Problem 1 (Reachability, 30 pts)

Suppose you are given a directed graph $G = (V, E)$ where each vertex, $u \in V$, is labeled with a unique value $L(u)$ from the set $\{1, \ldots, |V|\}$. For each vertex u, let $R(u)$ denote the set of vertices *reachable* from u in G. Define $\max(u)$ to be the *vertex* v_u^* in $R(u)$ with the maximum label, i.e. $L(v_u^*) \geq L(v)$ for all $v \in R(u)$.

Give an $O(|V| + |E|)$ time algorithm to compute $\max(u)$ for all vertices in $u \in V$. Briefly justify correctness and runtime of your algorithm.

Problem 2 (Helping Botanical Garden, 30 pts)

Suppose the New York Botanical Gardens are trying to drum up interest for their tree and shrub collections by offering tours through the arboretum, and you are tasked with reviewing their proposed routes. There are a set V of trees they want some tour to stop at, and a set E of routes from tree to tree their proposed tours would make. In order to make sure all the trees are visited, they want the following property:

For all pairs $u, v \in V$ such that $u \neq v$, we must have a path from u to v ***or*** *a path from v to u (or both).*

Notice how this property is different from strongly connectedness. They provide you and your classmates with many proposed plans and a strict deadline for your feedback, and so you want to develop an efficient algorithm to automate this task.

(a) Suppose $G = (V, E)$ is a directed acyclic graph (DAG). Your classmates propose two possible algorithms to determine whether or not the given graph G satisfies the desired property. For each of the proposed algorithms below, **analyze its runtime and whether it is correct. If it is correct, give a justification. If it is incorrect, provide a counterexample.**

(i) Student A proposes the following algorithm: First, find a source vertex s in the graph that has no incoming edges. If there are multiple sources, pick an arbitrary one. Then, run a **DFS-Visit** (notice this is only the recursive portion of DFS, not the entire DFS) on G starting from s. G satisfies the property if and only if the DFS-Visit visits all the vertices in the graph.

(ii) Student B proposes the following algorithm: First, run DFS on the graph to obtain a topological sort of the vertices. Let the topological sort be $v_1, v_2, \ldots, v_n$. Then, we check for the following edges $(v_1, v_2), (v_2, v_3), (v_3, v_4), \ldots, (v_{n-1}, v_n)$. If all of these edges exist, then we say G satisfies the property. If any of the edges are missing, we say G does not satisfy the property.

(b) Now you want to generalize the algorithm to work for **any** directed graph. Design a $O(|V| + |E|)$ time algorithm to determine whether or not the given graph G satisfies the desired property. You may use the algorithms from part (a) as a subroutine. Make sure to justify the correctness and runtime of your proposed algorithm.

Problem 3 (General MST, 20 pts)

Suppose you are given an algorithm P that can solve the minimum spanning tree (MST) problem on undirected and connected graphs with positive edge weights in time $T(P)$. Show how to use P to solve the general MST problem, in which edge weights are allowed to be negative or 0. That is, show how to reduce the general MST problem to the special case where the graph has only positive weights. The resultant algorithm (including calling P) for MST should take only $O(|V| + |E|) + T(P)$ time.

Problem 4 (MST Updates, 20 pts)

Let G be an undirected and connected graph with positive weights, and T be a minimum spanning tree T of the graph.

(a) Suppose that we decrease the weight of one of the edges in T. Will T still be an MST of G? If yes, provide a proof; if no, provide a counterexample.

(b) Suppose that we **increase** the weight of one of the edges **not** in T. Will T still be an MST of G? If yes, provide a proof; if no, provide a counterexample.