Lecture 7 - Complexity

Gidon Rosalki

2025-05-11

If you love definitions, then you are going to enjoy this lecture. One can argue we have finished the "computation" part of the course, and we have now reached **complexity**.

1 Deterministic Turing Machines

1.1 Definitions

Definition 1.1 (Complexity)**.** *What we can compute in a limited number of computing resources*

Definition 1.2 (Computing resources)**.** *We will focus on* ***time****, and touch on memory*

Definition 1.3 (Runtime)**.** *Given a TM M, and an input w, the runtime of M on w is the length of the full run of M on w (the series of configurations beginning at the initial configuration), less 1 (due to the starting configuration not being a computation step)*

Definition 1.4 (Space complexity)**.** *The number of cells at which the head arrives during the run*

Definition 1.5 (Worst case runtime of M)**.** *Given a TM M, its runtime will be $T : \mathbb{N} \to \mathbb{N}$, such that*

$$T(n) = \max_{|w| \leq n} \{\textit{Runtime of } M \textit{ on } w\}$$

Definition 1.6 (Worst case space complexity of M)**.** *Given a TM M, its space complexity will be $S : \mathbb{N} \to \mathbb{N}$, such that*

$$S(n) = \max_{|w| \leq n} \{\textit{Space complexity of } M \textit{ on } w\}$$

Definition 1.7. *We will say that the TM M runs in time $O(f(n))$ if*

$$T(n) = O(f(n))$$

We may similarly define all of $\Omega, \omega, \Theta, \theta$ as we saw in DAST, and similarly for space complexity.

Until now, everything we have seen is essentially more formal definitions of what we saw in DAST. Gird thy loins, here be dragons.

Definition 1.8 (TIME)**.** *Let there be $f : \mathbb{N} \to \mathbb{N}$ be a monotonically increasing function, we will define*

$$TIME(O(f(n))) = \{L : \textit{There exists a TM that decides } L \textit{ and runs in } O(f(n))\}$$

Which is to say, the language needs be decidable, and runs in $O(f(n))$

Definition 1.9 (SPACE)**.** *Let there be $f : \mathbb{N} \to \mathbb{N}$ be a monotonically increasing function, we will define*

$$SPACE(O(f(n))) = \{L : \textit{There exists a TM that decides } L \textit{ with space complexity } O(f(n))\}$$

Which is to say, the language needs be decidable, with space complexity $O(f(n))$

1.2 Examples

Example 1 (INPUT-LENGTH)**.** *The TM that finds the length of a given input.*

Solution. We saw a method of putting a # at the end of the input, and a counter after it, and for each letter, we jump to the counter, and increase it, thus requiring $O(n^2)$.

We could also place the counter to the left of the input, and every time we read a letter from the input, we add 1 to the input, and move it right 1. Both these steps require $\log(n)$, and we carry them both out n times, resulting in $O(n \log(n))$.

As it transpires, we cannot do better than $O(\log(n))$, but it is very difficult to show this. Quite often we cannot show this, but overall it is complicated to show optimality.

With regards to space, if the counter overwrites the input, then we will not need more than $O(n)$ space □

Example 2 ($L \in REG$)**.** *Solution.* This takes $O(n)$, with space complexity of $O(n)$ as well □

Example 3 (PALINDROME)**.** *Solution.* We need to perform $\frac{n}{2}$ comparisons, and for each we move n steps to reach the equivalent letter, resulting in $\Theta\left(n^2\right)$ steps, with $O(n)$ space required.

If we consider a TM with **2** heads, then we can in fact perform this in $\Theta(n)$, since we start the heads at opposite ends of the word, and each step move the left starting head right, and the right starting head left, and perform the comparison. After n steps, both will reach a $\sqcup$, and the machine will return accept, assuming it hasn't found 2 different letters yet, whereupon it would have already rejected. □

1.3 More definitions

Definition 1.10 (P)**.**

$$P = PTIME = POLYNOMIAL\ TIME = \bigcup_{k\in\mathbb{N}} TIME\left(O\left(n^k\right)\right)$$

So something that takes $5n^3 + 7 + n^2 = O\left(n^3\right)$*, since it is contained within the family that takes* $O\left(n^3\right)$

Theorem 1 (Extended Church-Turing theorem)**.** *Every computer that can be built in our universe, and runs in* $O(f(n))$*, may be simulated on a TM, in time* $O\left(f(n)^k\right)$*, for some constant* k*. This means that* P *is not dependent on the computational model.*

Definition 1.11 (P)**.**

$$PSPACE = POLYNOMIAL\ SPACE = \bigcup_{k\in\mathbb{N}} SPACE\left(O\left(n^k\right)\right)$$

Definition 1.12 (E)**.**

$$E = \bigcup_{k\in\mathbb{N}} TIME\left(O\left(2^{k\cdot n}\right)\right) = TIME\left(O\left(\left(2^k\right)^n\right)\right)$$

We may continue on to EXP*, which is bigger than* E

$$EXPTIME = EXP = \bigcup_{k\in\mathbb{N}} TIME\left(O\left(2^{n^k}\right)\right)$$

The E stands for exponential

1.4 Examples

Example 4.

$$CONNECTED = \{\langle G\rangle : G\ is\ connected\}$$

Solution. As was discussed in DAST, $CONNECTED \in P$ □

Example 5.

$$MIN - SPANNING - TREE \in P$$

Solution.

$$MIN - SPANNING - TREE \in P = \{\langle G\rangle, \langle w\rangle, (k)_b : \text{There exists in } G \text{ a spanning tree whose weight of } w \leq k\}$$

□

Example 6.

$$SIMP - PATH = \{G, k : \textit{There exists a simple path of length } k \textit{ in } G\}$$

Solution.

$$SIMP - PATH \in E$$

We will pass over $\binom{m}{k}$ k-tuples of edges, and check if there is a path that we want $m = |E(G)|$. The runtime will be $\binom{m}{k} \cdot k^2 \leq \binom{n}{k} \leq 2^n \leq 2^{2n}$ □

2 Non deterministic TMs

Definition 2.1 (ND-TM)**.** *Non deterministic Turing machine:*

$$\delta(q,a) \in P(Q \times \Gamma \times \{R,L\})$$

Definition 2.2 (Partial run)**.** *A partial run is a series of configurations where the first is the initial configuration, and all the following configurations are successors of the previous ones.*

Definition 2.3 (Full run)**.** *A partial run that is either infinite, or that finishes in a final configuration.*

Definition 2.4 (Runtime of M on w)**.** *The length of the longest full run, or ∞ is there does not exist such a run.*

The equivalent definition for space is left as an exercise for Rachel, mobile number +972 58-613-5791 (probably, I have not saved her as a contact). Should a reader, who is not Rachel, fail to figure out the definition for themselves, I'm sure that she'll be more than happy to help you.

2.1 Non deterministic time / space complexity sets

Definition 2.5.

$$NTIME(O(f(n))) = \{L : \textit{There exists a ND-TM that } \textbf{\textit{recognises}} \textit{ } L \textit{ and runs in time } O(f(n))\}$$

Note, we haven't defined an ND-TM recognising a language, so let's do that now:

Definition 2.6. *An ND-TM accepts the input w if there exists a run that accepts w.*

Definition 2.7. *An ND-TM recognises the language L if it accepts the input w* ***if and only if*** *$w \in L$*

We defined P earlier, so let's extend that

Definition 2.8 (NP)**.**

$$NPTIME = NP = \bigcup_{k\in\mathbb{N}} NTIME\left(O\left(n^k\right)\right)$$

Definition 2.9.

$$NPSPACE = \bigcup_{k\in\mathbb{N}} NSPACE\left(O\left(n^k\right)\right)$$

Definition 2.10.

$$NE = \bigcup_{k\in\mathbb{N}} NTIME\left(O\left(2^{k\cdot n}\right)\right) = NTIME\left(O\left(\left(2^k\right)^n\right)\right)$$

Definition 2.11.

$$NEXPTIME = NEXP = \bigcup_{k\in\mathbb{N}} NTIME\left(O\left(2^{n^k}\right)\right)$$

Or in short, everything we defined for the deterministic case, can also be defined for the non deterministic case. It is trivial to show that $P \subseteq NP$, and it is left as an exercise to the reader to show that $NP \subseteq P$. Doing this will get you both 100% on the course (and also a million dollars, but really, that's insignificant in comparison).

2.2 Examples

Example 7.

$$HAMILTON - PATH = \{G : \textit{There exists a simple path in } G \textit{ of length } |V(G)| - 1\}$$

Solution. Let us consider $SIMP - PATH \in NP$. There exists a nondeterministic TM, that runs in polynomial time, and recognises the answer. We will have an encoding on the tape of $\langle G\rangle\langle k\rangle$ followed by k question marks. We will have an ND-TM, that goes over each of these question marks, and either replaces them with a 0 or a 1, and moves right each time. When it reaches a space (i.e., the end of the word), it checks if this collection of 0s and 1s is a simple path, and if so, returns accept, otherwise reject. This resolves in polynomial time, and can also solve $HAMILTON - PATH$, ans so there is the answer. □

Example 8 (CNF-SAT)**.** *CNF = Conjunctive Normal Form (not important)*
Input*: A boolean function over the polynomial variables $x_1, \ldots, x_n$, of the shape*

$$\psi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

Where C is a clause. For each C_i, it holds that

$$C_i = \left(X_1 \vee X_2 \vee \overline{X_4} \vee X_7\right)$$

An input is in the language if there exists a placement of variables that satisfies ψ

Solution. We want to show that $CNF - SAT \in E$. This holds because there are at most 2^n placements. To check each of these it takes much less than exponential time (trivial), and so overall we must check an exponential number of placements, resulting in $CNF - SAT \in E$. □

3 NP != P

It's time to use reductions again! Remember

$$x \in L' \Leftrightarrow Red\left(x\right) \in L$$

Definition 3.1. *A TM is a reduction machine from L' to L if it is a reduction from L' to L, and runs in polynomial time, by length of the input. In this case we will write*

$$L' \leq_p L$$

Theorem 2 (Polynomial reduction theorem)**.** *If $L \in P$ and $L' \leq_p L$, then $L' \in P$.*

Proof . We will not show a full proof, we have already seen similar proofs.
Let there be $x \in L'$ of length n. Then the reduction $Red\left(x\right)$, which runs in $O\left(n^{k_1}\right)$, is at most of length n^{k_1}, and $Red\left(x\right) \in L$, so we now run $M \circ Red\left(x\right)$. We know that M runs in $O\left(n^{k_2}\right)$, so running $M \circ Red\left(x\right)$, will take at most $O\left(n^{k_1 \cdot k_2}\right)$, and so $L \in P$ □

Definition 3.2. *A language L is C-hard (C is a set of languages) with respect to a polynomial reduction if for every $L' \in C$, it holds that $L' \leq_p L$. The language is C-complete if it is C-hard, and also $L \in C$.*

It transpires that both SIMP-PATH, and HAMILTON-PATH are both NP-complete.