CS140: Algorithms

Fall 2000

Assignment 6

This assignment is due at the beginning of class on Wed., Oct. 11. You may collaborate with your classmates to complete this assignment but you must reference your collaborators. In addition, you must do your own write-up. This assignment is worth 100 points (50 points if handwritten). Your answers will be judged for clarity, conciseness, and correctness. You algorithms and bounds should be, respectively, as fast and as tight as possible.

1. An *independent set* in a graph $G = (V, E)$ is a subset of vertices, $W \subseteq V$, such that no pair of vertices in W have an edge in G. The problem of finding a largest independent set of a graph is NP-hard. The problem is tractable, however, when the input is restricted to forests.

   (a) Let F be a forest with n nodes. Suppose v is a leaf in some tree of F. Prove that some largest independent set of F contains v.

   (b) Suppose you are given a black box that will compute a largest independent set for any forest containing $n-1$ or fewer nodes. How could you use this black box to help construct a largest independent set for a forest containing n nodes.

   (c) Give a recursive algorithm for finding a largest independent set in a forest. (Be sure to specify your base cases.)

   (d) What is the running time of your algorithm. Give a brief sketch of the data structures you need to achieve this running time.

2. A boolean formula is *satisfiable* if there is some truth assignment for the underlying variables that causes the formula to evaluate to True. For example, the formula $(x_1 + x_2)(\bar{x_1} + \bar{x_2})$ is satisfiable; $x_1 = T, x_2 = F$ is a satisfying assignment. On the other hand $(x_1 + x_2)(\bar{x_1} + x_2)(x_1 + \bar{x_2})(\bar{x_1} + \bar{x_2})$ is not satisfiable; none of the four possible truth assignments cause the formula to evaluate to True.

   The problem of determining whether or not a boolean formula is satisfiable is NP-complete. There are some restricted versions of the problem, however, that are tractable. One is called 2-SAT. The input to 2-SAT is a boolean formula such that the formula is in *conjunctive normal form* and each clause contains at most two *literals*. *Conjunctive normal form* means the formula consists of some "or" clauses that are "anded" together. A *literal* is one of the underlying variables or its complement. The two examples given earlier are both instances of 2-SAT.

(a) First consider the problem 1-SAT; here the formula is in conjunctive normal form and each clause has exactly one literal. For example $(x_1)(x_2)(\bar{x_2})$ is a 1-SAT formula. Give necessary and sufficient conditions for a 1-SAT formula to be satisfiable.

(b) Suppose you have a black box that determines whether or not a 2-SAT formula with $n-1$ or fewer variables is satisfiable. How could you use this box to determine whether or non a 2-SAT formula with n variables is satisfiable.

(c) Give an algorithm for solving 2-SAT. (Your algorithm only has to answer Yes or No; it doesn't have to construct a satisfying assignment.

(d) What is the running time of your algorithm?