6.045: Automata, Computability and Complexity Spring 2019
Problem Set #8 **Due Date:** Wednesday, May 8, 11:59pm

Rules: Same as before!

1 Conondeterminism and Finite Automata (6 points)

In this question, we study what happens to finite automata when we replace *nondeterminism* with the (apparently) stronger power of *conondeterminism*. Namely, we define AFAs (*All-Paths* Finite Automata) analogously to NFAs, except a string is accepted if and only if *all* possible computation paths in the AFA lead to an accept state (instead of *some* computation path, as in NFAs).

For simplicity, we define an AFA $A = (Q, \Sigma, \delta, Q_0, F)$ analogously to NFAs, *except* we require that for every state $q \in Q$ and every $\sigma \in \Sigma$, $\delta(q, \sigma) \neq \emptyset$. That is, for every state and symbol there is at least one transition in the AFA. Also for simplicity, let's make $\delta : Q \times \Sigma \to 2^Q$; that is, there are no ε-transitions in an AFA. We say that an AFA $A = (Q, \Sigma, \delta, Q_0, F)$ *accepts* $w = w_1 \cdots w_n$ (where $w_i \in \Sigma$ for all i) if for *all* sequences $r_0, \ldots, r_n \in Q$ such that $r_0 \in Q_0$ and $r_{i+1} \in \delta(r_i, w_i)$, we also have $r_n \in F$. That is, **every** valid computation path in the AFA leads to a final state.

Prove or disprove: the class of languages recognized by AFAs equals the class of regular languages.

2 More Hamiltonicity (2 points)

We saw in lecture that

$$\text{HAMPATH} = \{(G, s, t) \mid G \text{ is a directed graph with a Hamiltonian path from } s \text{ to } t\}$$

is $\mathbf{NP}$-complete. A Hamiltonian cycle in a graph is a *cycle* which includes each vertex exactly once. Prove that

$$\text{HAMCYCLE} = \{G \mid G \text{ is a directed graph with a Hamiltonian cycle}\}.$$

is $\mathbf{NP}$-complete.

3 Traveling Salesperson Problem (2 points)

The Traveling Salesperson Problem is the problem of given a weighted directed graph G and an integer k, decide if G has a cycle of weight at most k which visits every vertex. That is

$$\text{TSP} = \{(G, k) \mid G \text{ is a weighted directed graph with a cycle of length at most } k \text{ containing every vertex of } G\}.$$

Prove that TSP is NP-complete.

4 Fun with coNP (5 points)

(a) **(4 points)** Let RESILIENT be the set of inputs of the form (G, k, d, s, t) such that no matter how you add k edges to G, the longest simple path from s to t in the resulting graph always has length less than d. Such graphs G are "resilient" to having long paths. Prove that RESILIENT is $\mathbf{coNP}$-complete.

(b) **(1 point)** Recall the language FACTORING from lecture. What would the consequences be for complexity theory, if we could show that FACTORING is $\mathbf{coNP}$-hard?

5 Minimum Weight Assignments (6 points)

Suppose ϕ is a boolean formula on n variables $x_1, \ldots, x_n$, and let $\alpha \in \{0,1\}^n$ be a variable assignment to $x_1, \ldots, x_n$. The *weight* $w(\alpha)$ is defined to be the number of 1's in α. In other words, the weight of α is the number of variables in α which are assigned to be true.

Let $SA(\phi) \subseteq \{0,1\}^n$ be the set of all variable assignments that satisfy ϕ, and let $m(\phi) := \min_{\alpha \in SA(\phi)} w(\alpha)$. Intuitively, $m(\phi)$ is the minimum weight over all satisfying assignments to ϕ.

(a) **(4 points)** Define

$$\text{LOW-WEIGHT-2SAT} = \{(\phi, k) \mid \phi \text{ is a satisfiable 2-cnf formula such that } m(\phi) \leq k\}.$$

Prove that LOW-WEIGHT-2SAT is $\mathbf{NP}$-complete.

(b) **(2 points)** Define

$$\text{MIN-WEIGHT-SAT} = \{(\phi, k) \mid \phi \text{ is a satisfiable boolean formula such that } m(\phi) = k\}.$$

Show that MIN-WEIGHT-SAT $\in \mathbf{P}^{\mathbf{NP}}$.

6 Minimum Formulas (3 points)

In this problem, we will see how $\mathbf{P} = \mathbf{NP}$ implies more than just efficient algorithms for $\mathbf{NP}$ problems: it also implies efficient algorithms for some problems that are *not known* to be in $\mathbf{NP}$!

Recall from lecture that two Boolean formulas are *equivalent* if they are defined over the same set of variables and they have the same output value on every variable assignment. Fix a binary encoding of formulas. A formula ϕ is a *minimal* formula if there is no formula ψ with a smaller encoding length that is equivalent to ϕ. Define

$$\text{MIN-FORMULA} = \{\phi \mid \phi \text{ is minimal}\}.$$

We noted in class that this problem is **not** known to be in $\mathbf{NP}$, in $\mathbf{coNP}$, or even in $\mathbf{P}^{NP}$; it seems to be even harder! Prove that if $\mathbf{P} = \mathbf{NP}$ then MIN-FORMULA is in $\mathbf{P}$.

7 NP-Hardness vs NP-Completeness (8 points)

This problem is meant to illustrate the differences between being in $\mathbf{NP}$, being $\mathbf{NP}$-hard, and being $\mathbf{NP}$-complete.

(a) **(2 points)** Show that A_{TM} is $\mathbf{NP}$-hard. Therefore, there is a language L which is $\mathbf{NP}$-hard but not in $\mathbf{NP}$.

(b) **(5 points)** Show that there exists a *decidable* language L which is $\mathbf{NP}$-hard but is (provably) not in $\mathbf{NP}$. Here is a suggested proof outline:

- Find a function $t(n)$ such that $\mathbf{NP} \subset \text{TIME}[t(n)]$ but $\mathbf{NP} \neq \text{TIME}[t(n)]$.
- Find a language L such that for every $L' \in \text{TIME}[t(n)]$, $L' \leq_p L$.
- Show that if $L \in \mathbf{NP}$, then $\text{TIME}[t(n)] = \mathbf{NP}$.

(c) **(1 point)** Show that there exists a language L which is in $\mathbf{NP}$ but which is not $\mathbf{NP}$-hard.