Lecture 15: Time Complexity
Computational Complexity Theory
Computational Complexity Theory

What can and can’t be computed with limited resources on computation, such as time, space, and so on.

Captures many of the significant issues in practical problem solving.

The field is rich with important open questions that no one has any idea how to begin answering!

We’ll start with: Time complexity
Let $f, g : \mathbb{N} \rightarrow \mathbb{N}$.
We say that $f(n) \leq O(g(n))$ if there are $c, n_0 \in \mathbb{N}$ so that for every integer $n \geq n_0$

$$f(n) \leq c \ g(n)$$

We say $g(n)$ is an upper bound on $f(n)$ if $f(n) \leq O(g(n))$

$$5n^3 + 2n^2 + 22n + 6 \leq O(n^3)$$

If $c = 6$ and $n_0 = 10$, then $5n^3 + 2n^2 + 22n + 6 \leq cn^3$
\[2n^{4.1} + 200283n^4 + 2 \leq O(n^{4.1})\]

\[3n \log_2 n + 5n \log_2 \log_2 n \leq O(n \log_2 n)\]

\[n \log_{10} n^{78} \leq O(n \log_{10} n)\]

\[\log_{10} n = \log_2 n \div \log_2 10\]

\[O(n \log_2 n) \leq O(n \log_{10} n) \leq O(n \log n)\]

Big-O isolates the “dominant” term of a function
A Simpler Big-O Definition

Let $f, g : \mathbb{N} \rightarrow \mathbb{N}$ We say $f(n) \leq O(g(n))$ if there is a $c \in \mathbb{N}$ so that for all $n \in \mathbb{N}$, $f(n) \leq c \cdot g(n) + c$
Measuring Time Complexity of a TM

We measure time complexity by counting the steps taken for a Turing machine to halt on an input.

Example: Let \( A = \{ 0^k1^k \mid k \geq 0 \} \)

Here’s a TM for \( A \). On input \( x \) of length \( n \):

1. Scan across the tape and reject if \( x \) is not of the form \( 0^i1^j \)

2. Repeat the following if both 0s and 1s remain on the tape:
   - Scan across the tape, crossing off a single 0 and a single 1
   - If 0s remain after all 1s have been crossed off, or vice-versa, reject. Otherwise accept.
Let $M$ be a TM that halts on all inputs.  
*(We will only consider decidable languages now!)*

**Definition:**
The **running time or time complexity of** $M$ **is the function** $T : \mathbb{N} \rightarrow \mathbb{N}$ **such that**

$$T(n) = \text{maximum number of steps taken by } M \text{ over all inputs of length } n$$
Time-Bounded Complexity Classes

Definition:
\[ \text{TIME}(t(n)) = \{ L' \mid \text{there is a Turing machine } M \]
with time complexity \( O(t(n)) \) so that \( L' = L(M) \} \]

\[ = \{ L' \mid L' \text{ is a language decided by a Turing machine with } \leq c \, t(n) + c \text{ running time, for some } c \geq 1 \} \]

We showed: \( A = \{ 0^k1^k \mid k \geq 0 \} \in \text{TIME}(n^2) \)

Is there a faster Turing machine?
$A = \{ 0^k1^k \mid k \geq 0 \} \in \text{TIME}(n \log n)$

$M(w) := \text{If } w \text{ is not of the form } 0^*1^*, \text{ reject.}$

Repeat until all bits of $w$ are crossed out:

If (parity of 0’s) $\neq$ (parity of 1’s), reject.

Cross out every other 0. Cross out every other 1.

Once all bits are crossed out, accept.

00000000000001111111111111
x0x0x0x0x0x0xx1x1x1x1x1x1x1x
xxx0xxx0xxx0xxxx1xxx1xxx1xxx1x
xxxxxxx0xxxxxxxxxxxxxxxx1xxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
\[ A = \{ 0^k1^k \mid k \geq 0 \} \in \text{TIME}(n \log n) \]

\[ M(w) := \text{If } w \text{ is not of the form } 0^*1^*, \text{ reject.} \]

Repeat until all bits of \( w \) are crossed out:

If (parity of 0’s) \( \neq \) (parity of 1’s), reject.

Cross out every other 0. Cross out every other 1.

Once all bits are crossed out, accept.

For a fixed \( w = 0^k1^k \):

Let \( zero_i \) be number of 0s left in \( w \), after iteration \( i \)

Let \( ones_i \) be number of 1s left in \( w \), after iteration \( i \)

Start with \( zero_0 = k, ones_0 = k \)

Key Observation:

\[ zero_{i+1} = \text{floor}(zero_i/2), ones_{i+1} = \text{floor}(ones_i/2) \]

Number of iterations \( \leq O(\log n) \)
It can be proved that there is no (one-tape) Turing Machine that can decide \( A \) in less than \( O(n \log n) \) time!

(Hard) Puzzle:

Let \( f(n) = O\left(\frac{n \log n}{\alpha(n)}\right) \) where \( \alpha(n) \) is unbounded.

Prove: \( \text{TIME}(f(n)) \) contains only regular languages(!)

For example, \( \text{TIME}(n \log \log n) \) contains only regular languages!
In particular, define the class
\[ \text{REGULAR} = \{L \mid L \text{ is a regular language}\} \]

**Proposition:** \( \text{REGULAR} \subset \text{TIME}(n) \)

Assuming the Puzzle is true, then for every \( f(n) = O\left(\frac{n \log n}{\alpha(n)}\right) \) where \( \alpha(n) \) is unbounded,

\[ \text{TIME}(f(n)) = \text{TIME}(n) \]

A “collapse” of complexity classes!
Two Tapes Can Be More Efficient

Theorem: $A = \{ 0^k1^k \mid k \geq 0 \}$ can be decided in $O(n)$ time with a two-tape TM.

Proof Idea:
Sweep over all 0s, copy them over on the second tape.
Sweep over all 1s. For each 1, cross off a 0 from the second tape.
Different models of computation can yield different running times for the same language!

Let’s revisit some of the key concepts from computability theory...
**Theorem:** Let \( t : \mathbb{N} \to \mathbb{N} \) satisfy \( t(n) \geq n \), for all \( n \).
Then every \( t(n) \) time multi-tape TM has an equivalent \( O(t(n)^2) \) time one-tape TM

Our simulation of multitape TMs by one-tape TMs achieves this!

**Corollary:** Suppose language \( A \) can be decided by a multi-tape TM in \( p(n) \) time, for some polynomial \( p \).
Then \( A \) can also be decided by a one-tape TM in \( q(n) \) time, for some polynomial \( q \).
Theorem: For every $t(n)$ time multi-tape TM, there is an equivalent $O(t(n)^2)$ time one-tape TM.
Theorem: For every $t(n)$ time multi-tape TM, there is an equivalent $O(t(n)^2)$ time one-tape TM.
Theorem: For every $t(n)$ time multi-tape TM, there is an equivalent $O(t(n)^2)$ time one-tape TM
Theorem: For every \( t(n) \) time multi-tape TM, there is an equivalent \( O(t(n)^2) \) time one-tape TM
An Efficient Universal TM

**Theorem:** There is a (one-tape) Turing machine $U$ which takes as input:
- the code of an arbitrary TM $M$
- an input string $w$
- and a string of $t$ 1s, $t > |w|$

such that $U(M, w, 1^t)$ halts in $O(|M|^2 t^2)$ steps and $U$ accepts $(M, w, 1^t) \iff M$ accepts $w$ in $t$ steps

The Universal TM with a Clock

**Idea:** Make a multi-tape TM $U'$ that does the above, and runs in $O(|M| t)$ steps
The Time Hierarchy Theorem

**Intuition:** If you get more time to compute, then you can solve strictly more problems.

**Theorem:** For all “reasonable” $f, g : \mathbb{N} \rightarrow \mathbb{N}$ where for all $n$, $g(n) > n^2 f(n)^2$, $\text{TIME}(f(n)) \subsetneq \text{TIME}(g(n))$

**Proof Idea:** Diagonalization with a clock
Make a TM $N$ that on input $M$, simulates the TM $M$ on input $M$ for $f(|M|)$ steps, then flips the answer.

We will show $L(N)$ cannot have time complexity $f(n)$
The Time Hierarchy Theorem

**Theorem:** For “reasonable” $f$, $g$ where $g(n) > n^2 f(n)^2$, \( \text{TIME}(f(n)) \subsetneq \text{TIME}(g(n)) \)

**Proof Sketch:** Define a TM $N$ as follows:

\[
N(M) = \text{Compute } t = f(|M|) \\
\text{Run } U(M, M, 1^t) \text{ and output the opposite answer.}
\]

**Claim:** $L(N)$ does not have time complexity $f(n)$.

**Proof:** Assume $N'$ runs in $f(n)$ time, and $L(N') = L(N)$.

By assumption, $N'(N')$ runs in $f(|N'|)$ time and outputs the *opposite* answer of $U(N', N', 1^{f(|N'|)})$.

So $N'(N')$ accepts $\iff U(N', N', 1^{f(|N'|)})$ rejects

$\iff N'(N')$ rejects in $f(|N'|)$ steps \([U \text{ is universal}]\)

This is a contradiction!
The Time Hierarchy Theorem

Theorem: For "reasonable" $f$, $g$ where $g(n) > n^2 f(n)^2$, \( \text{TIme}(f(n)) \subsetneq \text{TIme}(g(n)) \)

Proof Sketch: Define a TM $N$ as follows:

$N(M) = \text{Compute } t = f(|M|)$

Run $U(M, M, 1^t)$ and output the opposite answer.

So, $L(N)$ does not have time complexity $f(n)$.

For what functions $g(n)$ will $N$ run in $O(g(n))$ time?

1. Compute $t = f(|M|)$ in $O(g(|M|))$ time ["reasonable"]
2. Run $U(M, M, 1^t)$ in $O(g(|M|))$ time

Recall: $U(M, w, 1^t)$ halts in $O(|M|^2 t^2)$ steps

So set $g(n)$ so that $g(|M|) > |M|^2 f(|M|)^2$ for all $n$. QED

Remark: Time hierarchy also holds for multitape TMs!
A Better Time Hierarchy Theorem

**Theorem:** For “reasonable” $f$, $g$ where $g(n) > f(n) \log^2 f(n)$,

$$\text{TIME}(f(n)) \subsetneq \text{TIME}(g(n))$$

**Corollary:** $\text{TIME}(n) \subsetneq \text{TIME}(n^2) \subsetneq \text{TIME}(n^3) \subsetneq \ldots$

There is an infinite hierarchy of increasingly more time-consuming problems

**Question:** Are there important everyday problems that are high up in this time hierarchy? A natural problem that needs exactly $n^{10}$ time?

THIS IS AN OPEN QUESTION!
The analogue of “decidability” in the world of complexity theory
The EXTENDED Church-Turing Thesis

Everyone’s Intuitive Notion of Efficient Algorithms = Polynomial-Time Turing Machines

A controversial (dead?) thesis!

Counterexamples include $n^{100}$ time algorithms, randomized algorithms, quantum algorithms, ...
Nondeterminism and NP

The analogue of “recognizability” in complexity theory