The Time Hierarchy Theorem

Last Updated March 27th, 2024

1 Statement and Proof

In this lecture we use the `self` programming construct along with the Turing machine model of computation to prove the Time Hierarchy Theorem (see Chapter 9 of Sipser's Theory of Computation).

Definition 1.1. Let function $t : \mathcal{N} \to \mathcal{N}$ satisfy $t(n) \geq n \log n$ for all $n \geq 0$. Then t is said to be **time constructible** iff there is a transducer TM that, on input 1^n is able to output the binary representation of $t(n)$ in at most $\mathrm{O}(t(n))$ steps.

Note that the (log, polynomial, and exponential) functions that we tend to care most about are all time constructible (exercise!).

Theorem 1.2. (Time Hierarchy Theorem) If $t(n)$ is time constructible, then there is a decision problem that can decided in $\mathrm{O}(t(n))$ steps, but cannot be decided in $o(t(n)/\log t(n))$ steps.

Proof. Consider the decision problem L for which a positive instance is a pair $\langle M, w\rangle$, where M is the encoding of a deterministic Turing machine, $w \in \Sigma^*$ is an input word for M, and M accepts w within $t(n)/\log t(n)$ steps. Then L can be decided in $\mathrm{O}(t(n))$ steps via a TM $\hat{M}$ which, on input $\langle M, w\rangle$ simulates the computation of $M(w)$ for $t(n)/\log t(n)$ steps. It does this by interleaving the work tape with

1. cells $c_0, c_1, c_2, \ldots$ that represent the actual computation $M(w)$ along with
2. $\mathrm{O}(\log t(n))$-lengthed blocks of cells $B_0, B_1, B_2, \ldots$ each of which stores a copy of M along with a binary number of length $\lfloor \log t(n) \rfloor$ that serves as a "timer" for keeping track of the remaining number of steps to perform in the simulation.

This interleaving process is necessary, since $\hat{M}$ can only afford a simulation overhead of $\mathrm{O}(\log t(n))$ steps per simulation step and thus must always be within a constant number of steps from M and

$\boxed{C_0}\ \boxed{B_0}\ \boxed{C_1}\ \boxed{B_1}\ \ldots$

the binary "timer". Therefore $\hat{M}$ can decide instance $\langle M, w\rangle$ in

$$\mathrm{O}(\log(t(n)) \cdot t(n)/\log(t(n))) = \mathrm{O}(t(n))$$

steps as desired. For the sake of concreteness, let $c_1 > 0$ be a constant so that $\hat{M}$ runs in at most $c_1 \cdot t(n)$ number of steps, where $n = |\langle M, w\rangle|$.

$c_1 t(n)$

Now suppose L is *also* decidable by TM M' in $s(n) = o(t(n)/\log t(n))$ steps, meaning that

$$L = L(\hat{M}) = L(M').$$

Consider the following informal program for some TM Q.

> Input w.
>
> Simulate M' on input $\langle \texttt{self}, w\rangle$.
>
> Return $1 - M'(\langle \texttt{self}, w\rangle)$.

To finish the proof, we make the following points.

1. Unlike $\hat{M}$'s simulation of $M(w)$ which requires $\hat{M}$ to keep track of the number of simulation steps, Q does not care about the length of the simulation of $M'(\langle \texttt{self}, w\rangle)$, and follows it all the way to it completion. Because of this Q completes its simulation in no more than $c_2 \cdot s(n)$ steps, where $c_2 > 0$ is some constant and $n = |\langle \texttt{self}, w\rangle|$.

2. Since $s(n) = o(t(n)/\log t(n))$, by definition this means that

$$\lim_{n\to\infty} \frac{c_2 \cdot s(n)}{c_1 \cdot t(n)} = 0.$$

   Thus, for n sufficiently large, $c_2 \cdot s(n) < c_1 \cdot t(n)$.

3. Now let w be a word for which $n = |w|$ is so large that $c_2 \cdot s(n) < c_1 \cdot t(n)$. Then Q will complete its simulation of $M'(\langle \texttt{self}, w\rangle)$ in fewer than $c_1 \cdot t(n)$ steps, which means

$$\hat{M}(\langle Q, w\rangle) = Q(w) = 1 - M'(\langle Q, w\rangle),$$

   $M'(\langle Q,w\rangle) = \hat{M}(\langle Q,w\rangle)$

   where the second equality comes from the fact that the computation $Q(w)$ involves simulating $M'(\langle Q, w\rangle)$ and returning the opposite result of that simulation. Thus, $\hat{M}$ and M' return different values on input $\langle Q, w\rangle$, which contradicts the assumption that both machines accept the same language. Therefore, M' does not exist which proves the theorem. $\square$

2 The Complexity Class EXPTIME

Complexity class EXPTIME is defined as the set of all decision problems L that can be decided in $O(2^{p(|x|)})$ steps for each instance x of L, where p is some polynomial.

$2^{O(p(n))}$ for some polynomial $p(n)$

Corollary 2.1. Class P is a proper subset of EXPTIME.

Proof. Since $\log 2^n = n$ and $n^k = o(2^n/n)$ for all $k \geq 0$, it follows by the Time Hierarchy theorem that there is a decision problem L that requires $O(2^n/n)$ steps, yet cannot be decided in a polynomial number of steps. Therefore, $L \in \text{EXPTIME} - \text{P}$. □

3 The Space Hierarchy Theorem

We now provide an analogous space hierarchy theorem whose proof is left as an exercise.

Definition 3.1. Let function $s : \mathcal{N} \to \mathcal{N}$ satisfy $s(n) = \Omega(\log n)$. Then s is said to be **space constructible** iff there is a transducer TM that, on input 1^n is able to output the binary representation of $s(n)$ using at most $O(s(n))$ space.

Note that the (log, polynomial, and exponential) functions that we tend to care most about are all space constructible (exercise!).

Theorem 3.2. (Space Hierarchy Theorem) If $s(n)$ is space constructible, then there is a decision problem that can be decided in $O(s(n))$ space, but cannot be decided in $o(s(n))$ space.