CS 331: Algorithms and Complexity (Spring 2024)

Unique numbers: 51310/51315

Discussion Section 2 Solution

Problem 1

A binary tree is a tree in which each node has at most two children. A full node in a binary tree is a node with two children. Prove by mathematical induction that the number of full nodes is one less than the number of leaves in any non-empty binary tree.

Sol. This is a proof by induction on the number of full nodes N in any non-empty binary tree. We have to prove the following.
If there are N full nodes in a non-empty binary tree then there are $N + 1$ leaves.

Base Case: $N = 0$: If there are 0 full node in a non-empty binary tree (each non-leaf node has only one child), then there is only 1 leaf in the tree.

Induction Hypothesis: If there are k full nodes in a non-empty binary tree then there are $k + 1$ leaves, for any $k \geq 0$.

Inductive Step: We have to show that if there are $k + 1$ full nodes in a non-empty binary tree then there are $k + 2$ leaves.

Pick a leaf node and keep removing it's parent recursively (i.e., remove its parent and then parent's parent and so on) until a full node is reached. That is, you are traversing from a leaf along the path towards the root, while removing the nodes along the path before a full node is reached. This full node becomes a non-full node because one of it's child node is removed. At this point the tree will have one less leaf and one less full node.

The resulting tree is a binary tree (since no new edge has been added) and it has k full nodes. By induction hypothesis, the resulting tree has $k + 1$ leaves. Add all the nodes that were removed back into the tree the same way to create the original tree. We are adding one full node and one leaf node. Therefore, we have $k + 1$ full nodes with $k + 2$ leaves.

Problem 2

We have a connected graph $G = (V, E)$, and a specific 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. Proof by contradiction.
We assume that G and T are not equal. Thus, there exists an edge $\{u, v\}$ such that $\{u, v\}$ is in G but not in T.

Since T is a DFS tree of G, so either node u is an ancestor of node v or node v is an ancestor of node v. Say, u is ancestor of v.

Since T is a BFS tree of G the distance of the two nodes from any node a in T differ by at most one.

Combining these two facts, we obtain that u is a direct parent of v in T. Thus, edge $\{u, v\}$ is present in T which contradicts our assumption that $\{u, v\}$ is in G, but not in T. □

Problem 3

For an undirected and acyclic graph G, prove that the BFS-forest T produced by running BFS on G is identical to G.

Proof. Without loss of generality we consider that G is connected. For disconnected graph, we can prove separately for each component.

Proof by contradiction.
We assume that G and T are not equal. Since T contains all vertices of G and G and T are not equal, then there exists an undirected edge $\{u, v\}$ such that $\{u, v\}$ is in G but not in T. Now find the least common parent of nodes u and v in G. Let this node be a. Since G is connected, then there is a path p_1 from a to u and there is a path p_2 from a to v in G. Now together p_1, edge uv, and p_2 forms a cycle in G. But graph G is acyclic. Therefore, our assumption G and T are not equal does not hold. □

Problem 4

Given a directed graph $G = (V, E)$ such that
$V = \{a, b, c, d, e, f\}$ and $E = \{(a, b), (a, d), (b, c), (d, e), (c, f), (e, f)\}$.
Is G a DAG? If Yes, write down all topological orderings for G.

Sol. Yes G is a DAG because there exists a topological ordering of nodes of G. There are 6 possible topological orderings.

Since node a has no incoming edges, so a comes first. Node f does not have any outgoing edges, so f will come at the end.

Now how to order the middle 4 nodes. If b comes after a, then there are three possible orderings: b, c, d, e or b, d, c, e or b, d, e, c.

Similarly, if d comes after a, then there are three possible orderings: d, e, b, c or d, b, c, e or d, b, e, c.

We get total 6 orderings by inserting a at front and t at the end of each orderings.

We can do a quick sanity check that we have found all possible topological orderings by a counting argument. As before, a comes first and f comes at the end, so there are four remaining nodes to place. The only restrictions on the four remaining nodes are that b must come before c and d must come before e. We can choose two of the four spots to place b and c, in that order, and then the remaining two spots must be d and e, in that order. This is $\binom{4}{2} = 6$ total orderings.