Tutorial 12

Gidon Rosalki

2025-06-18

1 The Time Hierarchy Theorem

Definition 1.1 (Time-constructible)**.** *A function $t : \mathbb{N} \to \mathbb{N}$ where $t(n) = \Omega(n \log(n))$ is called **time-constructible** of the function that maps 1^n (n in unary form) to the binary representation of $t(n)$ is computable in time $O(t(n))$*

Note: All non linear polynomials with integer coefficients and non-negative leading coefficient are time-constructible, so are exponential functions such as 2^n.

Example 1. *Is the function n^2 time-constructible?*

Solution. Given n 1s written on the tape, we need to write n^2 1s. We will use the following algorithm:

1. Translate the input $1 \dots 1$ to the binary representation of n, twice
2. Multiply the two numbers together by "vertical multiplication"

Time complexity:

1. $n \log(n)$
2. $\log(n)^2$

In total, this is $O\left(n^2\right)$, so this function is in fact time-constructible. □

Reminder: $f(n) = o(g(n)) \implies \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ We also have the following time hierarchy theorem, which we state here without a proof:

Theorem 1 (Time Hierarchy)**.** *Let $t : \mathbb{N} \to \mathbb{N}$ be a time constructible function. Then there exists a language L, that is decidable in $O(t(n))$ time, but not decidable in $o\left(\frac{t(n)}{\log(t(n))}\right)$ time*

Corollary 1. *For any two real numbers $1 \leq \varepsilon_1 < \varepsilon_2$, we have $TIME(n^{\varepsilon_1}) \subset TIME(n^{\varepsilon_2})$*

Corollary 2. *$P \subset EXPTIME$*

Proof. For every k, it holds that $n^k = O(2^n)$, so $\text{TIME}\left(n^k\right) \subset \text{TIME}(2^n)$, and therefore $P \subset \text{TIME}(2^n)$. By the time hierarchy theorem, we know that

$$\text{TIME}(2^n) \subsetneq \text{TIME}(4^n) \subset \text{EXPTIME}$$

□

2 The Space Hierarchy Theorem

While with respect to time complexity simulation comes with a logarithmic overhead, the same is not true for space and so the analogous space hierarchy theorem is cleaner, and provides a tighter bound.

Definition 2.1 (Space-constructible)**.** *A function $s : \mathbb{N} \to \mathbb{N}$ where $s(n) = \Omega(n)$ is called **space-constructible** if the function that maps 1^n (n in unary form) to the binary representation of $s(n)$ is computable in space $O(s(n))$*

Theorem 2 (Space Hierarchy)**.** *Let $s : \mathbb{N} \to \mathbb{N}$ be a space-constructible function. Then there exists a language L, that is decidable in $O(s(n))$ space, but not decidable in $o(s(n))$ space*

Corollary 3. *$SPACE\left(n^k\right) \subsetneq SPACE\left(n^{k+1}\right)$ for all $k \geq 1$, and $PSPACE \subsetneq EXPSPACE$*

3 Padding

Theorem 3 (The following are equivalent). 1. *PSPACE = PTIME*

2. $\exists k > 1 : SPACE\left(n^k\right) \subseteq PTIME$

3. $SPACE\left(n\right) \subseteq PTIME$

Proof . It is clear that if PSPACE = PTIME, then SPACE $\left(n^k\right) \subseteq$ PSPACE = PTIME for all $k \geq 1$. Also, if SPACE $\left(n^k\right) \subseteq$ PTIME, for some $k \geq 1$, then

$$\text{SPACE}\left(n\right) \subseteq \text{SPACE}\left(n^k\right) \subseteq \text{PTIME}$$

It is left to prove that if SPACE $(n) \subseteq$ PTIME, then PSPACE = PTIME, and for this we use the padding argument. Assume that SPACE $(n) \subseteq$ PTIME, and consider a language $L \in PSPACE$. Let $f(n) = n^k$ be such that $L \in$ DSPACE $(f(n))$. Consider the language

$$L' = \left\{w\#1^m : w \in L \wedge m = |w|^k\right\}$$

It is not hard to see that $L' \in$ SPACE $(n) \subseteq$ PTIME. Indeed, $f(n) = n^k$ is space constructible, and so we can first compute $|w|^k$, as long that the space for the output does not exceed m cells (and if it does, we will reject), and then compare $|w|^k$ with m. If they are not equal, then we reject, and otherwise, if $m = |w|^k$, then we decide whether $w \in L$ using

$$O\left(f\left(|w|\right)\right) = O\left(|w|^k\right) = O\left(|w\#1^m|\right) = O\left(n\right)$$

space. Thus, $L' \in$ PTIME. It then follows that $L \in$ PTIME: On input w, we first compute $m = |w|^k$, in polynomial time in $|w|$, and then determine whether $w\#1^m \in L'$. The latter can be done in polynomial time in $|w\#1^m| = O\left(|w|^k\right)$, which is polynomial in $|w|$. Hence, $L \in$ PTIME, and we are done. □

Corollary 4. *For all $k \geq 1$, it holds that SPACE$\left(n^k\right) \neq$ PTIME*

Proof . By the previous lemma, SPACE $\left(n^k\right)$ = PTIME implies PSPACE = PTIME, which clearly implies that

$$\text{SPACE}\left(n^{k+1}\right) \subseteq \text{PSPACE} = \text{PTIME} = \text{SPACE}\left(n^k\right)$$

The latter is a contradiction to the separation

$$\text{SPACE}\left(n^k\right) \subsetneq \text{SPACE}\left(n^{k+1}\right)$$

given by the space hierarchy theorem □

4 Logarithmic space languages

Definition 4.1 (Logarithmic Space TM). *A logarithmic space TM is a TM with 3 tapes:*

1. *Input tape: Read only tape, containing letters only from Σ*

2. *Working tape: May write any $\sigma \in \Gamma$, and utilise $O\left(\log\left(n\right)\right)$ space*

3. *Output tape: This tape is write only, and may have any letter $\sigma \in \Sigma$ written to it, and it may only move right, or halt.*

We may now define the following languages:

$$L = \text{SPACE}\left(O\left(\log\left(n\right)\right)\right)$$
$$NL = \text{NSPACE}\left(O\left(\log\left(n\right)\right)\right)$$

Remember your laws of logarithms, $\log\left(n^k\right) = k\log\left(n\right)$. Thus, for every polynomial p, it is true that $\log\left(p\left(n\right)\right) = O\left(\log\left(n\right)\right)$. This is why we do not take powers of n in the log. However, note that $\log\left(n\right) = o\left(\log^2\left(n\right)\right)$. So a machine that works in $\log^2\left(n\right)$ does not belong to L.
We have seen that

$$\text{SPACE}\left(f\left(n\right)\right) \subseteq \text{TIME}\left(2^{O(f(n))}\right)$$

Thus, if $f = O\left(\log\left(n\right)\right)$, then

$$n \cdot 2^{O(f(n))} = n \cdot 2^{O(\log(n))} = n \cdot poly\left(n\right)$$

We can conclude that $L \subseteq P$, and next week we will show that $NL \subseteq P$

5 NL-Completeness

Similarly to the question of whether P = NP, we do not know whether L = NL. As we did in NP, we want to say on some problems in NL that they are the "hardest". Thus, we want to define the notion of NL-hardness. As we did in NP, we want to say that a problem is NL-hard if every other problem in NL is reducible to it. What kind of reductions should we use? This is a very important point, yet difficult to understand.

5.1 Log space reductions

Definition 5.1 (Log space reducible)**.** *We say that a language that A is log-space reducible to B if there exists a logarithmic space computable function $f : \Sigma^* \to \Sigma^*$ such that $\forall w,\ w \in A \Leftrightarrow f(w) \in B$. In this case we will say $A \leq_L B$.*

Definition 5.2 (NL-hard)**.** *We say that a language L is NL-hard if $\forall L' \in NL,\ L' \leq_L L$. If in addition, $L \in NL$, then we say that L is NL-complete*

Theorem 4 (Reduction theorem for LOGSPACE)**.** *If $B \in L$, and $A \leq_L B$, then $A \in L$. If $B \in NL$, and $A \leq_L B$, then $A \in NL$. Similarly, if $A \notin LNL$, and $A \leq_L B$, then $B \notin LNL$*

Corollary 5. *If $A \in$ NL-complete, and $A \in L$, then $L = NL$*

Theorem 5. *If $A \leq_L B$, and $B \leq_L C$, then $A \leq_L C$*

Proof . Left as an exercise, it is similar to the proof of log space reductions □

5.2 Relations between classes

So far we have seen the following relations between classes.

$$L \subseteq NL \subseteq P \subseteq NP \subseteq \text{PSPACE} = \text{NPSPACE} \subseteq EXP$$

From the hierarchy theorems, we thus get that

$$L \subsetneq \text{PSPACE},\ P \subsetneq \text{EXP}$$

And a non deterministic version of the hierarchy theorem shows that

$$NL \subsetneq \text{NPSPACE} = \text{PSPACE}$$

6 STRONGLY-CONNECTED is NL-Complete

A directed graph is strongly connected if every two nodes are connected by a directed path in each direction. We define the language

$$\text{STRONGLY-CONNECTED} = \{\langle G \rangle : G \text{ is a strongly connected graph}\}$$

and we claim that it is NL-complete. First, we will show that it is in NL, to do this we will use the Immerman theorem, which we will prove next week.

Theorem 6 (Immerman)**.** *$NL = coNL$*

We proved in the lecture that

$$\text{PATH} = \{\langle G, s, t \rangle : G \text{ is a directed graph, and there exists a path from } s \text{ to } t\}$$

is NL-complete. As such, $\overline{\text{PATH}}$ is NL-complete. So $\overline{\text{PATH}} \in NL$. Thus, there exists an NTM, that given $\langle G, s, t \rangle$, accepts **if and only if** there is no path from s to t, in G. Consider the following machine which recognises $\overline{\text{STRONGLY-CONNECTED}}$.
Given an input $\langle G \rangle$:

1. Non-deterministically select two nodes a, and b
2. Simulate the NTM that decides $\overline{\text{PATH}}$ on input $\langle G, a, b \rangle$. If it accepts, then the graph is not strongly connected, so accept. Otherwise, reject

Since storing node numbers a and b only takes logarithmic space, and $\overline{\text{PATH}}$ is recognised in logarithmic space, we get that $\overline{\text{STRONGLY-CONNECTED}} \in NL$. Again, from Immerman, we get that STRONGLY-CONNECTED $\in NL$.
Next, we show that every other language in NL is log-space reducible to STRONGLY-CONNECTED. We do this by reducing PATH (which is NL-complete) to STRONGLY-CONNECTED. Consider the following reduction:
On input $\langle G, s, t \rangle$, where $G = (V, E)$, the reduction outputs $\langle G' \rangle$, where G' is obtained from G by adding, for every $v \in V$, the edges (v, s) and (t, v).

We will claim that the reduction is correct, and that it can be done in log-space.

Correctness: We need to show that if there is a path from s to t in G, then G' is strongly connected. For $x, y \in V$ a path from x to y starts with the edge (x, s), and then takes the path from s to t, and then finally takes the edge (t, y).
Conversely, we need to show that if there is no path from s to t, then G' is not strongly connected. Indeed, we claim that t is not reachable from s in G', since a simple path from s to t in G' cannot use any of the new edges, and thus it is also a path in G. Since the only additional edges in the constructed graph go into s, and out of t, there can be no new ways of reaching t from s.

Space: We need to verify that the reduction can be performed by a log-space TM. A log-space TM can start by copying the input G, and for every vertex it writes, it also writes the two new edges. This is done vertex by vertex, and thus we always keep at most the encoding of a vertex, or an edge, on the tape. Both of these are logarithmic in the input. So the reduction TM will do the following:

1. Copy all of G onto the output tape
2. For each node i in G:
   (a) Add an edge from i to s
   (b) Add an edge from t to i

So, the space used is indeed logarithmic, although the output has size $O(n)$, essentially the only space necessary to perform the reduction is that used to keep track of the node number i in the above for loop.