UNIVERSITY OF CALIFORNIA, RIVERSIDE
DEPARTMENT OF COMPUTER SCIENCE

SECOND 2003 DEPTH EXAMINATION IN THEORY OF
OF COMPUTATION AND ALGORITHMS

- This is a closed-notes exam. You may only consult books during the exam.
- Each problem is worth 10 points.
- Answer exactly 7 out of 10 questions. Clearly mark which seven problems you want to have graded.
- Write legibly. What can’t be read won’t be credited.
- Algorithms can be described informally using pseudo-code. Remember to analyze the time complexity of your solution.
- Results should be available later this week.
- Good luck!

Name:
Problem 1. You are given \( m \) closed intervals \( I_i = [s_i, t_i] \), for \( i = 1, \ldots, m \), and \( n \) points \( x_1, \ldots, x_n \). Give an efficient algorithm that will compute the minimum number of intervals whose union contains all the points \( x_1, \ldots, x_n \). (If some point \( x_j \) does not belong to any given interval \( I_i \), then print an error message.)

Problem 2. Assume that you are given an integer \( x \) and two arrays \( A \) and \( B \) (not sorted) that contain a total of \( 2n \) integers \( (n \text{ each}) \). Write an algorithm to determine whether there exists an element in \( A \) and an element in \( B \) such that the sum of the two elements is equal to \( x \). (A straightforward algorithm takes \( O(n^2) \) time, so to receive any credit, your algorithm must be faster than \( O(n^2) \)).

Problem 3. Given a sequence of numbers \( a_1, \ldots, a_n \), and parameter \( k \), partition the sequence into \( k \) segments \( S_1 = (a_1, \ldots, a_{i_1-1}), S_2 = (a_{i_1}, \ldots, a_{i_2-1}), \ldots, S_k = (a_{i_k-1}, \ldots, a_n) \), such that \( \max_{1 \leq j \leq k} d(S_j) \) is minimized, where \( d(S_j) \) is the difference between the maximum and minimum numbers in segment \( S_j \). Give an efficient (dynamic programming) algorithm for this problem.

Problem 4. Show how to find a minimum vertex cover on a bipartite graph \( G \) in polynomial time. Hint: You may want to reduce the problem to bipartite matching.

Problem 5. Assume that you are given an unsorted list of \( n \) integers \( a_1, a_2, \ldots, a_n \). Let \( d < n \) be an integer and suppose that the list has the property that the location of element \( a_i \) after the list has been sorted is at most \( d \) positions away from the current position \( i \); that is, if \( a_i \) ends up in position \( j \) after the sorting, then \( |i - j| \leq d \). Describe an algorithm to sort the list \( a_1, a_2, \ldots, a_n \) in \( O(n \log d) \) time when the value of \( d \) is given to you as part of the input. For convenience, you may even assume that \( d \) divides \( n \).

Problem 6. Formulate the graph 3-coloring problem (i.e. determine if an undirected graph can be colored with at most 3 colors) as an integer linear program. In other words, for any graph \( G \), you need describe how to set up an integer linear program \( P \) (consisting of some integer variables and linear constraints/inequalities) such that \( P \)'s feasible solutions correspond to (valid) 3-colorings of \( G \).

Problem 7. Which of the following problems are decidable? Either give a reduction showing undecidability or describe an algorithm deciding the problem.

1. Given a Turing machine \( M \), does \( M \) ever print a non-blank symbol when started on a empty tape?

2. Given a Turing machine \( M \), does there exist an input that \( M \) accepts in an odd number of moves?

3. Given a Turing machine \( M \) and input \( x \), does \( M \) halt on \( x \) within \( 2|x| \) moves?

Problem 8. Prove or disprove: If \( L \) is context-free then

\[
L' = \{ww^R \mid w \in L\}
\]

is also context-free. \((w^R \text{ is the reversal of string } w)\).
Problem 9. Prove that the Hamiltonian cycle problem on (undirected) bipartite graphs is NP-complete:

BiPHC:
Instance: A bipartite graph $G = (V_1, V_2, E)$.
Query: Does $G$ have a Hamiltonian cycle?

You may assume that the Hamiltonian cycle problem on (undirected) graphs is NP-complete.

Problem 10. In the Reachability problem we are given a directed graph $G$ with two vertices $s, t$ and we want to determine whether there is a (directed) path from $s$ to $t$ in $G$. Let AReachability be the restriction of Reachability to directed acyclic graphs (DAGs). Prove that Reachability reduces to AReachability in logarithmic space.