6.541/18.405 Problem Set 0

due on February 15, 11:59pm

Problem 1 (2 points)

Question

For a Turing machine M, let $\langle M \rangle$ denote the description of M in binary. Prove that the following language is undecidable:

$$L = \{\langle M \rangle \mid \text{for all } n \text{ and input strings } x \text{ of length } n,$$
$$M \text{ is a Turing machine that halts on } x \text{ in } 18n^3 + 405 \text{ steps}\}$$

Answer

I will reduce this to the halting problem.

Given $\bar{M}$, we must determine whether $\bar{M}$ halts on the empty tape.

Let M be the Turing machine which does the following. On input x, it first simulates $\bar{M}$ for $|x|$ steps. If $\bar{M}$ has halted, M then runs for $|x|^4$ more steps, and then halts. Otherwise, if $\bar{M}$ has not halted, M halts.

For contradiction, suppose L is decidable. Let D be a Turing machine that decides it.

Say that D accepts M. Then M halts within $18|x|^3 + 405$ steps on every input x. Noting that $18 \times 100^3 + 405 < 100^4$, this means M does not run for $|x|^4$ or more steps on any x of length greater than 100. Thus, it must be that for all $n > 100$, $\bar{M}$ does not halt within n steps. Thus $\bar{M}$ must not halt.

Conversely, say D does not accept M. Then there is some x such that M runs for more than $18|x|^3 + 405$ steps. This can only happen if $\bar{M}$ halts within $|x|$ steps. Thus $\bar{M}$ halts.

Thus, if L is decidable, so is the halting problem.

Problem 2 (2 points)

Question

Prove that if 3SAT $\in P$, then there is a polynomial-time algorithm for solving SEARCH-3SAT (as defined in lecture 1). That is, under the hypothesis, there is a polynomial-time algorithm that given any 3CNF formula ϕ, the algorithm outputs a satisfying assignment to ϕ (when one exists) in polynomial time.

Answer

Say 3SAT $\in P$, and let M be a Turing machine that solves 3SAT in polynomial time. The following polynomial time algorithm can then be used to solve sc Search-3Sat.

Let ϕ be a given 3CNF formula, with variables $x_1, \ldots, x_n$.

First, call M on ϕ. If it rejects, return that no solution exists to ϕ.

Next, if $n = 1$, evaluate ϕ on $x_1 = 1$. If it is true, return $x_1 = 1$; else, return $x_1 = 0$.

Now say $n > 1$. Construct the CNF formula $\phi_1$by setting x_1 to 1 in ϕ. Call M on ϕ_1. If it accepts, assign $x_1 = 1$, and set $\phi \leftarrow \phi_1$. Else, set $x_1 = 0$ and construct the formula ϕ_0 by substituting $x_1 = 0$ in ϕ. Then set $\phi \leftarrow \phi_0$. Now recurse on the formula ϕ_0 or ϕ_1 to obtain an assignment to the variables in ϕ_0 or ϕ_1, which are $x_2, \ldots x_m$. This full assignment to x_1, and $x_2, \ldots, x_m$, is a solution to ϕ. Return it.

Each recursive step calls M once, and solves for one variable. Thus there are a polynomial number of steps in the input size (linear, in fact). Thus this algorithm takes polynomial time to run.

Problem 3 (3 points)

Question

Recall $\mathbf{EXP} = \bigcup_{c \geq 1} \mathbf{TIME}(2^{n^c})$ and $\mathbf{NEXP} = \bigcup_{c \geq 1} \mathbf{NTIME}(2^{n^c})$.
Prove that if $\mathbf{P} = \mathbf{NP}$, then $\mathbf{EXP} = \mathbf{NEXP}$.

Answer

Say $\mathbf{P} = \mathbf{NP}$. Let L be any language in $\mathbf{NEXP}$ on the alphabet $\{0, 1\}$. Let L' be the language on $\{0, 1, a\}$ (where a is just some other symbol) given by

$$L' = \{x \cdot a^{2^{|x|}} \mid x \in L\}$$

That is, L' contains each string x in L followed by a long string of the symbol a repeated $2^{|x|}$ times.

Let N be a nondeterministic Turing machine on the binary alphabet which decides L in $O(2^{n^c})$ time for some c. We now construct a nondeterministic Turing machine M on the alphabet $\{0, 1, a\}$ to decide L'. On any input, M runs N on the binary characters in the input and treats input squares with the symbol a as blank. Then on input $y = x \cdot a^{2^{|x|}}$, M runs in $O(2^{|x|^c})$. Thus M runs in $O(|y|^c)$. Thus M is a nondeterministic polynomial decider for L'. By the assumption $\mathbf{P} = \mathbf{NP}$, this means there exists a deterministic polynomial-time Turing machine R which decides L'.

Finally, let Q be a Turing machine on the binary alphabet which, on input x, simulates R. If R tries to move past the last filled bit in x, Q simulates that R reads in the character a if the tape head is at position $\leq |x| + 2^{|x|}$, and blank otherwise. Then R runs in time polynomial in $|x|2^{|x|}$, so exponential in $|x|$. Thus R is a deterministic exponential time decider for L, so $L \in \mathbf{EXP}$.

Problem 4 (3 points)

Define **E** to be $\bigcup_{c \geq 1} \mathbf{TIME}(2^{cn})$. Prove that $\mathbf{E} \neq \mathbf{PSPACE}$.
(Hint: find some property of **PSPACE** *that doesn't hold of* **E**.*)*

Problem 5 (4 points, 2 for each sub-problem)

In this problem we will see an "amplification" argument, in which we prove that a weak-looking lower bound actually implies a stronger lower bound. (We love these things.)

(A) Prove that if $\mathbf{TIME}(n^{1+\epsilon}) = \mathbf{TIME}(n)$ for some $\epsilon > 0$, then $\mathbf{TIME}(n^c) = \mathbf{TIME}(n)$ for all constant $c > 1$.
(You may assume without proof that the function $f(n) := \lceil n^{1+\epsilon} \rceil$ is time constructible, for every $\epsilon > 0$.)

If you want a hint, see the footnote.[1]

(B) Use (A) and the statement of the time hierarchy theorem from class to prove that $\mathbf{TIME}(n)$ is a proper subset of $\mathbf{TIME}(n^{1+\epsilon})$ for *every* $\epsilon > 0$.

Totally optional food for thought: We know that for Turing machines, $\mathbf{TIME}(n)$ is a proper subset of $\mathbf{TIME}(n \cdot \log^2 n)$. (This follows from Corollary 9.11 in Prof Sipser's book, for example.) Can you improve this proper containment (reduce the $\log^2 n$ factor) by using an amplification argument?

[1] First try to show that the assumption implies $\mathbf{TIME}(n^{(1+\epsilon)^2}) = \mathbf{TIME}(n^{1+\epsilon}) = \mathbf{TIME}(n)$, then try to iterate the argument.