Problem Set 7 - Last one! :)
Due: Monday, May 11th, by 10:10am

Problem 1. BELLMAN FORD
(a) [5 pts] Consider the following technique for finding the shortest path distances in a graph with negative edges: find the lowest negative edge weight and add the absolute value of this amount to all of the edge weights, then use Dijkstra’s algorithm as usual to find the shortest path distances. Does this algorithm work? Either prove that it works or give an example where it does not work.
(b) [10 pts] Show the steps of the Bellman-Ford algorithm that we discussed in class on the following graph using vertex s as the source. For each iteration of the for loop, show which nodes have their distances updated and what these distances are (as we did in class). Also give the final predecessor for each node. Assume, as we did in class, that the edges are examined in alphabetic order, so:
(a, c), (a, d), (b, d), (c, a), (d, c), (d, e), (e, c), (s, a), (s, b).

![Graph Diagram]

The Bellman-Ford (BF) algorithm executes an “update” on each edge |V| − 1 times. While this may be required in rare graphs, usually we can find the answer with far fewer updates. We now explore an approach to improve the typical performance of this algorithm. We will keep the same outer for loop (line 3 from the class notes) but improve the inner loop(line 4 from class notes).
(c) [10 pts] Consider two consecutive update steps for a given edge (v, u). Determine a condition, for example about (v, u), v, u, any other component of the graph, or any combination of these (yes, I’m being vague here on purpose) that would ensure that the second update is not needed. Argue your reasoning.
(d) [10 pts] Describe how to use your condition from (c) to modify Bellman-Ford so that it possibly uses fewer than |V| − 1 iterations to find the shortest paths. Also provide brief pseudocode.

Problem 2. GRAPHS/PATHS
(a) [10 pts] Given a directed acyclic graph (DAG), the longest path from a source s to a destination d is a simple (acyclic) directed path from s to t with the maximum length. Describe an O(|V| + |E|)-time algorithm that finds the longest path in a DAG. (Describe the algorithm in words and provide pseudocode [7 pts] ). Justify the running time [3 pts] .
(b) [15 pts] Suppose you have a beach house (lucky you!) which you are going to rent out for the rest of the year. You have solicited bids from interested renters of the form: s_i, f_i, r_i where s_i is the day they want to
start renting, \( f_i \) is the day they want to finish renting, and \( r_i \) is the rent they are willing to pay for their stay.

Only one renter at a time can use the beach house (so if two requested times overlap, only one of them can rent the house). You want to find a set of renters such that no two stays overlap and the total rental income you make is as large as possible.

For example, if the bids are: (Jan 2, March 10, 100), (Jan 10, Feb 7, 60), (Feb. 9, March 25, 90), (March 10, April 10, 80), the optimal solution takes the bids of the first and last renters for a total rent of 180. Note: if a rental ends on day \( f_i \) the next rental can start on day \( f_i \) (like a hotel, rentals end at noon, but don’t start until 4PM).

Carefully describe an efficient algorithm to find the optimal set of rental choices [12 pts]. If you are using a graph, be sure to describe how to construct the graph and then how to use it to solve the problem. Describe the running time of your solution in terms of \( b \), the number of bids [3 pts].

**Problem 3.** NP-COMPLETE PROOF

The S-Path problem takes as input an unweighted, undirected graph \( G \), two vertices \( a \) and \( b \) and an integer \( k \); and asks if \( G \) contains a simple path (i.e. acyclic) of length at most \( k \) from \( a \) to \( b \).

The L-Path problem also takes as input an unweighted, undirected graph \( G \), two vertices \( a \) and \( b \) and an integer \( k \); but asks if \( G \) contains a simple path (i.e. acyclic) of length at least \( k \) from \( a \) to \( b \).

(a)[12 pts] Show that S-PATH is in the class \( P \).

(b)[13 pts] Show that L-PATH is NP-complete.