseteq V$, and $R \cap S = \emptyset$. This implies that $|G| = n \geq |R| + |S|$.

Let's consider a node, $u \in R$. u must have degree at least $n/2$ by our assumption. Thus, $|R| \geq n/2 + 1$. By the same reasoning, let's consider a node, $v \in S$, which also must have degree at least $n/2$. This implies that $|S| \geq n/2 + 1$.

This means we have the following:

$$\begin{aligned} |G| &\geq |R| + |S| \\ &\geq (n/2 + 1) + (n/2 + 1) \\ &\geq n + 2 \\ &> n \end{aligned}$$

But this contradicts our assumption that $|G| = n$. Thus, we have a contradiction. And thus, it must be the case that if G is a graph with n nodes where every node has degree at least $n/2$, then G is connected.