BU CS 332 – Theory of Computation

https://forms.gle/tPzCFSW4dszu5dQB8

Lecture 19:

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

Reading:

Sipser Ch 9.1, 7.2

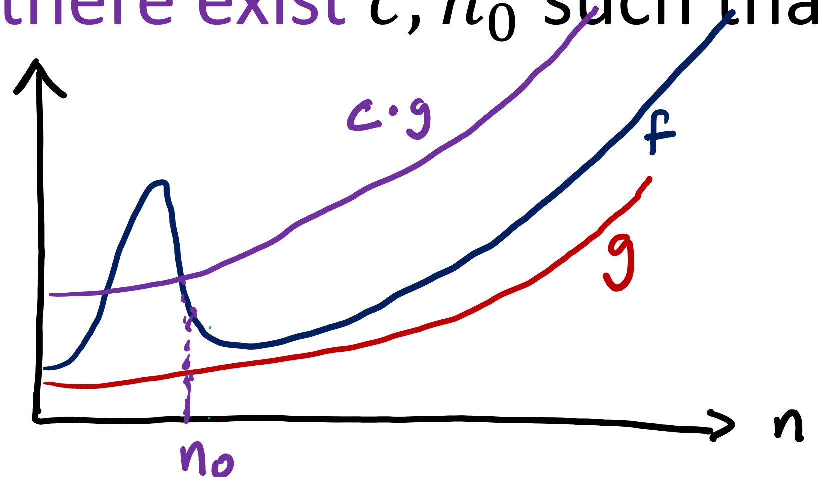
Mark Bun

April 10, 2024

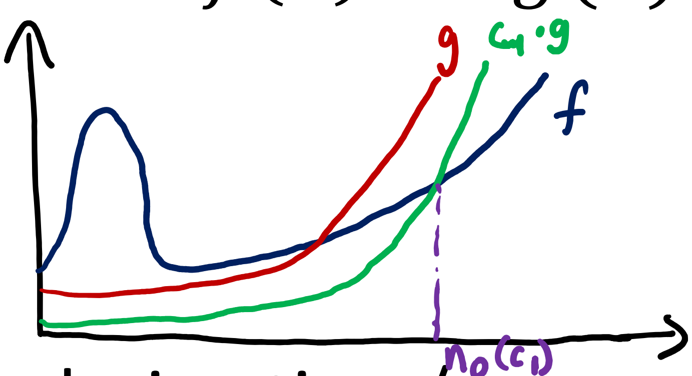
Last Time

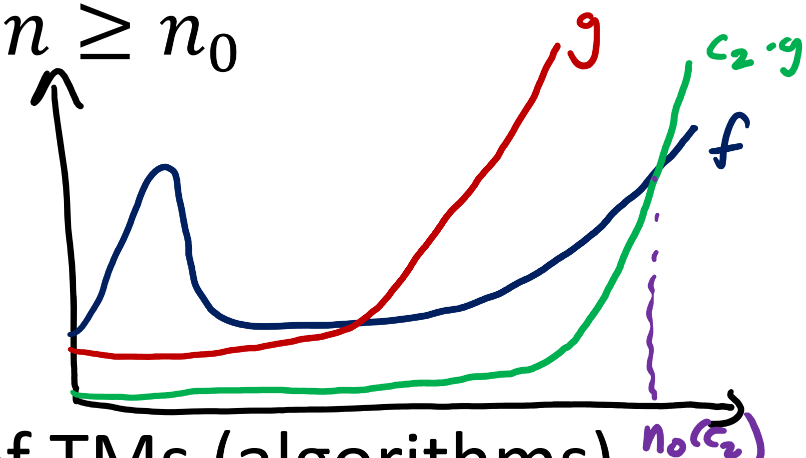
- Asymptotic notation

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



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





- Analyzing time/space usage of TMs (algorithms)

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

input length n runtime bound

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

A language $A \in \overbrace{\mathrm{TIME}(f(n))}^{\text{“class” or set of languages}}$ if there exists a basic single-tape (deterministic) TM M that

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

$A \in \mathrm{TIME}(n^2)$ if there exists any quadratic-time alg. solving A (e.g. running in time $10n^2$, $100n^2$, $10^6 n^2$, ...)

Time class containment

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

$\mathrm{TIME}(f(n)) = \{$ languages A s.t. A is decidable in time $O(f(n))$ $\}$

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

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

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

d) None of the above

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

$\exists$ TM M s.t. M decides A and M runs in time $O(f(n))$

$\Rightarrow \exists$ TM M s.t. M decides A and M runs in time $O(g(n))$

[since $f(n) = O(g(n))$]

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

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

Ex: $\underbrace{\mathrm{TIME}(n)}_{\text{class of problems solvable in linear time}} \subseteq \underbrace{\mathrm{TIME}(n^2)}_{\text{" " quadratic time}}$

Example

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

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:] $O(n)$

   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 $O(n^2)$ and decides language A

  $\Rightarrow A \in \mathrm{TIME}(n^2)$
- Is there a faster algorithm?

Example

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

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

M' = “On input w:

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

- Running time of M': $\underbrace{O(n)}_{\text{phase 1}} + \underbrace{O(\log n)}_{\text{times through phase 2 loop}} \cdot \underbrace{O(n)}_{\text{time per phase 2 procedure}} = O(n \log n)$

- Is there a faster algorithm?

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

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

Corollary: There is no $o(n \log n)$ time algorithm for $A = \{0^m 1^m \mid m \geq 0\}$ because A is nonregular

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

It matters: 2-tape TMs can decide A faster

00000 ~~11111~~

~~00000~~

M'' = "On input w:

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

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 $O(t(n)^2)$

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

B decidable in time $O(n^2)$ on 42-tape TM

$\Rightarrow$ B decidable in time $(O(n^2))^2$

Suppose B is decidable in time $O(n^2)$ on a <u>42-</u>tape TM. $= O(n^4)$

What is the best upper bound you can give on the runtime of a basic single-tape TM deciding B? on single-tape TM.

a) $O(n^2)$

b) $O(n^4)$

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

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

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)

If s cells in use at some point in computation

On input $w = w_1 w_2 \ldots w_n$

1. Format tape into $\underbrace{\# \dot{w}_1 w_2 \ldots w_n}_{\text{tape } 1} \underbrace{\# \dot{\sqcup} \#}_{\text{tape } 2} \dot{\sqcup} \# \ldots \underbrace{\#}_{\text{tape } k}$
2. For each move of M:

   Scan left-to-right, finding current symbols ← $O(s)$ steps

   Scan left-to-right, writing new symbols, ← $O(s)$ steps

   Scan left-to-right, moving each tape head ← $O(s)$ steps

   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

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

k is a constant (independent of n) = # of tapes of simulated multi-tape TM

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

Why?: If multi-tape TM runs in time $t(n)$, it touches $\leq k \cdot t(n)$ tape cells

Each simulation step takes time $O(\text{\# tape cells touched}) = O(k \cdot t(n))$

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

⇒ Total time: $\underbrace{O(n+k)}_{\text{initialization}} + \underbrace{O(t(n))}_{\text{\# steps to simulate}} \cdot \underbrace{O(k \cdot t(n))}_{\text{simulation time per multi tape step}} = O(n + k + k\,t(n)^2) = O(t(n)^2)$

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

More believable: ECT applies to deterministic, sequential models of computation

Space complexity

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

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 $f(n) \geq n$?

$TIME(f(n)) = \{ A \mid A$ decidable by a basic single tape TM in time $O(f(n))\}$

$SPACE(f(n)) = \{ A \mid \ldots$ space $\ldots \}$

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

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

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

$\Rightarrow \exists$ TM M decides A in space $O(f(n))$ [because M can touch at most $O(f(n))$ cells]

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

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

Thm [Hopcroft-Paul-Valiant '77]:

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

Back to our example

$A \in TIME(n \log n)$

$A \in SPACE(n)$

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

M = "On input w:

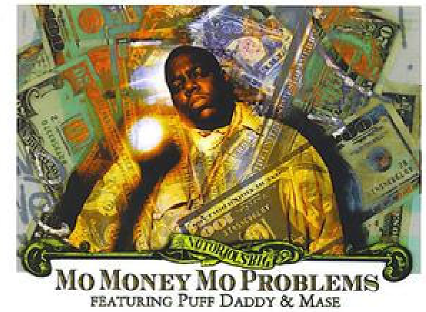
1. Scan input and reject if not of the form $0^* 1^*$
2. While input contains both 0's and 1's:

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

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 $L \notin TIME(n^2)$

$\exists L$ s.t. L decidable in cubic time

but L not decidable in quadratic time

"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"

$A \subsetneq B$ means $A \subseteq B$ but $\exists L \in B$ s.t. $L \notin A$.

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	N		
M_4	N	N	Y	~~N~~ Y		
$\vdots$					$\ddots$	
D						

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

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

An explicit separating language

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

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

Proof Sketch: <u>In $TIME(n^3)$</u> i.e. ∃ a cubic time alg. deciding L

On input $\langle M \rangle$: $n = |\langle M \rangle|$

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.

Trust me: This simulation can be done in $O(n^3)$ time

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: Not in $TIME(n^2)$

Suppose for contradiction that D decides L in time $O(n^2)$

Either:

1) D accepts $\langle D \rangle$ $\Rightarrow$ D accepts $\langle D \rangle$ w/in $O(n^2)$ steps
$\Rightarrow$ $D \notin L$ by defn. of L
contradicts correctness of D

2) D does not accept $\langle D \rangle$ $\Rightarrow$ D rejects $\langle D \rangle$ w/in $O(n^2)$ steps
$\Rightarrow$ $D \in L$ by def. of L
contradicts correctness of D.

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 s.t. L is decidable in time $\mathcal{O}(n^2)$ but not decidable in time $o\left(\frac{n^2}{\log(n^2)}\right) = o\left(\frac{n^2}{\log n}\right)$

← using different model of space accounting

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

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

- 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.)