Notes on Hierarchy Theorems

These notes discuss the proofs of the time and space hierarchy theorems. It shows why one has to work with the rather counter-intuitive diagonal problem defined in Sipser's book, and it describes an alternative proof of the time hierarchy theorem.

When we say "Turing machine" we mean one-tape Turing machine if we discuss time. If we discuss space, then we mean a Turing machine with one read-only tape and one work tape. All the languages discussed in these notes are over the alphabet $\Sigma = \{0, 1\}$.

Definition 1 (Constructibility) *A function* $t : \mathbb{N} \rightarrow \mathbb{N}$ *is* time constructible *if there is a Turing machine* M *that given an input of length* n *runs in* $O(t(n))$ *time and writes* $t(n)$ *in binary on the tape. A function* $s : \mathbb{N} \rightarrow \mathbb{N}$ *is* space constructible *if there is a Turing machine* M *that given an input of length* n *uses* $O(s(n))$ *space and writes* $s(n)$ *on the (work) tape.*

Theorem 2 (Space Hierarchy Theorem) *If* $s(n) \geq \log n$ *is a space-constructible function then* $\mathbf{SPACE}(o(s(n)) \neq \mathbf{SPACE}(O(s(n))$.

Theorem 3 (Time Hierarchy Theorem) *If* $t(n)$ *is a time-constructible function such that* $t(n) \geq n$ *and* $n^2 = o(t(n))$, *then* $\mathbf{TIME}(o(t(n)) \neq \mathbf{TIME}(O(t(n) \log t(n)))$.

1 Proof of The Space Hierarchy Theorem

Let $s : \mathbb{N} \rightarrow \mathbb{N}$ be a space constructible function such that $s(n) \geq \log n$.

We want to define a language L that can be solved in space $O(s(n))$ but not in space $o(s(n))$. The basic idea would be to define the language

$$L_1 = \{(\langle M \rangle, w) : \ M \text{ rejects } (\langle M \rangle, w) \text{ using } \leq s(|\langle M \rangle, w|) \text{ space } \}$$

Then no machine M' can solve L_1 using less than $s(n)$ space, because otherwise we would get a contradiction on all computations of M' on input $(\langle M' \rangle, w)$. On the other hand, deciding L_1 amounts to simulating a machine that uses $\leq s(n)$ space and it looks like such a simulation can be performed using $O(s(n))$ space.

Unfortunately, there are some difficulties in devising an $O(s(n))$ space algorithm for L_1.

One problem is that the machine M is part of the input, and so on an input of length n the description of M could be, say, $n/2$ bits long. The description of M includes a description of M's tape alphabet, and so the tape alphabet of M might have a large (depending on n) number of symbols, like, say, $\sqrt{n}$. In a computation of M in which $s(n)$ entries of tape are used, the tape is actually holding $O(s(n) \cdot \log n)$ bits of information, and if we want to simulate such a computation using a fixed Turing machine M_{L_1} with a finite alphabet it seems unavoidable that M_{L_1} will use more than $O(s(n))$ cells of tape.

Another problem is that we want to *decide* the language L_1, that is, we want to design a machine M_{L_1} that uses as little space as possible and that, on every input $(\langle M \rangle, w)$, halts with the right answer. But if M_{L_1} just simulates M on input $(\langle M \rangle, w)$, then M_{L_1} will never halt if M loops on input $(\langle M \rangle, w)$.

We solve the first problem by changing slightly the definition of L_1. We solve the second problem by introducing a counter of the number of steps used in the simulation of M, and aborting the simulation if the number of steps is too large.

To solve the first problem, we recall a result that we discussed several lectures ago when talking about Turing machine.

Lemma 4 (Size of Alphabet Tape) *Let L be a language decided by a Turing machine $M = (Q, \{0,1\}, \Gamma, \delta, q_0, q_A, q_R)$. Then there is another Turing machine $M' = (Q', \{0,1\}, \Gamma', \delta', q_0', q_A', q_R')$ that also decides L and whose tape alphabet Γ' has only four elements. Furthermore, if M uses space $s(n)$ then M' uses space $O(s(n))$.*

The "furthermore" part is new, but it follows from the same proof. Now, instead of L_1, consider the following language.

$$\begin{aligned} L_2 \quad = \{(\langle M\rangle, w) : \quad & M \text{ rejects } (\langle M\rangle, w) \text{ using } \leq s(|\langle M\rangle, w|) \text{ space} \\ & \text{and } M\text{'s tape alphabet has size four } \} \end{aligned}$$

Lemma 5 *$L_2 \notin$ **SPACE**$(o(s(n)))$.*

PROOF: Suppose there is a Turing machine M' that solves L_2 using $o(s(n))$ space. Then, using Lemma 4, there is a Turing machine M'' with a tape alphabet of size four that solves L_2 using $o(s(n))$ space. For sufficiently long w, M'' uses less than $s(|\langle M''\rangle, w|)$ space on input $\langle M''\rangle, w$, and now we get a contradiction no matter whether we assume $(\langle M''\rangle, w) \in L_2$ or not. □

Lemma 6 *$L_2 \in$ **SPACE**$(O(s(n) + \log n)) =$ **SPACE**$(O(s(n)))$.*

PROOF: We are given in input a machine $M = (Q, \{0,1\}, \Gamma, \delta, q_0, q_A, q_R)$ with $|\Gamma| = 4$ and a string w, we let n be the total length of $\langle M\rangle, w$. Note that $|Q| \leq n$.

We first compute $s(n)$ and mark $s(n)$ cells of work tape. This can be done using $O(s(n))$ space because of the space-constructibility of $s()$. Then we compute the number $t = |Q| \cdot 4^{s(n)} \cdot n$, and we initialize a counter to the value t, written in binary on the work tape. The counter uses space that is separate from the $s(n)$ cells that we have marked before. Writing the number t requires $O(s(n) + \log n)$ bits, and t can be computed using space $O(s(n) + \log n)$. Also, t is an upper bound to the number of configurations of machine M given an input of length n, assuming that M uses $\leq s(n)$ space on that input.

Then we proceed to a simulation of the computation of M on input $\langle M\rangle, w$. We use the $s(n)$ marked position of work tape to store the work tape of M and another $\log |Q| = O(\log n)$ positions of work tape to store the current state of M. After each step of the simulation, we decrease the counter.

- If the counter ever reaches zero, then M has gone through more than t steps on an input of length n and using space $s(n)$, which means that it must have gone through the same configuration twice, and so it loops on that input. If M loops on input $\langle M\rangle, w$ then $(\langle M\rangle, w) \notin L_2$. Therefore, if the counter reaches zero, we reject.
- If the $s(n)$ cells allocated for the simulation of the tape of M are not sufficient, that is, if M uses more than $s(n)$ space on input $\langle M\rangle, w$ then $(\langle M\rangle, w) \notin L_2$, and so we reject.

- If the simulation ends within t steps, then if M accepts we reject, and if M rejects we accept.

Note that, in all possible cases, our algorithm terminates, decides L_2 correctly, and uses $O(s(n) + \log n) = O(s(n))$ space. $\Box$

2 Two Proofs of the Time Hierarchy Theorem

Let $t : \mathbb{N} \to \mathbb{N}$ be a time constructible function such that $t(n) \geq n$ and $n^2 = o(t(n)$.

We want to define a language L that can be solved in time $O(t(n) \log t(n))$ but not in time $o(t(n))$. Continuing with the intuition from last section, the first idea would be to define a language

$$L_3 = \{(\langle M \rangle, w) : \; M \text{ rejects } w \text{ in } \leq t(|\langle M \rangle, w|) \text{ time steps } \}$$

Then no machine M' can solve L_3 using less than $t(n)$ time, because otherwise we would get a contradiction on all computations of M' on input $(\langle M' \rangle, w)$. As before, the difficult part is to give a good algorithm for L_3. Doing a time-efficient simulation is even harder than doing a space-efficient simulation, because one-tape Turing machine are very hard to program in a time-efficient way. If the size of $\langle M \rangle$ is, say, $n/2$, it may take time more than $O(n)$ to do even one step of a simulation of M on a given input. However, at least the following result is easy to prove.

Lemma 7 *There is a universal Turing machine U that, on input $(\langle M \rangle, w)$, where M is a Turing machine with tape alphabet of size four, simulates the computation of $M(\langle M \rangle, w)$. Every step of the simulation takes time $O(|\langle M \rangle|^2)$.*

In other words, if $M(\langle M \rangle, w)$ loops then $U(\langle M \rangle, w)$ loops. If $M(\langle M \rangle, w)$ accepts (respectively, rejects) in t steps then $U(\langle M \rangle, w)$ accepts (respectively, rejects) in $O(|\langle M \rangle|^2 \cdot t)$ steps.

PROOF: We organize the tape into two tracks (see the proof theorem 9.10 in Sipser) and we put a copy of $\langle M \rangle$ and the initial state q_0 of M on the second track. The first track contains the initial input. At every step, we see what is the symbol being read on the first track, what is the state, and we search $\langle M \rangle$ for a description of the next state, the next symbol and the move. We update the current state on track two and the element on track one accordingly, we move the state and $\langle M \rangle$ one position left of right as required, and we proceed with the next step of the simulation $\Box$

Regarding the universality of machines with small tape alphabet we have.

Lemma 8 *Let L be a language decided in time $t(n)$ by a turing machine M. Then L is decided in time $O(t(n) + n^2)$ by a Turing machine M' whose tape alphabet has size four.*

We then change the definition of L_3 in order to force the description of the Turing machine M to be short.

$$\begin{aligned} L_4 \;\; = \;\; & \{(\langle M \rangle, w) : \; M \text{ rejects } w \text{ in } \leq t(|\langle M \rangle, w|) \text{ time steps,} \\ & |\langle M \rangle| \leq \sqrt{\log t(|\langle M \rangle, w|)} \text{ and } M\text{'s tape alphabet has size } 4\} \end{aligned}$$

Lemma 9 $L_4 \notin \mathbf{TIME}(o(t(n)))$.

PROOF: Suppose L_4 can be solved by a machine M' in $o(t(n))$ time. Then it can also be solved by a machine M'' with a tape alphabet of size 4 in $o(t(n)) + O(n^2) = o(t(n))$ time. Let k be length of the description of M''. Let w be a string of length 2^{k^2}. Then

$$|\langle M''\rangle| \leq \sqrt{\log t(|\langle M''\rangle, w|)}$$

and we get a contradiction when we reason about whether the string $(\langle M''\rangle, w)$ should be in L_4 or not. $\Box$

Lemma 10 $L_4 \in \mathbf{TIME}(O(t(n) \cdot \log t(n)))$.

PROOF: We are given in input a machine M and a string w; the total length of the input is n. If the length of the description of M is more than $\sqrt{\log t(n)}$, we reject. Otherwise, we proceed as follows.

We first organize the tape into three tracks, as described in the proof of Theorem 9.10 in Sipser. Two tracks are as in the proof of Lemma 7, and the third track stores a counter initialized at $t(n)$. After every step of the simulation, we decrease the counter, and then we move the counter so that it is always on track with the copy of $\langle M\rangle$ and with the state on track two. Updating and moving the counter adds an overhead of $O(\log t(n))$ time to each step of the simulation, in addition to the $O(|\langle M\rangle|^2 = O(\log t(n))$ time that it takes to find and simulate the correct transition.

If the counter reaches zero we reject; if the simulation ends before the counter reaches zero, then we accept if M rejects and we reject if M accepts. We need to simulate at most $t(n)$ steps, and so the total running time is $O(t(n) \log t(n))$. $\Box$

The proof of the time hierarchy theorem in Sipser's book is slightly different. Sipser defines the language

$$L_5 = \{(\langle M\rangle, w) : \ U \text{ rejects } (\langle M\rangle, w) \text{ in } \leq t(|\langle M\rangle, w|) \text{ time steps}\}$$

Where U is the machine of Lemma 7. Then we can make the following claims.

Lemma 11 $L_5 \not\in \mathbf{TIME}(o(t(n)))$.

PROOF: Suppose some machine M' solves L_5 in $o(t(n))$ time, then some machine M'' with an alphabet tape of size 4 solves L_5 also in $o(t(n))$ time, and U runs in $o(t(n))$ time given inputs of the form $\langle M''\rangle, w$ of length n.

Let w be large enough so that U runs in time $\leq t(n)$ given in input $(\langle M''\rangle, w)$ of length n. Then we reach a contradiction whether or not we assume that this input is in L_5 or not. $\Box$

Lemma 12 $L_5 \in \mathbf{TIME}(O(t(n) \log t(n)))$.

PROOF: This is similar to the proof of Lemma 10, except that it is simpler because we only need to simulate a particular, fixed Turing machine. We can use two tracks, one being the tape of U and the other storing a counter initialized to $t(n)$. The counter is decremented and moved every time a step of U is simulated. If the counter reaches zero we reject. If the simulation ends within $t(n)$ simulated steps then we reject if U accepts and we accept if U rejects. Each of the $\leq t(n)$ steps of the simulation takes time $O(\log t(n))$, and so the total running time is $(O(t(n) \log t(n)))$. $\Box$