Time Hierarchy Theorem

Lecture 15

Binghui Peng

Announcement

- Homework 3 is due at April 12th, 11:59pm.
- Midterm grade stats can be found on Piazza; regrade due in one week; If your score does not meet your expectation, I encourage your to come to the my/TA's office hour.
- Final exam will be in-class, May 7 (Thursday), same format as midterm.

We will talk about time complexity

- Decidability asks whether a problem can be solved at all.

We will talk about time complexity

- Decidability asks whether a problem can be solved at all.
- Complexity theory asks a finer question: if a problem is solvable, how many resources does it need?

We will talk about time complexity

- Decidability asks whether a problem can be solved at all.
- Complexity theory asks a finer question: if a problem is solvable, how many resources does it need?
- We focus on **time complexity** for most of the course.

Running Time of a Turing Machine

- The runtime of a Turing machine M on input x is the number of steps M takes on input x (it equals to ∞ if loops forever);
- We use $T_M(n)$ to denote the maximum number of steps M takes on *any* input of length n.
- If M always halts and $T_M(n) \leq f(n)$ for all sufficiently large n, then we say that M runs in time $O(f(n))$.

The Class $\mathrm{TIME}(f(n))$

Definition: $\mathrm{TIME}(f(n))$ is the set of language decidable in $O(f(n))$ time, formally

$$\mathrm{TIME}(f(n)) = \{L : L \text{ is decided by some TM in time } O(f(n))\}.$$

Example: $\mathrm{TIME}(n^2)$ is the set of languages decidable in quadratic time.

A Natural Question

If we allow more time, do we really get more computational power?

In other words, is

$$\mathrm{TIME}(n) \subsetneq \mathrm{TIME}(n^2)$$

A Natural Question

If we allow more time, do we really get more computational power?

In other words, is

$$\mathrm{TIME}(n) \subsetneq \mathrm{TIME}(n^2)$$

true?

A Natural Question

If we allow more time, do we really get more computational power?

In other words, is

$$\mathrm{TIME}(n) \subsetneq \mathrm{TIME}(n^2)$$

true?

The time hierarchy theorem says **yes**.

Statement of the Time Hierarchy Theorem

Theorem Time Hierarchy Theorem states

$$\mathrm{TIME}(f(n)) \subsetneq \mathrm{TIME}(f(n) \log f(n))$$

for reasonable functions $f(n) \geq n$.

High level idea: Diagonalization Again

The proof uses the same high-level idea as the halting problem:

1. list all machines that run within time $f(n)$,
2. build a new machine D that disagrees with the ith machine on the ith relevant input,
3. make sure D still runs within the larger time bound.

Enumerating Fast Machines

Because Turing machines have finite descriptions, we can enumerate them:

$$M_1, M_2, M_3, \ldots$$

Among this list, many machines do not run in time $f(n)$, but the diagonal machine will simulate any candidate machine only for a bounded number of steps.

Construction

Fix a time bound $f(n)$. Define a machine D that on input x of length n does the following:

1. interpret x as the description of some TM machine,
2. simulate TM x on input x for at most $f(n)$ steps,
3. if x accepts within that time, then reject; otherwise accept.

Step 1. Runtime of D

The runtime of D on input of length n is at most $O(f(n) \log f(n))$.

Step 1. Runtime of D

The runtime of D on input of length n is at most $O(f(n) \log f(n))$.

Question: where does the extra $\log f(n)$ come from

Step 1. Runtime of D

The runtime of D on input of length n is at most $O(f(n) \log f(n))$.

Question: where does the extra $\log f(n)$ come from

Answer: The overhead comes from simulation.

That is why the theorem compares $\mathrm{TIME}(f(n))$ to a slightly larger class.

Step 2: D is not in $\mathrm{TIME}(f(n))$

Proof via contradiction: Suppse that D were decided by some machine M in time $O(f(n))$.

Step 2: D is not in $\mathrm{TIME}(f(n))$

Proof via contradiction: Suppse that D were decided by some machine M in time $O(f(n))$.

Now run D on M, the result should the same as run M on M.

Step 2: D is not in $\mathrm{TIME}(f(n))$

Proof via contradiction: Suppse that D were decided by some machine M in time $O(f(n))$.

Now run D on M, the result should the same as run M on M.

By construction, D must do the opposite of what M does on that input.

Contradiction. Therefore D is not in $\mathrm{TIME}(f(n))$.

Proof via code

```
def D(x):
  if x(x) halts in O(f(n)) steps and accept
    return False
  else:
    return True
```

Proof by contradiction: If M runs in $O(f(n))$ time and have the same input-output behaviour as D, take $x = M$ and then we have $M(M) = D(M)$, this can not happen according to the above definition.

Consequence: $\mathrm{P} \subsetneq \mathrm{EXP}$

Define

$$\mathrm{P} = \bigcup_{k \geq 1} \mathrm{TIME}(n^k),$$
$$\mathrm{EXP} = \bigcup_{k \geq 1} \mathrm{TIME}(2^{n^k}).$$

Using the time hierarchy theorem, one can prove

$$\mathrm{P} \subsetneq \mathrm{EXP}.$$

So not every decidable problem can be solved efficiently.

Takeaways

- Complexity classes compare problems by resource usage (beyond decidability)
- $\mathrm{TIME}(f(n))$ means decidable within time $O(f(n))$.
- Time hierarchy theorem states $\mathrm{TIME}(f(n)) \subsetneq \mathrm{TIME}(f(n)\log(f(n)))$