CSE200: Complexity theory
Time and space hierarchies

Shachar Lovett

October 3, 2021

1 Time hierarchy

We next look more closely at the resources used by a TM. For convenience, we will assume that any string $x \in \{0,1\}^*$ corresponds to a description of some Turing machine $x = \langle M \rangle$. Say, if x doesn't compile, then M halts on the first step and returns 0. The important thing is that for every TM M there exists at least one description $\langle M \rangle$. It will also be convenient to assume that the description of TMs can be padded to arbitrary length. Thus formally, we assume:

- Every $x \in \{0,1\}^*$ corresponds to some TM M.
- For every TM M, there are infinitely many strings x which correspond to M.

Definition 1.1 (Time complexity)**.** *Let $T : \mathbb{N} \to \mathbb{N}$ be a time bound. We say a TM has time complexity T, if for any $n \in \mathbb{N}$ and any input $x \in \{0,1\}^n$, the TM halts after making at most $T(n)$ many steps. A function $f : \{0,1\}^* \to \{0,1\}^*$ is computed in time T if there is a TM computing it running in time T. We define*

$$TIME(T) = \{f : \{0,1\}^* \to \{0,1\}^* \text{ computable in time } cT \text{ for some } c \geq 1\}.$$

We will always assume that the function T is computable, otherwise we will run into funky situations. Moreover, we will assume it is computed in $\mathrm{TIME}(T)$ (i.e. it doesn't take longer to compute the bound than the function in the class). Some examples are:

- Polynomial time: $\mathrm{P} = \cup_{c \geq 1} \mathrm{TIME}(n^c)$.
- Single exponential time: $\mathrm{E} = \cup_{c \geq 1} \mathrm{TIME}(2^{cn})$.
- Exponential time: $\mathrm{EXP} = \cup_{c \geq 1} \mathrm{TIME}(2^{n^c})$.

Some examples of known algorithms and where they lie:

- Graph connectivity is in P. Given a graph on n vertices, we can run BFS to find whether two vertices are connected. This takes time $O(n^2)$, which is polynomial.

- Satisfiability is in E. Given a boolean formula on n variables, we can try all 2^n assignments to see if it is satisfiable.
- Factoring is in E. The best algorithms to factor an n-bit number run in time about $2^{n^{1/3}}$.

Claim 1.2 (Robustness of P)**.** *P is invariant to changing the TM model. For example, for the model variants we have discussed:*

- *Changing the alphabet replace time T with $O(T)$.*
- *Reducing the number of tapes increases the time from T to $O(T^2)$ (this can be improved to $O(T \log T)$).*
- *Replacing TM with RAM with standard TM increases time from T to $O(T^2)$ (this can be improved to $O(T \log T)$).*

We will consider from now on TMs with a single work tape and some fixed alphabet. The first question that we aim to tackle is whether giving TM more time make them more powerful. For example, is EXP strictly more powerful than P? The next theorem shows that this is indeed the case.

Before stating it, we recall asymptotic notation: given two functions $f, g : \mathbb{N} \to \mathbb{N}$, we say that f is asymptotically smaller than g, denoted $f = o(g)$, if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. We also say in this case that g is asymptotically larger than f, denoted $g = \omega(f)$.

Theorem 1.3 (Time Hierarchy Theorem)**.** *Let $T_1, T_2 : \mathbb{N} \to \mathbb{N}$ be time bounds with $T_1(n), T_2(n) \geq n$. Assume that $T_1^2 = o(T_2)$. Then*

$$\mathit{TIME}(T_1) \subsetneq \mathit{TIME}(T_2).$$

We note (without proof) that the condition can be strengthened to $T_1 \log(T_1) = o(T_2)$.

Proof. We define a function $f : \{0,1\}^* \to \{0,1\}$ such $f \in \mathrm{TIME}(T_2)$ but $f \notin \mathrm{TIME}(T_1)$. To define it, define $T : \mathbb{N} \to \mathbb{N}$ by $T(n) = \lfloor \sqrt{T_2(n)} \rfloor$ and note that $T_1 = o(T)$. On input x of length $n = |x|$, we define $f(x)$ as follows:

- Interpret x as a description of a TM $x = \langle M \rangle$.
- Simulate the run $M(\langle M \rangle)$ for at most $T(n)$ steps.
- If M halts and outputs 0 then set $f(x) = 1$. Otherwise set $f(x) = 0$.
- If M doesn't halt within $T(n)$ steps then set $f(x) = 0$.

We first argue that $f \notin \mathrm{TIME}(T_1)$. Assume it was. Then there exists a TM M for which $f(x) = M(x)$ for all $x \in \{0,1\}^*$ and such that M halts after at most $cT_1(|x|)$ steps for some constant c. As $T_1 = o(T)$, we have that $cT_1(n) \leq T(n)$ for any large enough n, namely $n \geq n_0$. We may assume that $|\langle M \rangle| \geq n_0$ by padding the TM description. We now do a case analysis:

- If $f(\langle M \rangle) = 0$ then $M(\langle M \rangle) = 1$. Note that $M(\langle M \rangle)$ halts in at most $T(|\langle M \rangle|)$ steps by our assumption.
- If $f(\langle M \rangle) = 1$ then $M(\langle M \rangle) = 0$.

Thus $\langle M \rangle \in \{0,1\}^*$ satisfies $f(\langle M \rangle) \neq M(\langle M \rangle)$, contradicting our assumption.

We next argue that $f \in \text{TIME}(T_2)$. To show this, we need to build a TM computing f which runs in time $O(T_2)$. Let U be a universal Turing machine with a step counter. That is, U takes inputs $\langle M \rangle, x$, and its work tape contains, in addition to the standard work memory needed to simulate M, also a counter which counts the number of steps.

Consider a simulation of running M on input x. First, we initialize the counter with $T(|x|)$. Then, to simulate a single step of M, U needs to

- Read the symbol under M's simulated reading head.
- Scan the input tape for the transition step corresponding to the current state.
- Update the current state.
- Decrease the counter.

When the counter reaches zero U halts. Similar to how decreasing the number of tapes increases the runtime quadratically. Concretely, to simulate a single step of M, if the tape of U has k symbols then it takes $O(k)$ steps for U to simulate M. Consider now simulating $M(\langle M \rangle)$ with $n = |\langle M \rangle|$. The length of the work tape of U is at most $O(T(n))$, so U halts after at most $O(T^2(n))$ steps. Hence $f \in \text{TIME}(T_2)$. □

Corollary 1.4. *$P \subsetneq EXP$.*

2 Space hierarchy

Similar to time, we can can measure the space (in other words, memory) that Turing machines use.

Definition 2.1 (Space complexity)**.** *Let $S : \mathbb{N} \to \mathbb{N}$ be a space bound. We say a TM has space complexity S, if for any $n \in \mathbb{N}$ and any input $x \in \{0,1\}^n$, the TM halts while using only at most the first $S(n)$ symbols of the work tape. A function $f : \{0,1\}^* \to \{0,1\}^*$ is computed in space S if there is a TM computing it running in space S. We define*

$$SPACE(S) = \{f : \{0,1\}^* \to \{0,1\}^* \text{ computable in space } cS \text{ for some } c \geq 1\}.$$

As before, we will always assume that the function S is computable, and moreover S is computable in SPACE(S). Some examples are:

- Polynomial space: $\text{PSPACE} = \cup_{c \geq 1} \text{SPACE}(n^c)$.
- Logarithmic space: $\text{LOGSPACE} = \text{SPACE}(\log n)$.

Some examples of well-known algorithms and where they lie:

- BFS or DFS use linear space, and hence are in in PSPACE.
- Satisfiability needs to store the assignment, which uses linear space, and hence is in PSPACE.
- Adding two numbers can be done in LOGSPACE. That is, adding two n-bit numbers, where the output is an $(n+1)$-bit number, can be done using $O(\log n)$ work memory. We would dig deeper into log-space later in the class to understand the intricacies involved.

Time complexity and space complexity are related, as the following two claims show.

Claim 2.2. *Let $T : \mathbb{N} \to \mathbb{N}$. Then TIME($T$) $\subset$ SPACE(T).*

Proof. In $T(n)$ steps you can move the head at most $T(n)$ times. □

Claim 2.3. *Let $S : \mathbb{N} \to \mathbb{N}$ and take $T(n) = n \cdot 2^{O(S(n))}$. Then SPACE($S$) $\subset$ TIME(T).*

Proof. Let M be a TM with states Q and tape alphabet Γ. The total number of configurations it has, if at most s cells of work tape has been used, is $Q \cdot n \cdot s \cdot |\Gamma|^s = O(n2^{O(s)})$ for $|Q|, |\Gamma| = O(1)$. If a TM terminates, it can reach each configuration at most once. Thus this puts a limit on its runtime of $O(n \cdot 2^{O(S(n))})$. □

Theorem 2.4 (Space Hierarchy Theorem)**.** *Let $S_1, S_2 : \mathbb{N} \to \mathbb{N}$ be space bounds where $S_1(n), S_2(n) \geq \log(n)$. Assume that $S_1 = o(S_2)$. Then*

$$SPACE(S_1) \subsetneq SPACE(S_2).$$

Proof. Let $f : \{0,1\}^* \to \{0,1\}$ be the following function. On input x of length $n = |x|$,

- Interpret x as a description of a TM $x = \langle M \rangle$.
- Simulate the run $M(\langle M \rangle)$ using at most $S_2(n)$ space in the work and output.
- If the reading head in either work or output takes moves beyond the first $S_2(n)$ cells, halt and output 0.
- If the TM makes more than $T_2(n) = 2^{S_2(n)}$ steps, halt and return 0.
- If M halts and outputs 0 then $f(x) = 1$. Otherwise $f(x) = 0$.

We first argue that $f \notin \text{SPACE}(S_1(n))$. Assume it was. Then there exists a TM M for which $f(x) = M(x)$ for all $x \in \{0,1\}^*$ and such that M runs in space $cS_1(n)$ and time $2^{cS_1(n)}$ and halts, for some constant $c \geq 1$. As in the Time Hierarchy Theorem, assume we have padded the description of the TM M such that, for $n = | < M > |$, $S_2(n) > cS_1(n)$ and $T_2(n) > 2^{cS_1(n)}$. So, the simulation lets M run its course. Then

- If $f(\langle M \rangle) = 0$ then $M(\langle M \rangle) = 1$.
- If $f(\langle M \rangle) = 1$ then $M(\langle M \rangle) = 0$.

We next argue that $f \in \text{SPACE}(S_2)$. To show this, we need to build a TM computing f which runs in space $O(S_2(n))$. Let U be a universal Turing machine with a protection against memory overflow. That is, U takes inputs $\langle M \rangle, x$; it first computes $S_2(|x|)$ and marks the first disallowed cell in the work and output tape with a special symbol. After that symbol it stores the current state of M. In addition, it has a counter for the number of steps it makes. It then simulates $M(x)$. It is easy to see that space that U uses is $O(S_2(n))$, and so $f \in \text{SPACE}(S_2)$. □

Corollary 2.5. $SPACE(\log n) \subsetneq SPACE(\log^2(n))$.