CS180 Homework 3

Due: 8:00pm, 1/31/2019

1. A node v in a directed rooted tree $T = (V, E)$ is an *ancestor* of node u if the path from the root to u passes through v. The *lowest common ancestor(LCA)* of two nodes u and v in T is the deepest node w that is an ancestor of both u and v, where the depth of node is defined as the number of edges on the path from the root to that node. Given two nodes u and v in an directed rooted tree T, design an algorithm that finds the lowest common ancestor in $O(h)$ time, where h is the distance between u and v in the underlying undirected version of T. You are allowed to do $O(|E|)$ preprocessing on the tree before various pairs u and v are given to you, and the cost of preprocessing will not be counted in your LCA algorithm.

2. In some country, there are n flights among n cities. Each city has an exactly one outgoing flight but may have no incoming flight or multiple incoming flights. Design an algorithm to find the largest subset S of cities such that each city in S has at least one incoming flight from another city in S. (hint: recall the in-degree array we used in developing the topological sort algorithm in class)

3. A directed graph $G = (V, E)$ is acyclic if there is no cycle in G. Given a directed graph G, design an $O(|E|)$ algorithm to determine whether it is acyclic or not.

4. Given a directed acyclic graph(DAG) $G = (V, E)$ which is just a directed simple path and $|V| = n$. The topological sorting can be solved on G by doing DFS from the source node and numbering nodes from n down to 1 when the DFS colors a node *black*, i.e., reverse post-ordering on the DFS tree. If we are given another DAG G' which are two node-disjoint simple paths, we can extend G' by creating a super source node and two edges from the super source to the two sources of the paths. The topological sorting on extended G' can be solved if we start a DFS from the super source node and allocate numbers from n down to 0 in the manner described above. The super source node will get the number 0, and all the other nodes will be topologically sorted from 1 to n. Given an arbitrary DAG $G = (V, E)$, extend this idea to give an $O(|E|)$ algorithm to solve topological sorting on G by using DFS. Argue the correctness of your algorithm and its $O(|E|)$ time complexity.

★ Express your algorithm in a well-structured manner. Use pseudo code and the textbook has good examples to follow. Avoid using a long continuous piece of text to describe your algorithm. Start each problem on a NEW page. Unless specified, you should justify the time complexity of your algorithm and why it works. For grading, we will take into account both the correctness and the clarity. Your answers are supposed to be in a simple and understandable manner and sloppy answers are expected to receiver fewer points.

★ Homework assignments are due on Gradescope. Email attachments or paper submissions are NOT acceptable.

★ Raw photo is NOT acceptable. Upload your homework as a PDF scan by using a scanner or mobile scanner app. Match each problem with your answer on Gradescope. Use dark pen or pencil and your handwriting should be clear and legible.

★ We recommend using LaTeX, LyX or other word processing software for writing the homework. This is **NOT** a requirement but it helps us to grade and give feedback.