CS 473: Fundamental Algorithms, Spring 2011

Discussion 12

April 19, 2011

1. From Set Cover to Monotone SAT.

   Consider an instance of the satisfiability problem specified by clauses $C_1, C_2, \ldots, C_k$ over a set of boolean variables $x_1, x_2, \ldots, x_n$. We say that the instance is ***monotone*** if each term in each clause consists of a nonnegated variable i.e. each term is equal to x_i, for some i, rather than $\overline{x_i}$. They could be easily satisfied by setting each variable to 1. For example, suppose we have three clauses $(x_1 \vee x_2), (x_1 \vee x_3), (x_3 \vee x_2)$. These could be satisfied by setting all three variables to 1, or by setting x_1 and x_2 to 1 and x_3 to 0.

   Given a monotone instance of satisfiability, together with a number k, the problem *Monotone Satisfiability* asks whether there is a satisfying assignment for the instance in which at most k variables are set to 1. Give a polynomial time reduction from Set Cover to monotone satisfiability and prove that it is correct.

   (The Set Cover problem asks, given a collection of subsets $S_1, S_2, \ldots, S_m$ of a set S, what is the size of the smallest subcollection whose union is S?)

2. SAT is a decision problem that asks whether a given Boolean formula in conjunctive normal form has an assignment that makes the formula true. The 3-Coloring problem is a decision problem that asks given an undirected graph G, can its vertices be colored with three colors, so that every edge touches vertices with two different colors? Give a polynomial time reduction from 3-coloring to SAT.

3. In the 2SAT problem, you are given a set of clauses, where each clause is the disjunction (or) of two literals (a literal is a Boolean variable or the negation of a Boolean variable). You are looking for a way to assign a value `true` or `false` to each of the variables so that *all* clauses are satisfied—that is, there is at least one true literal in each clause. For example, here's an instance of 2SAT:

   $$(x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_3) \wedge (x_1 \vee x_2) \wedge (\overline{x}_3 \vee x_4) \wedge (\overline{x}_1 \vee x_4) \quad .$$

   This instance has a satisfying assignment: set x_1, x_2, x_3, and x_4 to `true`, `false`, `false`, and `true`, respectively.

   (a) Are there other satisfying truth assignments of this 2SAT formula? If so, find them all.

   (b) Give an instance of 2SAT with four variables, and with no satisfying assignment.

We can solve 2SAT efficiently by reducing it to the problem of finding the strongly connected components of a directed graph. Given an instance I of 2SAT with n variables and m clauses, construct a directed graph $G_I = (V, E)$ as follows.

- G_I has $2n$ nodes, one for each variable and its negation.
- G_I has $2m$ edges: for each clause $(\alpha \vee \beta)$ of I (where α, β are literals), G_I has an edge from the negation of α to β, and one from the negation of β to α.

Note that the clause $(\alpha \vee \beta)$ is equivalent to either of the implications $\overline{\alpha} \Rightarrow \beta$ or $\overline{\beta} \Rightarrow \alpha$. In this sense, G_I records all implications in I.

(c) Carry out this construction for the instance of 2SAT given above, and for the instance you constructed in (b).

(d) Show that if G_I has a strongly connected component containing both x and $\overline{x}$ for some variable x, then I has no satisfying assignment.

(e) Now show the converse of (d): namely, that if none of G_I's strongly connected components contain both a literal and its negation, then the instance I must be satisfiable. (*Hint*: Assign values to the variables as follows: repeatedly pick a sink strongly connected component of G_I. Assign value `true` to all literals in the sink, assign `false` to their negations, and delete all of these. Show that this ends up discovering a satisfying assignment.)

(f) Conclude that there is a linear-time algorithm for solving 2SAT.