CS 6150: HW3 – NP Completeness, Intro to Graphs

Submission deadline: Friday, Oct 21, 2016, 11:59PM

This assignment has 5 questions, for a total of 50 points. Unless otherwise specified, complete and reasoned arguments will be expected for all answers.

Question	Points	Score
Easy relatives of 3-SAT	10	
Decision vs Search	5	
Reductions, reductions	15	
Graphs – definitions	10	
Weary traveler	10	
Total:	50	

Question 1: Easy relatives of 3-SAT .... **[10]**

Recall the CNF-satisfiability problem we saw in class, where we are given a formula ϕ, which is an AND of a bunch of clauses, and each clause is an OR of literals (x_i, $\overline{x_i}$, for variables $\{x_i\}_{i=1}^n$). We defined a 3-CNF formula as one in which every clause is the OR of precisely three literals. We showed that the 3-SAT problem, which asks to determine of a 3-CNF formula ϕ is satisfiable, is NP-complete.

Let us now consider **2-CNF** formulas, in which every clause has precisely two literals. Interestingly, the satisfiability problem for 2-CNF formulas (called 2-SAT) has a polynomial time algorithm! [An example of a 2-CNF formula in 3 variables: $\phi = (x_1 \vee x_2) \wedge (\overline{x_1} \vee x_3) \wedge (x_2 \vee \overline{x_3})$.]

(a) [**2**] Consider one clause of a 2-CNF formula, say $x_1 \vee x_2$. Prove that it is logically equivalent to the clause ($\overline{x_1} \implies x_2$). [Those with questions about what this means, please post on the Canvas discussion forum.]

(b) [**4**] Clauses in the *implication* form naturally lead to a directed graph. Consider a graph in which the vertices are literals (thus for n variables there are $2n$ vertices, corresponding to x_i and $\overline{x_i}$, for $1 \le i \le n$). Now, for any clause of the form $l_i \implies l_j$, place a directed edge from literal l_i to literal l_j.

Prove that a 2-CNF formula is satisfiable if and only if there is *no* variable x_i for which there is a path from x_i to $\overline{x_i}$ and a path from $\overline{x_i}$ to x_i. Use this to show that 2-SAT has a polynomial time algorithm.

(c) [**4**] Let us now consider a new problem, which we call *2-or-more 3-SAT*. Here, we are given a 3-CNF formula ϕ, and an assignment is said to "2-or-more satisfy" a clause iff *at least two* of the literals in that clause are TRUE for that assignment. For example, the clause $(x_1 \vee x_2 \vee \overline{x_3})$ is "2-or-more satisfied" by the assignment T, F, F, but not by the assignment T, F, T. The problem is now, given a formula, find if there exists an assignment that "2-or-more satisfies" all the clauses.

Prove that *2-or-more 3-SAT* has a polynomial time algorithm.

Question 2: Decision vs Search.. **[5]**

We mentioned in class that for purposes of complexity, it often helps to think of *decision* versions of problems. We now see a *justification* of this (such results are true for many other problems).

Consider the Independent-Set problem, in which the input is an undirected graph $G = (V, E)$ and a parameter k, and the goal is to determine if G has an independent set of size k. Suppose we have an oracle $\mathcal{O}$ for solving this decision version of independent set (think of it as a library function that takes input a graph G and k and answers YES/NO).

Prove that there exists an algorithm that can *find* an independent set of size k, if one exists, using a polynomial number of calls to the oracle $\mathcal{O}$, and possibly a polynomial amount of computation of its own.

Question 3: Reductions, reductions .... **[15]**

We will introduce some commonly arising problems in CS, and study their complexity. For all reductions, you need to reduce from one of the problems we have seen in class (SAT, 3-SAT, INDEPENDENT SET, VERTEX COVER, SUBSET SUM).

(a) [**4**] Consider the Integer Linear Programming (ILP) problem. We have n variables $x_1, x_2, \dots, x_n$. An instance of ILP is defined by a set of *linear* constraints, i.e., constraints of the form $a_1x_1 + a_2x_2 + \cdots + a_nx_n \le b$, where a_i and b are integers. The problem is to determine if there exist **integers** x_i that satisfy all the constraints.

Prove that ILP is NP-hard *[Hint: give a reduction from SAT.]*

(b) [**1**] Do you think ILP is NP-complete? [You are not expected to *answer* this. Give two-three lines of thought.]

(c) [**2**] Next, we will study equations over integers *modulo 2*, i.e., where the variables take 0/1 values, and we consider addition and multiplication (mod 2). First, consider linear systems modulo 2.

Suppose we a collection of linear equations on n 0/1-variables x_i:

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m.
\end{aligned}$$

This set of linear equations is said to be *satisfiable* iff there is a 0/1 assignment to the variables x_i such that all the equalities are satisfied (mod 2). Let us call this problem LINEQ(mod2). Does LINEQ(mod2) have a polynomial time algorithm? [Again, answer YES/NO with a couple of lines justification.]

(d) [**8**] Now, let us study **quadratic** equations (mod 2). Here, we have n variables as above, but the equations now are quadratic. I.e., each equation has the form $\sum_{i,j} c_{ij}x_i x_j = b$, where c_{ij} and b are 0/1 (and addition/multiplication are mod 2). For instance, consider the equation $x_1x_2 + x_3^2 + x_3x_1 = 1$. This is satisfied by the assignment $(1,1,1)$ and $(1,1,0)$, but not by $(1,0,1)$.

The QUADEQ(mod2) problem is to determine, given quadratic equations as above, if there is a 0/1 assignment such that all the equations are satisfied. Prove that QUADEQ(mod 2) is NP-complete. *[Hint: for the hardness, reduce from 3-SAT. As an intermediate step, you may want to prove that determining satisfiability of "cubic" equations (mod 2) is NP-hard; you get partial credit for this.]*

Question 4: Graphs – definitions .... [**10**]

(a) [**3**] All vertices have degree ≥ 2. Show that G has a cycle. Argue from first principles.

(b) [**2**] Does there exist an undirected graph on 10 nodes with degrees $2, 3, 4, 4, 7, 1, 4, 5, 3, 2$ respectively?

(c) [**5**] Recall that a directed graph $G = (V, E)$ is said to be strongly connected if for every $u, v \in V$, there is a directed path from u to v and vice versa. Given such a graph G, give an algorithm that finds a cycle of odd length, or answers that such a cycle does not exist. Your algorithm should run in time $O(|E| + |V|)$.

Question 5: Weary traveler .... [**10**]

Consider the common conundrum of a frequent traveler: you want to travel from place A to place B sometime during the week. Suppose you are given departure and arrival times of every flight in the world, during that week. The sole goal of the traveler is to minimize the *total travel time*, i.e., flying time plus time spent at intermediate airports. Suppose that for every connection, there must be a 10 minute gap between the scheduled arrival into that airport and the scheduled departure from that airport (time taken to connect).

Suppose there are n airports total, and m flights, give a polynomial time algorithm to find the schedule with the smallest total travel time (as defined above). Give an analysis of the running time of your algorithm. *[Hint: construct an appropriate graph and run Dijkstra's algorithm.]*
[Source: Jeff Erickson's Exercises]