EECS 336: Introduction to Algorithms — Sample Final
80 minutes

1. Suppose you have an undirected graph with weighted edges, and perform a depth-first search, such that the edges going out of each vertex are always explored in order by weight, smallest first. Is the depth first search tree resulting from this process guaranteed to be a minimum spanning tree? Explain why, if it is, or, if it is not, provide a counterexample (specifying the start vertex for the DFS and showing both trees).

2. The Independent Set decision problem, i.e., deciding whether there is an independent set $S \subset V$ in a graph $G = (V, E)$ with cardinality at least $|S| \geq k$, is NP-complete. (Recall that an independent set is a set S that contains no pair of vertices that share an edge.) Reduce the problem of *finding* an independent set of size at least k to the decision problem.

3. Given an arbitrary *network flow problem* specified by a directed graph $G = (V, E)$, integer capacities $c(e)$ for all $e \in E$, source s and sink t; answer the following true-false questions. If the answer is false, give a counter example. Recall that an edge is *saturated* by a flow f iff $f(e) = c(e)$.

   (a) For any integral max-flow f and e that is not saturated by f, if we were to increase the capacity of e by one then the max-flow value of G must increase.

   (b) For any integral max-flow f and e that is not saturated by f, if we were to increase the capacity of e by one then the max-flow value of G must not increase.

   (c) For any integral max-flow f and e that is saturated by f, if we were to increase the capacity of e by one then the max-flow value of G must increase.

   (d) For any s-t path P in G, if the capacity of each edge in P is increased by one then the max-flow value of G must increase.

   (e) For any s-t path P in G, if the capacity of each edge in P is increased by one then the max-flow value of G increases by at most one.

4. Consider a variant on the Weighted Interval Scheduling Problem where instead of one machine there are two machines.

   Input:

   - n jobs $S = \{1, \ldots, n\}$.
   - two machines a and b.
   - start time s_i and end time f_i for each job i.
   - value v_i for object i if scheduled.

   Output: assignment of jobs to machines where

   - each machine has at most one job scheduled on it at any given time.
   - the total value of scheduled jobs is maximized.

From each job's perspective, it does not matter which machine it is scheduled on, only whether it is scheduled or not.

This problem can be solved with dynamic programming. Answer the following questions. Note: you are not asked to give the final algorithm.

(a) Identify a subproblem that will lead to an efficient dynamic program for the two-machine weighted interval scheduling problem. State that subproblem in English.

(b) Give a recurrence (formula) for calculating the total value of a subproblem from "smaller" subproblems.

(c) What is the runtime of the algorithm that would result from your approach (expressed in terms of n; you do not have to give the algorithm)?

(d) If you were to generalize your algorithm to $k > 2$ machines, what would be the runtime's dependence on k (expressed in terms of n and k; you do not need to give the algorithm)?

5. The Cycle Cover Problem is defined as follows: given a directed graph $G = (V, E)$, determine whether there exists a collection $\{C_1, C_2, \cdots, C_k\}$, where each C_i is a *simple directed cycle* in G (i.e., a cycle with no repeated vertices), such that for *every* vertex v in G, there is a *unique* cycle C_i containing it.

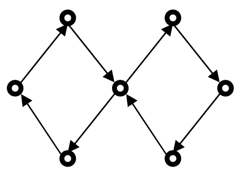
   The 3-Cycle Cover Problem is defined identically to the Cycle Cover Problem except for the additional constraint that each cycle must have length at most three, i.e., $|C_i| \leq 3$ for all i.

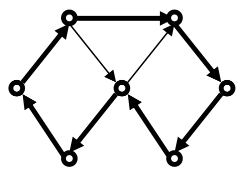
   Examples of "Yes" and "No" instances for the Cycle Cover and 3-Cycle Cover problems. (The cycle covers are displayed in bold.)

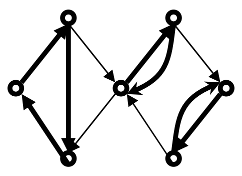
   (a) A "No" instance for Cycle Cover and 3-Cycle Cover;

   (b) A "Yes" instance for Cycle Cover, but a "No" instance for 3-Cycle Cover;

   (c) A "Yes" instance for both Cycle Cover and 3-Cycle Cover.


(a)


(b)


(c)

   Prove that Cycle Cover is in **P**.

6. Prove that 3-Cycle Cover (defined in Problem 5) is **NP**-complete.