I collaborated with: went to TA sessions

Question
Given a directed graph $G = (V, E)$ with a distinguished vertex $s \in V$ (s for "start"), consider the following 2-person game on G in which players alternate moves: Player 1 begins by placing a stone on vertex s, after which Player 2 can move that stone to any "out" neighbor of s (that is, to any u such that $(s, u) \in E$). After these initial moves, players continue to alternate turns, at each turn moving the stone from its current location x to a new location y such that $(x, y) \in E$. There is one additional rule: the stone can never return to a vertex it has previously visited. A player loses if, when it becomes their turn, they have no move (all out neighbors of the vertex upon which the stone currently rests have already been visited). For some instances (G, s) of this game, Player 1 may be able to force a win (that is, always be able to win regardless of the moves that Player 2 makes), and for some instances, Player 2 might be able to force a win. It turns out that determining whether a player can force a win is NP-hard. This problem appears to be so hard, in fact, that it is not generally believed to be in the class NP!
G (directed acyclic graph) then one can determine, in polynomial time, whether one of the players has a winning strategy.
Hint: Sinks are inherently losing positions. Perhaps you can label each node as inherently winning or losing...?
Solution
Given a graph G that is a DAG, we use its Topological Order in order to label it in a way that helps us determine whether one of the players has a winning strategy. The labeling will happen starting from those vertices with out-degree 0 and finishing with vertices with in-degree 0. We label all vertices with out-degree 0 as a losing vertex, these will be our sinks and losing positions. Now, we label all of the vertices pointing to a sink with as a winning vertex, because those are winning moves. Lastly, we have two cases: either a vertex x has at least one edge to a losing vertex, or all the edges in its out-degree go to winning vertices. In the former case, we label x as a winning vertex and label it as a losing vertex in the latter case. We do this until all vertices are labeled.
Time Complexity: Topological Sorting: $O(n + m)+$ labeling backwards: $O(n + m) = O(n + m)$.
Space Complexity: $O(n + m)$

Proof. We want to show that players can determine winning strategies with the labeling done by the algorithm. We know all Directed Acyclical Graphs can be topologically sorted. We also know that a every DAG G has a vertex with in-degree (out-degree) 0. Thus, we can find sink vertices in G. Inherently, sink vertices are losing positions because there is nowhere to go from them. Therefore, vertices pointing to them are winning vertices, because it leaves the opponent with no moves. Now, we have to cases for labeling the remaining vertices. Either a vertex x has at least one edge to a losing vertex, or all the edges in its out-degree go to winning vertices. This is the case because labeling is done backwards, so x preceeds labeled vertices in the labeling process.
Case 1: If a vertex x has at least one edge to a losing vertex, then the player can just make the move to this vertex and make their opponent lose. This means that x is a winning vertex.
Case 2: If all the edges in x's out-degree go to winning vertices, then the opponent will win in all cases. Thus, x is a losing node.
We have now show that all vertices will be labeled correctly. A player can simply make moves along winning vertices in order to win. □