CS140: Exam 2

This is a 3 three hour timed exam. You may not discuss the exam with anyone. You may use a one page cheat sheet but no text or other notes during this exam. You may not use a calculator during the exam.

This exam consists of 4 problems and 7 pages. Make sure your exam is complete. Each problem is worth 25 points. Your answers will be judged for clarity, conciseness, and correctness. The space provided should be adequate for your solutions.

Write your name in the space below.

Name ____

Write the times you started and stopped working on the exam in the spaces below.

Start time: __________________________

Stop time: __________________________

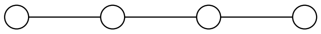
1. A wall in my office has a number of nail holes, n to be exact, running across it at eye level. I have found some posters I'd like to hang to cover the holes but I'd like use as few posters as possible. Each poster is 20" wide. Design an algorithm that determines the minimum number of poster I need to cover the n holes.

   You may assume the holes are infinitesimally small, all holes occur at the same height, and that the i-th hole is x_i inches from the left edge of the wall. Each poster, when hung, will cover a space exactly 20" wide.

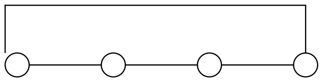
   To receive full credit your algorithm should run in $O(n)$ time. No credit will be given for exponential time algorithms.

2. A graph G is *2-colorable* if the vertices can be colored *red* and *blue* so that no two vertices connected by an edge have the same color.

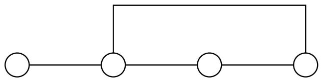
   (a) For each of the following graphs determine whether the graph is 2-colorable and, if so, give a valid 2-coloring.



G1



G2



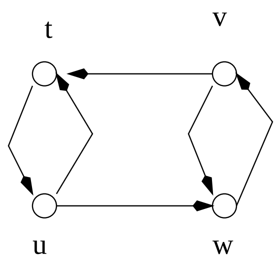
G3

   (b) Let G be a graph containing vertex v. Prove that if G is 2-colorable then there is a 2-coloring of G in which v is colored *red*.

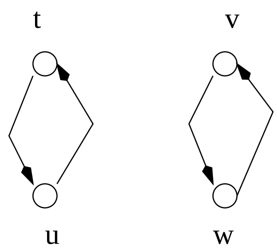
   (c) Let G be a 2-colorable graph. If G has K connected components, how many valid colorings does G have? Prove your answer.

(d) Design an algorithm that takes an input graph G and outputs a 2-coloring, if one exists, and otherwise outputs the statement "G is not 2-colorable." To receive full credit your algorithm should run in time linear in the size of the graph. No credit will be given for exponential time algorithms.

3. A cycle cover of a directed graph is a collection of disjoint, simple cycles that touch each vertex. By disjoint we mean no vertex is in two different cycles. By simple we mean that no cycle visits a vertex more than once. In other words, every vertex has one in-edge and one out-edge in the set of cycle edges. Consider graph G_1 and one if its cycle covers.



G1

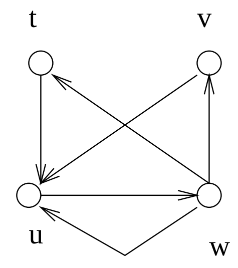


Cycle Cover of G1

(a) Find another cycle cover of G_1.

(b) Clearly some, but not all graphs, have cycle covers. Give a graph that does not have a cycle cover.

(c) Find a largest subgraph of the graph G_2 that has a cycle cover. By largest subgraph we mean one with the most vertices. (You need not include any edges of G_2 that are not in the cycle cover.)



G2

(d) The problem of finding a largest subgraph containing a cycle cover is hard, in general, but can be solved efficiently for some special cases. Design a recursive algorithm that takes as input a graph G in which each vertex has out-degree 1 and outputs a largest subgraph of G that has a cycle cover. (Note: the in-degree of each vertex in G need not be one.) To receive full credit your algorithm should run in linear time when implemented with appropriate data structures. (You need not fill in the implementation details.) No credit will be given for an exponential time algorithm.

4. The input to this problem is n pairs of integers $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. The objective is to choose one integer from each pair so as to maximize the product of the selected integers. For example, given the pairs $(-2, 4), (2, 9), (-3, 1)$ we can choose $-2, 9$ and -3 to achieve the product 54. No other selection yields a larger product.

   (a) Find the optimal selection for the following input.

      i. $(1, 5), (2, 3)$

      ii. $(-1, 3), (-9, 2)$

      iii. $(-1, 3), (1, 5), (-9, 2), (2, 3)$

   (b) Design an algorithm for this problem. Hint: Try self-reduction, strengthen the inductive hypothesis as needed, then use dynamic programming.

      To receive full credit your algorithm should run in time $O(n)$.