BU CS 332 – Theory of Computation

https://forms.gle/bZBpiC6L6P8foBbJA

Lecture 19:

- Time/Space Complexity
- Time/Space Hierarchies
- Complexity Class P

Reading:

Sipser Ch 7.1-2, 8.0, 9.1

Mark Bun

April 9, 2025

Time and space complexity

The basic questions

1. How do we measure complexity?

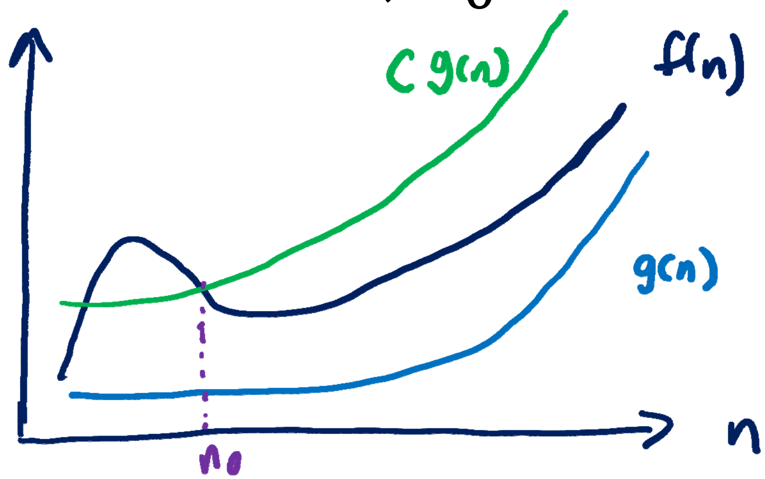
2. Asymptotic notation

3. How robust is the TM model when we care about measuring complexity?

4. How do we mathematically capture our intuitive notion of "efficient algorithms"?

Last Time: Asymptotic Notation

<u>Big-Oh:</u> $f(n) = O(g(n))$ if there exist c, n_0 such that $f(n) \le cg(n)$ for all $n \ge n_0$



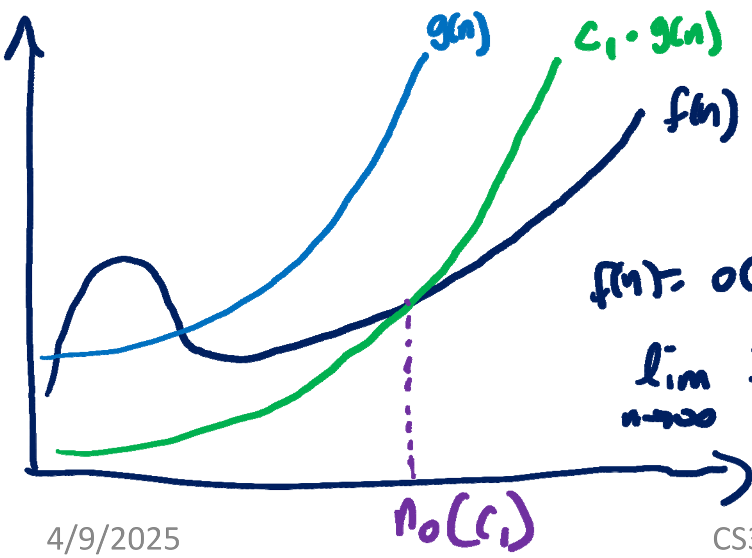
$100n^2 + n = O(n^2)$

but $100n^2 + n \ne o(n^2)$

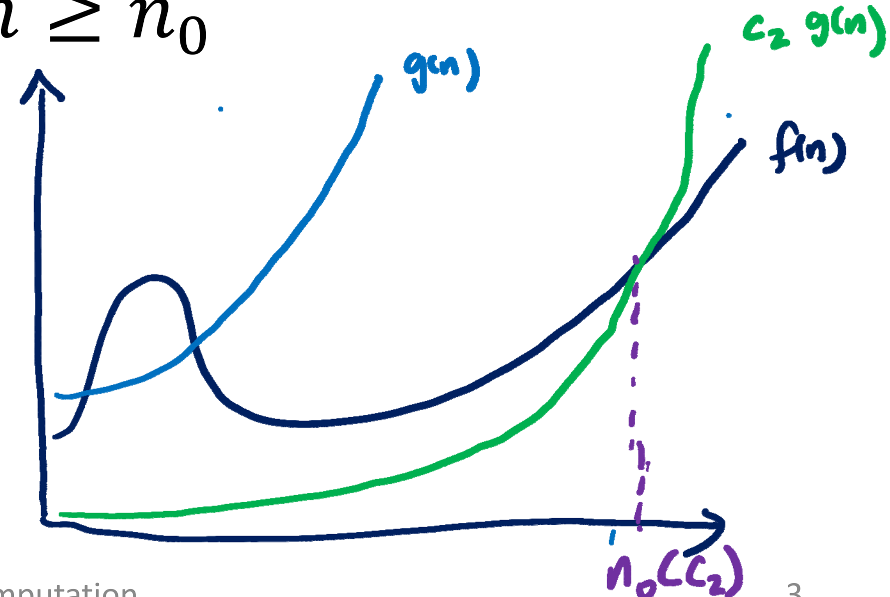
$= o(n^3) = o(n^2 \log n)$

<u>Little-Oh:</u> $f(n) = o(g(n))$ if for every c there exists n_0 such that $f(n) \le cg(n)$ for all $n \ge n_0$

$c > 0$



$f(n) = o(g(n)) \iff \lim_{n \to \infty} \frac{f(n)}{g(n)} \to 0$



Time complexity

Time complexity of a TM (algorithm) = maximum number of steps it takes on a worst-case input as a function of input length n

"runtime bound" input length n # steps

Formally: Let $f : \mathbb{N} \to \mathbb{N}$. A TM M runs in time $f(n)$ if for every n and every input $w \in \Sigma^n$, M halts on w within at most $f(n)$ steps

i.e. A is solvable in $O(f(n))$ time

A language $A \in \mathrm{TIME}(f(n))$ if there exists a basic single-tape (deterministic) TM M that

1) Decides A, and
2) Runs in time $O(f(n))$

Ex. $\mathrm{TIME}(n^2) =$ $\{ A \mid$ language A is decidable in quadratic time $\}$

Time class containment

If $f(n) = O(g(n))$, then which of the following statements is always true?

$\{A \mid \text{Language } A \text{ is decidable in } O(f(n)) \text{ time}\}$

a) $\boxed{\text{a)}}$ $\mathrm{TIME}(f(n)) \subseteq \mathrm{TIME}(g(n))$

b) $\mathrm{TIME}(g(n)) \subseteq \mathrm{TIME}(f(n))$

c) $\mathrm{TIME}(f(n)) = \mathrm{TIME}(g(n))$

d) None of the above

$A \in \mathrm{TIME}(f(n))$

$\Rightarrow \exists$ TM M deciding A in time $O(f(n))$

$\Rightarrow M$ decides A in time $O(g(n))$

$\Rightarrow A \in \mathrm{TIME}(g(n))$

Example

$\cancel{0}\,\cancel{0}\,\cancel{0}\,\cancel{0}\,\cancel{1}\,\cancel{1}\,\cancel{1}\,\cancel{1}$

$A = \{0^m 1^m \mid m \geq 0\}$

M = "On input w:

1. Scan input and reject if not of the form 0^*1^*] $O(n)$ time
2. While input contains both 0's and 1's:] $O(n)$ times through loop

   Cross off one 0 and one 1] $O(n)$

3. Accept if no 0's and no 1's left. Otherwise, reject."

- M runs in time $O(n^2)$ $\quad O(n) + O(n) \cdot O(n) = O(n^2)$

  $\Rightarrow \{0^m 1^m \mid m \geq 0\} \in \mathrm{TIME}(n^2)$

- Is there a faster algorithm?

Example

$\cancel{0}\cancel{0}\cancel{0}\cancel{0}\ \cancel{1}\cancel{1}\cancel{1}\cancel{1}$

$A = \{0^m 1^m \mid m \geq 0\}$

M' = "On input w:

1. Scan input and reject if not of the form 0^*1^* $O(n)$
2. $O(\log n) \rightarrow$ While input contains both 0's and 1's:
   $O(n)$
   - Reject if the total number of 0's and 1's remaining is odd
   - Cross off every other 0 and every other 1
3. Accept if no 0's and no 1's left. Otherwise, reject."

- Running time of M': $\underbrace{O(n)}_{\text{step 1}} + \underbrace{O(\log n)}_{\text{times through outer loop}} \cdot \underbrace{O(n)}_{\text{inner loop}}$

  $= O(n \log n)$

  $\Rightarrow A \in \text{TIME}(n \log n)$

- Is there a faster algorithm?

Example

Running time of M': $O(n \log n)$

Theorem (Sipser, Problem 7.49): If L can be decided in $o(n \log n)$ time on a basic single-tape TM, then L is regular

$A = \{0^m 1^m \mid m \geq 0\}$ is a non-regular language

$\Rightarrow$ $A \notin \mathrm{TIME}(f(n))$ for any $f(n) = o(n \log n)$

L is regular $\Rightarrow$ L decidable in time $O(n)$

Does it matter that we're using the 1-tape model for this result?

It matters: 2-tape TMs can decide A faster

tape 1: $\cancel{0}\cancel{0}\cancel{0}\cancel{0}$ 1111

tape 2: $\cancel{x}\cancel{x}\cancel{x}\cancel{x}$

M'' = "On input w:

$O(n)$ {
1. Scan input and reject if not of the form 0^*1^*
2. Copy 0's to tape 2

$O(n)$ {
3. Scan tape 1. For each 1 read, cross off a 0 on tape 2
4. If 0's on tape 2 finish at same time as 1's on tape 1, accept. Otherwise, reject."

Analysis: A is decided in time $O(n)$ on a 2-tape TM

Moral of the story (part 1): Unlike decidability, time complexity depends on the TM model

How *much* does the model matter?

Theorem: Let $t(n) \geq n$ be a function. Every multi-tape TM running in time $t(n)$ has an equivalent single-tape TM running in time $O(t(n)^2)$

Ex: If A decidable in time $O(n^7)$ on a 2-tape TM
also decidable in time $O(n^{14})$ on a single-tape TM

Proof idea:

We already saw how to simulate a multi-tape TM with a single-tape TM

Need a runtime analysis of this construction

Moral of the story (part 2): Time complexity doesn't depend too much on the TM model (as long as it's deterministic, sequential)

Single vs. Multi-Tape

Theorem: Let $t(n) \geq n$ be a function. Every multi-tape TM running in time $t(n)$ has an equivalent single-tape TM running in time $O(t(n)^2)$

Suppose B is decidable in time $O(n^2)$ on a 42-tape TM. What is the best upper bound you can give on the runtime of a basic single-tape TM deciding B?

a) $O(n^2)$

b) $O(n^4)$

c) $O(n^{84})$

d) $2^{O(n)}$

$t(n) = O(n^2)$

$O(t(n)^2) = O(n^4)$

Single vs. Multi-Tape

Theorem: Let $t(n) \geq n$ be a function. Every multi-tape TM running in time $t(n)$ has an equivalent single-tape TM running in time $O(t(n)^2)$

Proof idea:

We already saw how to simulate a multi-tape TM with a single-tape TM

Need a runtime analysis of this construction

Simulating Multiple Tapes

(Implementation-Level Description)

To simulate 1 step of M need O(S) steps on simulating machine where S = # of tape cells in use

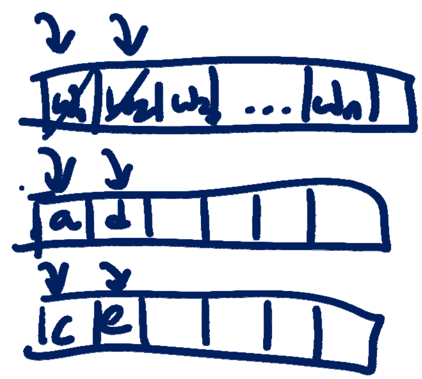
On input $w = w_1 w_2 \ldots w_n$

1. Format tape into $\# \overbrace{\dot{w}_1 w_2 \ldots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \ldots \#}^{S}$
2. For each move of $\boxed{M}$: original k-tape TM

    Scan left-to-right, finding current symbols

    Scan left-to-right, writing new symbols,

    Scan left-to-right, moving each tape head



If a tape head goes off the right end, insert blank

If a tape head goes off left end, move back right

Single vs. Multi-Tape

k tapes
k constant independent of n

Theorem: Let $t(n) \geq n$ be a function. Every multi-tape TM running in time $t(n)$ has an equivalent single-tape TM running in time $O(t(n)^2)$

Proof: Time analysis of simulation

- Time to initialize (i.e., format tape): $O(n+k)$
- Time to simulate one step of multi-tape TM: $O\big(k \cdot t(n)\big)$

  constant # of linear passes over $O(k \cdot t(n))$ cells of possible content
  because $t(n)$ time alg. can only touch $O(k \cdot t(n))$ cells

- Number of multi-tape steps to simulate: $t(n)$

$\Rightarrow$ Total time: $\underbrace{O(n+k)}_{\text{init.}} + \underbrace{t(n)}_{\text{steps of simulation}} \cdot \underbrace{O(k \cdot t(n))}_{\text{time per simulated step}} = O(k \cdot t(n)^2 + n + k) = O(t(n)^2)$

Extended Church-Turing Thesis

Every "reasonable" (physically realizable) model of computation can be simulated by a basic, single-tape TM with only a **polynomial** slowdown.

E.g., doubly infinite TMs, multi-tape TMs, RAM TMs

Does not include nondeterministic TMs (not reasonable)

Possible counterexamples? Randomized computation, parallel computation, DNA computing, quantum computation

Restrict ECT to deterministic, sequential models

might be true

Space complexity

Space complexity of a TM (algorithm) = maximum number of tape cells it uses on a worst-case input as a function of input length n

space bound, input length, worst-case # of cells

Formally: Let $f : \mathbb{N} \rightarrow \mathbb{N}$. A TM M runs in space $f(n)$ if for every n and every input $w \in \Sigma^n$, M halts on w using at most $f(n)$ tape cells

A language $A \in \mathrm{SPACE}(f(n))$ if there exists a basic single-tape (deterministic) TM M that

1) Decides A, and
2) Runs in space $O(f(n))$

How does space relate to time?

Which of the following is true for every function $f(n) \geq n$?

$TIME(f(n)) = \{ A \mid A \text{ is decidable in time } O(f(n)) \}$

$SPACE(f(n)) = \{ A \mid A \text{ is decidable by a TM using } \leq O(f(n)) \text{ tape cells} \}$

a) $\boxed{}$ $TIME(f(n)) \subseteq SPACE(f(n))$

b) $SPACE(f(n)) \subseteq TIME(f(n))$

c) $TIME(f(n)) = SPACE(f(n))$

d) None of the above

An algorithm touching $\geq f(n)$ cells must take $\geq f(n)$ time to touch those cells

$A \in TIME(f(n)) \Rightarrow$

$\exists$ TM M deciding A in time $O(f(n))$

$\Rightarrow M$ decides A in space $O(f(n))$

$\Rightarrow A \in SPACE(f(n))$.

Hopcroft-Paul-Valiant 75:

$TIME(f(n)) \subseteq SPACE(f(n)/\log f(n))$

Williams '24:

$TIME(f(n)) \subseteq SPACE(\sqrt{f(n) \log f(n)})$

Back to our example

$$A = \{0^m 1^m \mid m \geq 0\}$$

$\cancel{0}\ \cancel{0}\ \cancel{0}\ \cancel{0}\ \cancel{1}\ \cancel{1}\ \cancel{1}\ \cancel{1}$

M = "On input w:

1. Scan input and reject if not of the form 0^*1^*
2. While input contains both 0's and 1's:

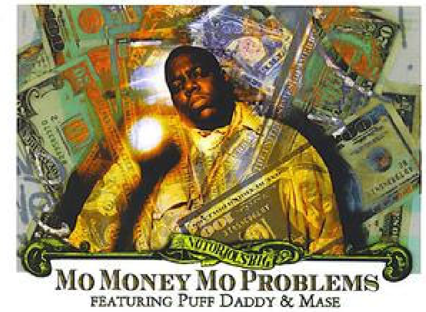
   Cross off one 0 and one 1
3. Accept if no 0's and no 1's left. Otherwise, reject."

$A \in SPACE(n)$

Theorem: Let $s(n) \geq n$ be a function. Every multi-tape TM running in space $s(n)$ has an equivalent single-tape TM running in space $O(s(n))$

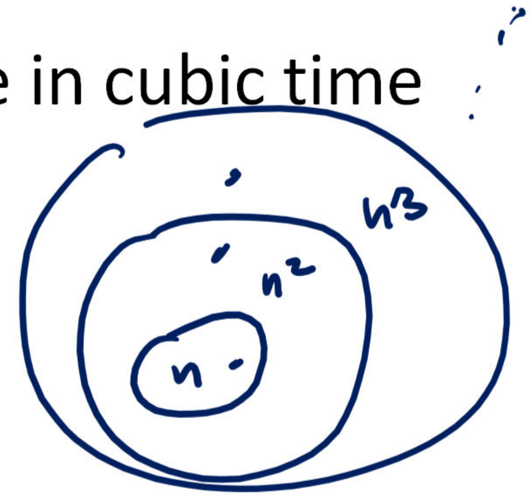
Hierarchy Theorems

More time, more problems



We know, e.g., that $TIME(n^2) \subseteq TIME(n^3)$

(Anything we can do in quadratic time we can do in cubic time)



Question: Are there problems that we can solve in cubic time that we cannot solve in quadratic time?

Theorem: There is a language $L \in TIME(n^3)$,

but $L \notin TIME(n^2)$

$TIME(n) \subsetneq TIME(n^2)$ means
1) $TIME(n) \subseteq TIME(n^2)$
2) $\exists\, L \in TIME(n^2)$ s.t. $L \notin TIME(n)$

"Time hierarchy":

$$TIME(n) \subsetneq TIME(n^2) \subsetneq TIME(n^3) \subsetneq TIME(n^4) \quad \ldots$$

"is a subset of, but not equal to"

Diagonalization redux

TM M	$M(\langle M_1 \rangle)$?	$M(\langle M_2 \rangle)$?	$M(\langle M_3 \rangle)$?	$M(\langle M_4 \rangle)$?		$D(\langle D \rangle)$?
M_1	~~Y~~ N	N	Y	Y	…	
M_2	N	~~N~~ Y	Y	Y		
M_3	Y	Y	Y	N		
M_4	N	N	Y	N		
⋮					⋱	
D						

$UD = \{\langle M \rangle \mid M$ is a TM that does not accept input $\langle M \rangle\}$

$L = \{\langle M \rangle \mid M$ is a TM that does not accept input $\langle M \rangle$ within $n^{2.5}$ steps$\}$

An explicit separating language

Theorem: $L = \{\langle M \rangle \mid M$ is a TM that does not accept input $\langle M \rangle$ within $n^{2.5}$ steps$\}$

is in $TIME(n^3)$, but not in $TIME(n^2)$

Proof Sketch: In $TIME(n^3)$ TM V

On input $\langle M \rangle$:

Takes $O(n^3)$ time

1. Simulate M on input $\langle M \rangle$ for $n^{2.5}$ steps
2. If M accepts, reject. If M rejects or did not yet halt, accept.

- M does not accept on $\langle M \rangle$ w/in $n^{2.5}$ steps $\Rightarrow$ V accepts
- M accepts on $\langle M \rangle$ w/in $n^{2.5}$ steps $\Rightarrow$ V rejects

An explicit separating language

Theorem: $L = \{\langle M \rangle \mid M$ is a TM that does **not** accept input $\langle M \rangle$ **within** $n^{2.5}$ **steps**$\}$

is in $TIME(n^3)$, but not in $TIME(n^2)$

Proof Sketch: <u>Not in $TIME(n^2)$</u>

Suppose for contradiction that $\boxed{D \text{ decides } L}$ in time $O(n^2)$

Two cases:

1) D accepts input $\langle D \rangle$ $\Rightarrow$ D accepts $\langle D \rangle$ w/in $O(n^2)$ steps $< n^{2.5}$

$\Rightarrow \langle D \rangle \notin L$ ✗

2) D rejects input $\langle D \rangle$ $\Rightarrow$ D does not accept $\langle D \rangle$ w/in $O(n^2)$ steps

$\Rightarrow \langle D \rangle \in L$

Time and space hierarchy theorems

- For every* function $t(n) \geq n \log n$, there exists a language decidable in $t(n)$ time, but not in $o\left(\frac{t(n)}{\log t(n)}\right)$ time.

Ex: $t(n) = n^2$

$\exists\ L \in TIME(n^2)$ but $L \notin TIME(f(n))$ for any $f(n) = o\left(\frac{n^2}{\log(n^2)}\right) = o\left(\frac{n^2}{\log n}\right)$

e.g. $L \notin TIME\left(\frac{n^2}{\log^2 n}\right)$

- For every* function $s(n) \geq \log n$, there exists a language decidable in $s(n)$ space, but not in $o(s(n))$ space.

Different measure of space complexity than space

*"time constructible" and "space constructible", respectively

Complexity Class $\mathbf{P}$

Time and space complexity

The basic questions

1. How do we measure complexity?

2. Asymptotic notation

3. How robust is the TM model when we care about measuring complexity?

4. How do we mathematically capture our intuitive notion of “efficient algorithms”?

Complexity class P

Definition: P is the class of languages decidable in polynomial time on a basic single-tape (deterministic) TM

$$\mathrm{P} = \bigcup_{k=1}^{\infty} \mathrm{TIME}(n^k) = \mathrm{TIME}(n) \cup \mathrm{TIME}(n^2) \cup \mathrm{TIME}(n^3) \ldots$$

- Class doesn't change if we substitute in another reasonable deterministic model (Extended Church-Turing)
- Cobham-Edmonds Thesis: Roughly captures class of problems that are feasible to solve on computers

A note about encodings

We'll still use the notation $\langle \quad \rangle$ for "any reasonable" encoding of the input to a TM…but now we have to be more careful about what we mean by "reasonable"

How long is the encoding of a V-vertex, E-edge graph…

… as an adjacency matrix?

… as an adjacency list?

How long is the encoding of a natural number k

… in binary?

… in decimal?

… in unary?

Describing and analyzing polynomial-time algorithms

- Due to Extended Church-Turing Thesis, we can still use high-level descriptions on multi-tape machines
- Polynomial-time is robust under composition: $\mathrm{poly}(n)$ executions of $\mathrm{poly}(n)$-time subroutines run on $\mathrm{poly}(n)$-size inputs gives an algorithm running in $\mathrm{poly}(n)$ time.
  $\Rightarrow$ Can freely use algorithms we've seen before as subroutines if we've analyzed their runtime
- Need to be careful about size of inputs! (Assume inputs represented in <u>binary</u> unless otherwise stated.)