CSE421: Design and Analysis of Algorithms	April 16, 2021
Lecturer: Shayan Oveis Gharan	Lecture 9

1 DFS Properties

Observation 1. *During $dfs(x)$ every vertex marked discovered is a descendant of x in the DFS tree*

Lemma 2. *If $\{x, y\}$ is a non-tree of the DFS tree, then one of x, y is an ancestor of the other.*

Proof One of x, y will be discovered first in DFS. Wlog say x is discovered first. We want show that x will be an ancestor of y. We claim that y will be discovered while x is in stack (or $dfs(x)$ is still running). The reason is that by the time that x look at its neighbor y in its for loop y must have been discovered already, (because $\{x, y\}$ is a non-tree of the DFS tree). Therefore by observation y is a descendant of x. ■

2 DAGs

Let G be a directed graph; we prove that G is a DAG iff G has a topological sorting.

Lemma 3. *If G has a topological order, the G is a DAG.*

Proof Suppose G has a topological order, i.e., we can name its vertices $1, 2, \dots, n$ put them on a line in order such that all edges go from left to right, i.e., if $i \to j$ is a directed edge of G, we have $i < j$. We prove by contradiction. Suppose G has a cycle $C = c_1, \dots, c_k$. Let c_i be the smallest index among $c_1, \dots, c_k$ in the topological order. Then, $c_{i-1} \to c_i$ is a directed edge in G (if $i = 1$ take the directed edge $c_k \to c_1$). But since c_i is the smallest index vertex in C we must have $c_{i-1} > c_i$. But by definition of the topological order every directed edge of G is from a smaller index to a bigger one. That is a contradiction. So, G is a DAG. ■

In the rest we see that DAGs can be seen as analogues of trees in directed graphs. Many proofs that we did for trees naturally extend to DAGs.

Definition 4. *For a directed graph G, we say a vertex v is a* source *node if the indegree of v is 0 and we say v is a* sink *node if the outdegree of v is 0.*

Lemma 5. *If $G = (V, E)$ is a DAG then it has a source node.*

Proof We prove by contradiction. Suppose G has no source node, so $indeg(v) \geq 1$ for all $v \in V$. Now run the following process:

Start with an arbitrary vertex v_1.

$$indeg(v_1) \geq 1 \Rightarrow \exists v_2 \in V, s.t., v_2 \rightarrow v_1$$
$$indeg(v_2) \geq 1 \Rightarrow \exists v_3 \in V, s.t., v_3 \rightarrow v_2, v_3 \neq v_1 (\text{o.w., } G \text{ has a cycle})$$
$$indeg(v_3) \geq 1 \Rightarrow \exists v_4 \in V, s.t., v_4 \rightarrow v_3, v_4 \neq v_1, v_2 (\text{o.w., } G \text{ has a cycle})$$

Because G has a finite number of vertices after at most $|V|$ iterations we should stop to get either a cycle in G or a node of indegree 0 both of which are contradictions with the assumptions. So, G must have a source node. ■

We said above that DAGs resemble trees, source node resembles leaves. So, similar to trees that we induct by deleting a leaf, we induct in DAGs by deleting source nodes.

Lemma 6. *If G is a DAG, then G has a topological order.*

Proof We prove by induction, and in fact our proof will give an algorithm to construct the topological order. Define $P(n) =$"Any DAG with n vertices has a topological order".

Base Case: $P(1)$ holds. A DAG with a vertex has no edges so the claim obviously holds.

IH: $P(n-1)$ holds for some $n \geq 2$.

IS: We prove $P(n)$. Given an *arbitrary* DAG G with n vertices. By Lemma 5, G has a source node, call it v. Define $G' = G - v$. We claim G' is also a DAG. This is because by deleting vertices/edges we do not introduce cycles. Since G' also has $n-1$ vertices by IH G' has a topological order. So, we can label its vertices say with $1, 2, \ldots, n-1$ such that for every directed edge $i \rightarrow j$ we have $i < j$.

Now, to get a topological order of G, we simply give v the label 0, i.e., we put it at the beginning of the topological order. Since v is a source node, i.e., it has indegree 0, all new edges of v are going out to the rest of the nodes. So, we get a topological order of G such that for every edge $i \rightarrow j$ we have $i < j$. ■