BU CS 332 – Theory of Computation

https://forms.gle/zQ6NcWNc98FhDGnH9

Lecture 19:

- Time/Space Hierarchies
- Complexity Class P

Reading:

Sipser Ch 9.1, 7.2

Mark Bun

November 16, 2021

Last Time

- Asymptotic notation
- Analyzing time / space usage of Turing machines (algorithms)
- Multi-tape TMs can solve problems faster than single-tape TMs

  E.g., $A = \{0^m 1^m \mid m \geq 0\}$ can be decided in $O(n)$ time on a 2-tape TM, but cannot be decided in $o(n \log n)$ time on a basic TM

Time complexity

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

input length max running time on inputs of that length

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

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

1) Decides A, and

2) Runs in time $O(f(n))$

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 $O(t(n)^2)$

$t(n) = n^2$

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) $O(n^4)$ $\longrightarrow$ $O((n^2)^2) = O(n^4)$

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)

On input $w = w_1 w_2 \ldots w_n$

tape 1   tape 2   ...   tape k

1. Format tape into $\# \dot{w}_1 w_2 \ldots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \ldots \#$ $O(n+k)$ time
2. For each move of M:

   Scan left-to-right, finding current symbols

   Scan left-to-right, writing new symbols,

   Scan left-to-right, moving each tape head

   up to $k \cdot t(n)$ cells to scan

   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

- 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? In the worst case, each multi-tape tape may use $t(n)$ cells, (a TM running in time $t(n)$ touches up to $t(n)$ cells / tape)

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

$\Rightarrow$ Total time: $O(n+k) + \underbrace{t(n)}_{\#\text{ steps}} \cdot \underbrace{O(k \cdot t(n))}_{\text{time / step}} = O(t(n))^2$ (absorbing k into O)

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

Less controversial version of ECT:

All deterministic, sequential models can be simulated w/ poly. slowdown

Space complexity

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

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

space

How does space relate to time?

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

Want: If $A \in TIME(f(n))$, then $A \in SPACE(f(n))$

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

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

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

d) None of the above

False

Hopcroft-Paul-Valiant '77

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

If $A \in TIME(f(n))$, $\exists$ TM M running in time $O(f(n))$ deciding A.

M also decides A in space $O(f(n))$.

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

Back to our example

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

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."

M runs in space $O(n)$

$\Rightarrow A \in \mathrm{SPACE}(n)$

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 <u>cannot</u> solve in quadratic time?

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

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

"Time hierarchy":

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

TIME(n) is a subset of TIME(n^2), and TIME(n) $\neq$ TIME(n^2)

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						

$UD = \{\langle M \rangle \mid M$ 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$\}$ $n = |\langle M \rangle|$

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 $TIME(n^3)$

On input $\langle M \rangle$:

Simulation of TMs can be done with low time overhead (logarithmic)

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.

$O(n^{2.5} \log n)$ 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: <u>Not in $TIME(n^2)$</u>

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

<u>Case 1)</u> If $\langle D \rangle \in L$, then D does not accept $\langle D \rangle$ in time $n^{2.5}$ $\times$.

<u>Case 2)</u> If $\langle D \rangle \notin L$, then D accepts $\langle D \rangle$ in time $n^{2.5}$ $*$

Time and space hierarchy theorems

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

$\mathrm{TIME}(o(t(n)/\log t(n))) \subsetneq \mathrm{TIME}(t(n))$

Ex: $\mathrm{TIME}(n^2) \subsetneq \mathrm{TIME}(n^3)$, $\mathrm{TIME}(n^2/\log^2 n) \subsetneq \mathrm{TIME}(n^2)$

- For every* function $s(n) \geq \log n$, a language exists that is 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

Check your type checker: P

msb lsb carry

$x_1 \; x_2 \; \ldots \; x_n$

$y_1 \; y_2 \; \ldots \; y_n$

$(x_n + y_n) \bmod 2$

Consider the following computational problem: Given two numbers x, y (written in binary), output their sum $x + y$ (in binary). Which of the following is true?

There is a linear time 2-tape TM alg. (grade-school addition)

a) This is a problem in P

b) This problem is not in P because it cannot be solved by a Turing machine (i.e., it's undecidable)

c) This problem is not in P because it cannot be solved in polynomial time

d) This problem is not in P because it is not a decision problem (i.e., does not correspond to a language)

E.g. $\{\langle x, y, z\rangle \mid x + y = z\}$ is a decision problem / language

A note about encodings

We'll still use the notation ⟨ ⟩ 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? $O(|V|^2)$

… as an adjacency list? $O(|V| + |E|)$

Polynomial difference

How long is the encoding of a natural number k

… in binary? $\lceil \log_2 k \rceil$

… in decimal? $\lceil \log_{10} k \rceil$

constant factor difference

… in unary? k

exponential gap

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 binary unless otherwise stated.)