CS 125 Algorithms & Complexity — Fall 2016

Problem Set 7

Due: 11:59pm, Friday, October 28th

See homework submission instructions at `http://seas.harvard.edu/~cs125/fall16/schedule.htm`

Problem 1

(a) (5 points) Prove that Vertex Cover is NP-complete even when restricted to graphs in which every vertex has degree at most 3. **Hint:** Show how to reduce the degree of a vertex in a graph by 1 while increasing the size of the minimum vertex cover by 1 by introducing two new vertices and adding appropriate edges. Be sure to prove that the minimum vertex cover always increases by exactly 1 in your transformation.

(b) (5 points) Recall that, as discovered in lecture, the Maximum Matching problem on graphs can be solved in polynomial time (in class we proved it, using flow, for bipartite graphs, but it's also poly-time solvable for general graphs). That is, given a collection of "compatible" pairs among n individuals and a number k, we can decide whether there exist at least k disjoint compatible pairs. Show that the Maximum 3-Way Matching problem, where we instead want to select among compatible *triples*, is NP-complete. **Hint:** as a warm-up, first do a reduction from Vertex Cover on 3-*regular* graphs, i.e. ones where every vertex has degree 3.

Problem 2

Let co-NP $= \{L : \bar{L} \in \text{NP}\}$ be the class of languages whose complement is in NP.

(a) (4 points) Show that a language L is complete for NP iff $\bar{L}$ is complete for co-NP. (Here completeness is with respect to poly-time mapping reductions, aka Karp reductions.)

(b) (2 points) Show that if NP $\neq$ co-NP, then P $\neq$ NP.

(c) (4 points) Let Tautology$=\{\phi$: ϕ a boolean formula s.t. $\forall a, \phi(a) = 1\}$. Show that Tautology is co-NP-complete, even when restricted to 3-DNF formulas.

Problem 3

Each of the following problems is either in P or is co-NP-hard. Determine which is the case and prove your answers.

(a) $\text{ALL}_{\text{DFA}} = \{\langle M \rangle : M \text{ a DFA such that } L(M) = \Sigma^*\}$.

(b) $\mathrm{EMPTY}_{\mathrm{DFA}} = \{\langle M\rangle : M$ a DFA such that $L(M) = \emptyset\}$.

(c) $\mathrm{ALL}_{\mathrm{NFA}} = \{\langle M\rangle : M$ an NFA such that $L(M) = \Sigma^*\}$.

(d) $\mathrm{EMPTY}_{\mathrm{NFA}} = \{\langle M\rangle : M$ an NFA such that $L(M) = \emptyset\}$.

Problem 4

A *search problem* is a mapping S from strings ("instances") to sets of strings ("valid solutions"). An algorithm M *solves* a search problem S if for every input x such that $S(x) \neq \emptyset$, $M(x)$ outputs some solution in $S(x)$. An NP *search problem* is a search problem S such that there exists a polynomial p and a polynomial-time algorithm V such that for every x, y:

- $y \in S(x) \Rightarrow |y| \leq p(|x|)$ and
- $y \in S(x) \iff V$ accepts $\langle x, y\rangle$

An NP *optimization problem* is given by a polynomial-time computable *objective function* $\mathrm{Obj} : \Sigma^* \times \Sigma^* \to \mathbb{N}$, and $\mathrm{Obj}(x, y) = 0$ if $|y| > p(|x|)$ for some polynomial p. The problem is: given an input x, find y maximizing $\mathrm{Obj}(x, y)$. An example is the problem of finding the shortest tour in an instance of the TRAVELING SALESMAN PROBLEM.

Prove that the following are equivalent:

- P = NP
- Every NP search problem can be solved in polynomial time.
- Every NP optimization problem can be solved in polynomial time.