CS 1510 Midterm 2
Fall 1996

1. The input to this problem is a collection $A = \{a_1, \ldots, a_n\}$ of n points on the real line. The problem is to find a minimum cardinality collection S of unit intervals that cover every point in A. Another way to think about this same problem is the following. You know a collection of times (A) that trains will arrive at a station. When a train arrives there must be someone manning the station. Due to union rules, each employee can work at most one hour at the station. The problem is to find a scheduling of employees that covers all the times in A and uses the fewest number of employees. Prove or disprove that the following algorithm correctly solves this problem. Let I be the interval that covers the most number of points in A. Add I to the solution set S. Then recursively continue on the points in A not covered by I.

2. The input to this problem is a collection $A = \{a_1, \ldots, a_n\}$ of n points on the real line. The problem is to find a minimum cardinality collection of unit intervals that cover every point in A. (This is exactly the same problem as the previous one.) Prove or disprove that the following algorithm correctly solves this problem. Let a_j be the smallest (leftmost) point in A. Add the interval $I = (a_j, a_j + 1)$ to the solution set S. Then recursively continue on the points in A not covered by I.

3. Show that the independent set problem is decision reducible. In the decision version of the independent set problem the input is a graph G and an integer k, and the problem is to determine if G contains an independent set of size k. In the optimization version of the independent set problem the input is a graph G, and the problem is to find the largest independent set in G. An independent set I has the property that no pair of vertices in I are adjacent in G. To show that that the independent set problem is decision reducible you need to show that if the decision version of the problem has a polynomial time algorithm then the optimization version of the problem has a polynomial time algorithm.

4. Show that if there is a polynomial time algorithm for the graph 4-coloring problem then there is a polynomial time algorithm for the graph 3-coloring problem. The input for the graph 3-coloring problem is a graph G, and the problem is to determine if each vertex of G can be colored by one of 3 colors so that no pair of adjacent vertices receive the same color. The input for the graph 4-coloring problem is a graph G, and the problem is to determine if each vertex of G can be colored by one of 4 colors so that no pair of adjacent vertices receive the same color.

5. Design a polynomial time algorithm that given two sequences $A = a_1, \ldots, a_m$ and $B = b_1, \ldots, b_n$ produces the shortest sequence C such that A is subsequence of C and B is a subsequence of C. Note that a sequence $A = a_1, \ldots, a_m$ is a subsequence of $C = c_1, \ldots, c_k$ if and only if there exists a function f from $\{1, \ldots, m\}$ to $\{1, \ldots, k\}$ such that $f(i) < f(j)$ for $i < j$ and $a_i = c_{f(i)}$ for all i. So for example, *aba* is a subsequence of *accabccac* but *cbaca* is not a subsequence of *accabccac*.