CIT 592 Fall 2021 **Homework 14** **Solutions**

Problem Set

1. **[10 pts]** Let G be a DAG with exactly one source r and such that for any vertex v there exists a unique directed path from r to v. Let G^u be the graph obtained by erasing the direction on each edge of G. Prove that (G^u, r) is a rooted tree.

Solution:

CORRECTED!

Any node x of a tree T can be chosen as a root to make (T, x) into a rooted tree. Therefore it is sufficient to show that the (undirected) graph G^u is a tree, that is, G^u is connected and acyclic.

G^u is connected: Let x, y be two nodes of G^u (therefore also of G). In G there are directed paths $r \twoheadrightarrow x$ and $r \twoheadrightarrow y$. When we erase the directions we obtain two (undirected) paths $r\text{-}\cdots\text{-}x$ and $r\text{-}\cdots\text{-}y$ in G^u. Therefore $x\text{-}\cdots\text{-}r\text{-}\cdots\text{-}y$ is a walk in G^u so any two nodes in G^u are connected.

G^u is acyclic: Suppose toward a contradiction that G^u contains a cycle C. By restoring the erased direction to the edges of C we obtain a (directed) subgraph of G, call it C^d.

Claim C^d is a directed cycle. (Therefore G is not DAG and we have a contradiction. So this is all we need to prove.)

Proof of claim WLOG let the cycle C be described by $a_1\text{-}a_2\text{-}a_3\text{-}\cdots\text{-}a_n\text{-}a_1$ all distinct nodes (we know that an undirected cycle has 3 or more vertices). Also WLOG assume that after restoring direction $a_1\text{-}a_2$ becomes $a_1 \to a_2$.

Subclaim After restoring direction $a_2\text{-}a_3$ becomes $a_2 \to a_3$.

Proof of subclaim Suppose toward a contradiction that, instead, $a_3 \to a_2$. We know that G we have paths $r \twoheadrightarrow a_1$ and $r \twoheadrightarrow a_3$. These give us two paths $r \twoheadrightarrow a_1 \to a_2$ $r \twoheadrightarrow a_3 \to a_2$ which are two distinct paths from r to a_2. Contradiction. **End of proof of subclaim**

Therefore $a_1 \to a_2 \to a_3$. Repeating this argument gives that $a_3 \to a_4$ and so on. (Strictly speaking this should be shown by induction, but it's straightforward.) We conclude that $a_1 \to a_2 \to a_3 \twoheadrightarrow a_n \to a_1$ so C^d is a directed cycle. **End of proof of claim**

2. **[10 pts]** Let G be a digraph with 2 or more vertices which is strongly connected and in which every node has indegree 1 and outdegree 1. Prove that G is a directed cycle.

Solution:

Let u, v be two distinct vertices of G. Since G is strongly connected, there exist two paths $u \to w_1 \to \cdots \to w_m \to v$ and $v \to z_1 \to \cdots \to z_n \to u$. Suppose, toward a contradiction, that there exists some i and j such that $w_i = z_j$. Since the indegrees are all 1 it follows that $w_{i-1} = z_{j-1}$. We can similarly backtrack until some w or some z must equal u or v or $u = v$, none of which is possible. Therefore $u \to w_1 \to \cdots \to w_m \to v \to z_1 \to \cdots \to z_n \to u$ is a directed cycle. We will show that that's the whole G.

Indeed, assume for contradiction that G had another vertex x, distinct from the ones in the directed cycle above. Since G is strongly connected, we must have a path $x \to y_1 \to \cdots \to y_p \to u$. Again, because the indegrees are 1 we must have $y_p = z_n$ then $y_{p-1} = z_{n-1}$ and so on, until x equals one of the vertices in the directed cycle, contradicting our assumption.

3. **[10 pts]** Let G be a DAG with n vertices.

(a) How many strongly connected components are there in G? Justify your answer.

(b) We add a directed edge from every sink of G to every source of G. Let G' be the resulting digraph. How many strongly connected components are there in G'? Justify your answer.

Solution:

(a) There are n strongly connected components, each corresponding to one of the n vertices. Because G is a DAG, two vertices can never belong to the same strongly connected component. Assume for contradiction that vertices a and b are in the same strongly connected component. That means there is a path from a to b and also from b back to a. However, the two paths form a closed walk hence there exists a cycle, contradicting the fact that G is a DAG. Thus, all n vertices must form their own strongly connected component.

(b) There is exactly 1 strongly connected component. We will argue that for any vertex in G, there is a path from that vertex to some sink and also there is a path from some source to that vertex. Once we have proven these two facts, we can connect any vertex a to any other vertex b in the following manner: First take the path from a to some sink. Since every sink is connected to every source, go from that sink to a source that can reach b. Finally take the path from the source to b.

To argue that there is a path from any vertex v to some sink, consider the subset S of natural numbers that are length of directed paths that start in v. We have $0 \in S$ corresponding to the length of the path from v to v. Moreover, since paths cannot repeat vertices there are only finitely many paths in G therefore S is finite. By the Well-Ordering Principle S has a maximum element k. Thus we have a path p that starts in v and whose length is maximum which means that p cannot be extended with another edge. (By the way,

such a path is called *maximal*.) It follows that the node at the end of p is a sink.

Similarly we consider the set of lengths of all the paths that end in v and conclude that there is a path from some source to v.

4. [15 pts] Recall that a rooted tree can be seen as a digraph. This digraph is a DAG with the root as the only source.

Consider the rooted tree (T, r) where $T = (V, E)$ is an undirected tree with nodes $V = \{r, a, b, c\}$ and r is the root. We are given *all* the topological sorts of this DAG:

$$r\ c\ a\ b \qquad\qquad r\ b\ c\ a \qquad\qquad r\ c\ b\ a$$

List the edges in E. Justify your answer.

Solution:

Notice that c follows immediately after r in one of the toposorts (actually in two of them). This must mean that c is a child of r. If this were not true, there would be an arrow "pointing backwards" from a or b back to c.

Similarly, b must be a child of r.

Since the above contains *all* toposorts and a does not follow immediately after r in any of them, a cannot be a child of r. That is, a must be preceded by its parent. Since there is one toposort in which a precedes b but c always precedes a, it must be the case that a is a child of c.

We conclude that $E = \{r{-}c, r{-}b, c{-}a\}$. It's also OK to list the edges as directed: $E = \{r \to c,\ r \to b,\ c \to a\}$

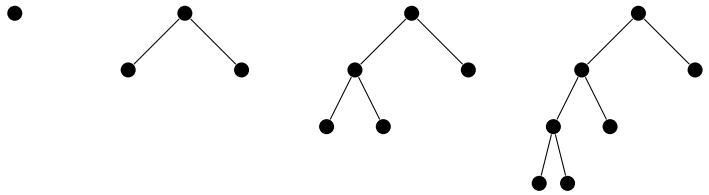
5. [15 pts] A binary tree is *full* if every non-leaf node has exactly two children. For context, recall that we saw in lecture that a binary tree of height h can have at most $2^{h+1} - 1$ nodes and at most 2^h leaves, and

that it achieves these maxima when it is *complete*, meaning that it is full and all leaves are at the same distance from the root. Find $\nu(h)$, the *minimum* number of leaves that a full binary tree of height h can have, and prove your answer using ordinary induction on h. Note that tree of height of 0 is a single (leaf) node.

Hint 1: try a few simple cases ($h = 0, 1, 2, 3, \ldots$) and see if you can guess what $\nu(h)$ is.

Solution:

We draw such minimum-leaf, full binary trees for $h = 0, 1, 2$ and 3:



They have 1, 2, 3 and respectively 4 leaves. Based on this pattern, we make our guess that $\boxed{\nu(h) = h + 1}$.

Note that proof by pattern is not rigorous, but building on small cases helps us build intuition for the problem and what we want to prove. Also, it provides insight that the way to minimize the number of leaves is by making one side of each internal node a leaf node and continuing the tree on the other side. Note that we can also think of this as maximizing the height of a full binary tree given a certain number of nodes.

Now we prove that the minimum number of leaves in a full binary tree of height h is $\nu(h) = h + 1$. "Minimum" means that we have to prove two statements:

(a) Any full binary tree of height h has $h + 1$ or more leaves.

(b) For any h there exists a full binary tree of height h that has exactly

$h + 1$ leaves.

In summary about why two statements are needed, the first statement sets a minimum bound, but the second statement proves there actually exist trees with that minimum bound.

Note to elaborate why the second statement is necessary, proving a full binary tree of height h has at least $h+1$ leaves does not necessarily mean there exists such a tree with exactly $h + 1$ leaves. For example, the first statement could be any full binary tree of height h has h or more leaves, which would technically still be true. However, it would be impossible to show that there exists a full binary tree of height h that has exactly h leaves. Thus, the minimum actually must be a higher number.

As outlined above we begin by proving, by induction on h, that any full binary tree of height h has $h + 1$ or more leaves.

(BC) $h = 0$. There is only one such binary tree and it is 1 leaf.

(IH) FOr some arbitrary $k \in \mathbb{N}$, assume that any full binary tree of height k has $k + 1$ or more leaves.

(IS) Now let T be a full binary tree of height $k + 1$ and let n be its number of leaves. (We want to show that $n \geq (k + 1) + 1$.) Delete all the leaves that are at distance $k + 1$ from the root. There is at least one such leaf, but there cannot be just one because its parent must have two children. Therefore we are deleting at least two leaves of T. However, the parent would now be a leaf so we're back to deleting at least one leaf. The resulting tree, call it T', has height k and it has at most $n-1$ leaves. By IH we have $n - 1 \geq k + 1$ therefore $n \geq (k + 1) + 1$. This ends the induction step.

We now prove the second statement, namely that we can build a full binary tree of height h with $h + 1$ leaves. We do this also by induction on h. Less formal proofs describing (rigorously) a construction are also

acceptable.

(BC) $h = 0$. The tree has one single node which is both root and leaf. $\nu(0) = 0 + 1 = 1$. Yes, it was possible to build a full binary tree of height 0 with 1 leaf!

(IH) For some arbitrary $k \in \mathbb{N}$, assume that we have a full binary tree of height k which has $k + 1$ leaves.

(IS) We want to show that it's possible to build a full binary tree with height $k + 1$ that has $k + 2$ leaves. By IH, we know that a full binary tree of height k can be constructed out of $k + 1$ leaves. There must be a leaf that is at distance k from the root. Call it v. We add two leaves as children of v. The result is still a binary tree, still full, and has height $k + 1$. Note that we removed a leaf by adding children to v, but then added back two leaves. Thus, this new tree has $k + 2$ leaves, completing the induction step.