CS 300 Final Sample Problems

1.(5 pts × 4 = 20 pts) Give True ot False for each of the following statements. Justify your answers. Incorrect justification will earn 0 pts even with correct answer.

(1) A graph algorithm with O(E log V) running time is asymptotically better than an algorithm with O(E logE) running time for a connected, undirected graph G(V, E).
(2) If the depth-first search of a graph G yields no back edges, then the graph G is acyclic.
(3) If some of the edge weights in a graph are negative, the shortest path from s to t can be obtained using Dijkstra's algorithm by first adding a large constant C to each edge weight, where C is chosen large enough that every resulting edge weight will be nonnegative.
(4) Let G = (V, E) be a weighted graph and let M be a minimum spanning tree of G. The path in M between any pair of vertices v_1 and v_2 must be a shortest path in G.

2. (5 x 2 = 10 pts) Consider the following linear-programming system of constraints.

$$x_1 - x_4 \leq -1$$
$$x_1 - x_5 \leq -4$$
$$x_2 - x_1 \leq -4$$
$$x_2 - x_3 = -9$$
$$x_3 - x_1 \leq 5$$
$$x_3 - x_5 \leq 2$$
$$x_4 - x_3 \leq -3$$
$$x_5 - x_1 \leq 5$$
$$x_5 - x_4 \leq 1$$

(1) Draw the constraint graph for these constraints.
(2) Solve for unknowns x_1, $x_{2,}$ $x_{3,}$ $x_{4,}$ and x5, or explain why no solution exists.

4. (10 x 4 = 40 pts) You have an exam with n questions. Each question i has integral point value $c_i > 0$ that requires $m_i > 0$ minutes to solve. Suppose that no partial credit is awarded in this exam. Your goal is to come up with an algorithm which, given c_1, c_2, …, c_n, $m_{1,}$ m_2, … , m_n, and C, computes the minimum number of minutes required to earn

at least C points on the exam. (C is a positive integer.)

(1) Let M(i, c) denote the minimum number of minutes needed to earn c points when you are restricted to selecting from questions 1 through i. Give a recurrence expression for M(i, c). (The base cases: for all i, and $c \le 0$, M(i, c) = 0; for $c > 0$, $M(0, c) = \infty$.)
(2) Give an algorithm to compute the minimum number of minutes required to earn at least C points on the exam and analyze the running time.
(3) Explain how to extend your solution from the previous part to output a list S of the questions to solve such that the total score is at least C and the total number of minutes is minimized.
(4) Suppose now that partial credit is given so that the number of points you receive on a question is proportional to the number of minutes you spend working on it. That is, you earn c_i/m_i points per minute on question i (up to a total of c_i points), and you can work for fractions of minutes. Give an O(n log n)-time algorithm to determine which questions to solve (and how much time to devote to them) in order to receive C points the fastest. Prove the correctness of your algorithm.

5. (5 pts x 6 = 30 pts) Given 2 decision problems L_1 and L_2 in NP and $L_1 \le_p L_2$, for each of the following statements, give one of T(true), F(false), or O(open question), and briefly justify your answer. Incorrect justification will earn no credit.

(1) If $L_1 \in P$, then $L_2 \in P$.
(2) If $L_2 \in P$, then $L_1 \in P$.
(3) If $L_1 \in NPC$, then $L_2 \in NPC$.
(4) If $L_2 \in NPC$, then $L_1 \in NPC$.
(5) If $L_2 \le_p L_1$, then L_1 and L_2 are NP-complete.
(6) Suppose L_2 is solvable in O(n). Then L_1 is also solvable in O(n).

1. (3 pts × 6 = 18 pts) Give True or false. Justify your answer briefly. Incorrect justification will earn 0 pts even with correct T/F.

(1) (T/F) We can use a 2-approximation algorithm for the minimum vertex cover problem as a 2-approximation algorithm for the maximum clique problem.

(2) (T/F) If one NP-complete problem can be solved in polynomial time, all of the NP-complete problem can be solved in polynomial time.
(3) (T/F) Suppose Q is in NP, but not necessarily NP-complete. Then, a polynomial-time algorithm for 3-SAT would necessarily imply a polynomial-time algorithm for Q.
(4) (T/F) NP-complete problems can be reduced to any other NP-complete problem.
(5) (T/F) If SAT problem is in P, then $\mathrm{co}-\mathrm{NP} \neq \mathrm{P}$.
(6) (T/F) Given a graph with negative weights but no negative-weight cycles, reweighting a graph as in Johnson's algorithm can be used to solve the single-source shortest paths problem more efficiently than Bellman-Ford algorithm.

2. Let G = (V, E) be an undirected graph. A strongly independent set is a subset S of vertices such that for any two vertices, u, v ∈ S there is no path of length ≤ 2 between u and v.
Consider the following Strongly Independent Set (SIS) problem :
Given an undirected graph G = (V, E) and an integer k, does G have a strongly independent set of size k?
For each following question, give Yes or No or Unknown. Justify your answer
(1) (4 pts) Is SIS ∈ P?
(2) (4 pts) Is SIS ∈ NP?
(3) (4 pts) Is SIS ∈ co-NP?
(4) (10 pts) Is SIS ∈ NP-hard?
(5) (5 pts) Is SIS ∈ NP-complete?

3. (10 pts) Let $Ax \leq b$ be a system of m difference constraints in n unknowns. Show that the Bellman-Ford algorithm, when run on the corresponding constraint graph, maximizes $\sum_{i=1}^{n} x_n$ subject to $Ax \leq n$ and $x_i \leq 0$ for all x_i.

4. Suppose that a certain country has the coins with the following denominations v(1) < v(2) < ... < v(k) (all integers). Given an integer n, we want to find the minimum number of coins to make n. (You can assume that v(1)= 1 to make any amount n.)
(1) (10 pts) Consider the greedy algorithm which repeatedly takes the largest coin possible. For the set of denominations {50, 25, 10, 5, 1}, prove that the greedy algorithm always gives the minimum number of coins.
(2) (5 pts) Give the set of denominations for which the greedy algorithm does not give the minimum number of coins.

(3) (10 pts) Give an efficient algorithm to solve the problem for any set of k different denominations. What is the running time of your algorithm?