# 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.

To solve 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 $\phi$, the algorithm outputs a satisfying assignment to $\phi$ (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

SEARCH-3SAT.

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

First, call $M$ on $\phi$. If it rejects, return that no solution exists to $\phi$.

Next, if $n = 1$, evaluate $\phi$ 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 $\phi$. Call $M$ on $\phi_1$. If it accepts, assign $x_1 = 1$, and set $\phi' \leftarrow \phi_1$. Else, set $x_1 = 0$ and construct the formula $\phi_0$ by substituting $x_1 = 0$ in $\phi$. Then set $\phi' \leftarrow \phi_0$. Now recurse on the formula $\phi'$ to obtain an assignment to the variables in $\phi'$, which are $x_2, \ldots x_m$. This full assignment to $x_1$, and $x_2, \ldots, x_m$, is a solution to $\phi$. 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). Since **P** is closed under composition, 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 **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 $Q$ runs in $O$(the runtime of $R$), which is polynomial in $|x| + 2^{|x|}$ (the input size to $R$), so exponential in $|x|$. Thus $Q$ is a deterministic exponential time decider for $L$, so $L \in \mathbf{EXP}$. Since $L$ was arbitrary, this means $\mathbf{NEXP} = \mathbf{EXP}$.

# Problem 4 (3 points)

## Question

Define $\mathbf{E}$ 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* $\mathbf{E}$*.)*

## Answer

**PSPACE** is closed under composition, whereas $\mathbf{E}$ is not.

To prove that $\mathbf{E}$ is not closed under composition, we perform diagonalization. Consider the language

$$L = \{\langle M \rangle \cdot 1^k \mid M \text{ is a Turing machine which does not accept on input } \langle M \rangle \cdot 1^k \text{ within } k2^{c(|M|+k)} \text{ steps for any } c \leq k\}$$

Say $L \in \mathbf{E}$. Then there exists a Turing machine $T$ which decides it in $O(2^{nc})$ time. Say $T$ accepts $\langle T \rangle \cdot 1^k$ for some $k \geq c$ such that $T$ runs in less than $k2^{nc}$ time on all inputs of length $n > k$. But then because $T$ runs within $k2^{cn}$ steps, the string $\langle T \rangle \cdot 1^k \notin L$! Thus it must be that $T$ rejects input $\langle T \rangle \cdot 1^k$ for all such $k$. But certainly on each of these inputs, $T$ does not accept within $2^{mn}$ steps for any $m \leq k$ (as $T$ does not accept at all). But this means that $\langle T \rangle \cdot 1^k \in L$. This is a contradiction. Thus, it must be that $L \notin \mathbf{E}$.

The last step to show $\mathbf{E}$ is not closed under composition is to show $L \in \mathbf{E}^{\mathbf{E}}$. Let

$$\bar{L} := \{\langle M \rangle \cdot 1^k \cdot 0 \cdot 1^n \mid M \text{ is a Turing machine which accepts on input } (\langle M \rangle \cdot 1^k) \text{ within } 2^n \text{ steps}\}$$

Certainly $\bar{L} \in \mathbf{E}$ since we can simply simulate $M$ for $2^n$ steps. And given an $\bar{L}$ oracle, we can decide $L$ by querying $\bar{L}$ on input $\langle M \rangle \cdot 1^k \cdot 0 \cdot 1^{c(|M|+k)}$ for each $c \leq k$. Thus $L \in \mathbf{E}^{\mathbf{E}}$. (In fact, this argument showed that $L \in \mathbf{P}^{\mathbf{E}}$. This is the issue with not allowing runtimes with a polyomial in the exponent, like runtime $2^{n^k}$, and only allowing $2^{cn}$: we can write down strings of polynomial size in the input length, but can't perform an exponential operation on them while staying within class $\mathbf{E}$.)

Finally, we must show that **PSPACE** is closed under composition. This follows for roughly the same reason $\mathbf{P}$ is closed under composition: if $A$ is a p-space Turing machine for language $X$, and $B$ is a p-space Turing machine for $Y$ which requires an $X$-oracle, we can decide $Y$ in p-space by running $B$, and calling $A$ whenever $B$ calls the $X$-oracle. (This only requires polynomial space because we can erase the memory used by $A$ each time the oracle is called. Anything from the computation of $A$ which $B$ needs to retain is something that $B$ would have had to write in its own memory anyway. Thus, the algorithm only needs at most the amount of memory used by $B$ plus that used by one call to $A$, which is the sum of two polynomials and hence polynomial.)

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

## Question

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* $\epsilon > 0$.

## Answer to A

Say $\mathbf{TIME}(n^{1+\epsilon}) = \mathbf{TIME}(n)$ for some $\epsilon > 0$.

Let $L \in \mathbf{TIME}(n^{(1+\epsilon)^2}) = \mathbf{TIME}(n^{1+2\epsilon+\epsilon^2})$.

Let

$$L' = \{x \cdot a^{|x|^{1+\epsilon}} \mid x \in L\}$$

Here, $L'$ is a language on the alphabet $\{0, 1, a\}$, where $a$ is some other symbol. ($L$ is a language on alphabt $\{0, 1\}$.)

For each $x \in L$, $y := x \cdot a^{|x|^{1+\epsilon}}$ can be decided as being in $L'$ in time $O(|x|^{(1+\epsilon)^2})$, by simply calling a Turing machine for $L$ on the bits in $y$ before the first $a$. Equivalently, $L$ can be decided as being in $L'$ in $O(|y|^{1+\epsilon})$. Thus $L' \in \mathbf{TIME}(n^{1+\epsilon})$. Because $\mathbf{TIME}(n^{1+\epsilon}) = \mathbf{TIME}(n)$, $L' \in \mathbf{TIME}(n)$.

Let $M'$ be a Turing machine for $L'$ that runs in $O(n)$. We can construct a Turing machine $M$ for $L$ which runs in time $O(n^{1+\epsilon})$ as follows. Given input $x$, $M$ simulates $M'$ on $x$. Whenever $M'$ moves the head on the input tape past the last character of $x$, $M$ simulates $M'$ observing symbol $a$ if the head position is less than $|x| + |x|^{1+\epsilon}$, and simulates it observing a blank square otherwise. This Turing machine runs in $O(\text{the runtime of } M')$, which is $O((|x|+|x|^{1+\epsilon})) = O(|x|^{1+\epsilon})$ (since $M'$ is linear time in its input). Thus, $M$ is $O(n^{1+\epsilon})$ so $L \in \mathbf{TIME}(n^{1+\epsilon})$. Since $\mathbf{TIME}(n^{1+\epsilon}) = \mathbf{TIME}(n)$, $L \in \mathbf{TIME}(n)$.

Thus $\mathbf{TIME}(n^{(1+\epsilon)^2}) = \mathbf{TIME}(n^{1+\epsilon}) = \mathbf{TIME}(n)$. Since $(1+\epsilon)^2 > 1+2\epsilon$, this means $\mathbf{TIME}(n^{1+2\epsilon}) = \mathbf{TIME}(n)$. Iterating this argument, now taking $\epsilon$ to be twice our original $\epsilon$, shows that $\mathbf{TIME}(n^{1+4\epsilon}) = \mathbf{TIME}(n)$. Repeating this lets us show $\mathbf{TIME}(n^{1+2^k\epsilon}) = \mathbf{TIME}(n)$ for all $k > 0$.

For any $c$, there exists a $k$ such that $1 + 2^k\epsilon > c$. Thus $\mathbf{TIME}(n^c) = \mathbf{TIME}(n)$ for all $c > 1$.

[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.

## Answer to B

The time hierarchy theorem states that there exists a $c > 0$ such that for any time constructable function $T(n)$, $\mathbf{TIME}(T(n)^c) \not\subseteq \mathbf{TIME}(T(n))$. Taking $T(n) = n$, this implies that $\mathbf{TIME}(n^c) \neq \mathbf{TIME}(n)$ for some $c$. By part A, this maens that in fact $\mathbf{TIME}(n^{1+\epsilon}) \neq \mathbf{TIME}(n)$ for all $\epsilon > 0$. And certainly $\mathbf{TIME}(n) \subseteq \mathbf{TIME}(n^{1+\epsilon})$, so we must have $\mathbf{TIME}(n) \subsetneq \mathbf{TIME}(n^{1+\epsilon})$.