BU CS 332 – Theory of Computation

https://forms.gle/HWyXLDYAQp1UihtF6

Lecture 20:

- Complexity Class P
- Nondeterministic time, NP

Reading:

Sipser Ch 7.2, 7.3

Mark Bun

November 18, 2021

Complexity class P

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

$$\mathrm{P} = \bigcup_{k=1}^{\infty} \mathrm{TIME}(n^k) = \mathrm{TIME}(n) \cup \mathrm{TIME}(n^2) \cup \mathrm{TIME}(n^3) \cup \mathrm{TIME}(n^4) \ldots$$

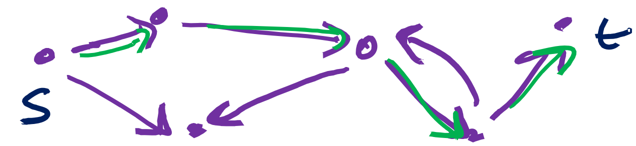
- Class doesn't change if we substitute in another reasonable deterministic model (Extended Church-Turing)
- Cobham-Edmonds Thesis: Roughly captures class of problems that are feasible to solve on computers

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: $\text{poly}(n)$ executions of $\text{poly}(n)$-time subroutines run on $\text{poly}(n)$-size inputs gives an algorithm running in $\text{poly}(n)$ 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.)

Examples of languages in P



$PATH =$
$\{\langle G, s, t\rangle \mid G$ is a directed graph with a directed path from s to $t\}$

Idea: Breadth-first search

Assume G is presented as adjacency matrix

"On input ⟨G, s, t⟩: Input length = $|V|^2 + 2\log|V|$

1. Mark start vertex s] $O(|V|)$
2. For $i = 1, 2, \ldots, |V|$:] Run for $|V|$ iterations
3. Traverse adj matrix of G to mark all neighbors of currently marked vertices] $O(|V|^2)$
4. If t is marked, accept. Otherwise, reject." $O(|V|)$

Runtime: $O(|V| \cdot |V|^2) + O(|V|) = O(|V|^3)$

Correctness: There is a path from s to t iff $\leq |V|$ iterations of BFS gets us to t from s.

Examples of languages in P

$E_{\mathrm{DFA}} = \{\langle D\rangle \mid D \text{ is a DFA that recognizes the empty language}\}$

Same alg. as before (can solve w/ BFS or DFS)

i.e. check if it is possible to reach an accept
state from start state.

Examples of languages in P

- $RELPRIME = \{\langle x, y \rangle \mid x \text{ and } y \text{ are relatively prime}\}$

i.e. given x, y, is it the case that $\gcd(x,y) = 1$?

Euclid's alg. solves in polynomial time

- $PRIMES = \{\langle x \rangle \mid x \text{ is prime}\}$



2006 Gödel Prize citation

The 2006 Gödel Prize for outstanding articles in theoretical computer science is awarded to Manindra Agrawal, Neeraj Kayal, and Nitin Saxena for their paper "PRIMES is in P."

In August 2002 one of the most ancient computational problems was finally solved....

A polynomial-time algorithm for $PRIMES$?

Consider the following algorithm for $PRIMES$

Encoded in binary

On input $\langle x \rangle$:

For $b = 2, 3, 4, 5, \dots, \sqrt{x}$:

- Try to divide x by b
- If b divides x, ~~accept~~ reject

If all b fail to divide x, ~~reject~~ accept

Input length n means that x could be as large as 2^n

$\Rightarrow$ # divisions could be as large as $\sqrt{2^n} = 2^{n/2}$

How many divisions does this algorithm require in terms of $n = |\langle x \rangle|$? a) $O(\sqrt{n})$ b) $O(n)$ c) $2^{O(\sqrt{n})}$ d) $2^{O(n)}$

Beyond polynomial time

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

$$\text{EXP} = \bigcup_{k=1}^{\infty} \text{TIME}(2^{n^k})$$

$= \text{TIME}(2^n) \cup \text{TIME}(2^{n^2}) \cup \text{TIME}(2^{n^3}) \cup \ldots$

$\text{TIME}(10^n) \not\subseteq \text{TIME}(2^n)$ [via time hierarchy]

$\text{TIME}(10^n) \subseteq \text{TIME}(2^{n^2})$

$= \text{TIME}(2^{n \log 10})$

Why study P ?

Criticism of the Cobham-Edmonds Thesis:

- Algorithms running in time n^{100} aren't really efficient

  Response: Runtimes often improve with more research

- Does not capture some physically realizable models using randomness, quantum mechanics

  Response: Randomness may not change P, useful principles

  If "hard functions" exist, then every poly-time rand. algorithm can be "derandomized"

$TIME(n)$ vs. $TIME(n^2)$

P vs. EXP

decidable vs. undecidable

Nondeterministic Time and NP

Nondeterministic time

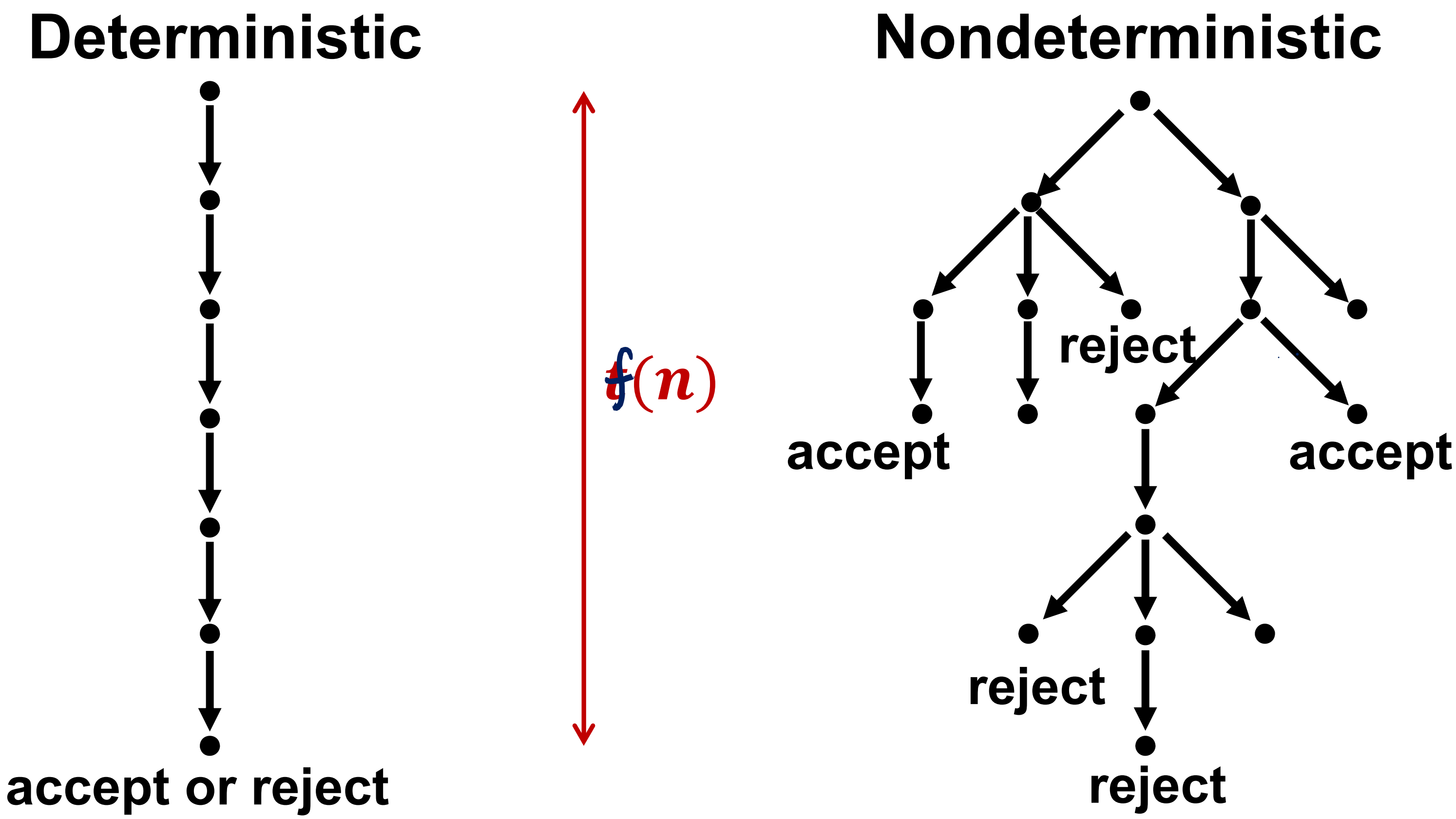
input length

Let $f : \mathbb{N} \rightarrow \mathbb{N}$ ← worst-case runtime

A NTM M runs in time $f(n)$ if on every input $w \in \Sigma^n$,

M halts on w within at most $f(n)$ steps on every computational branch

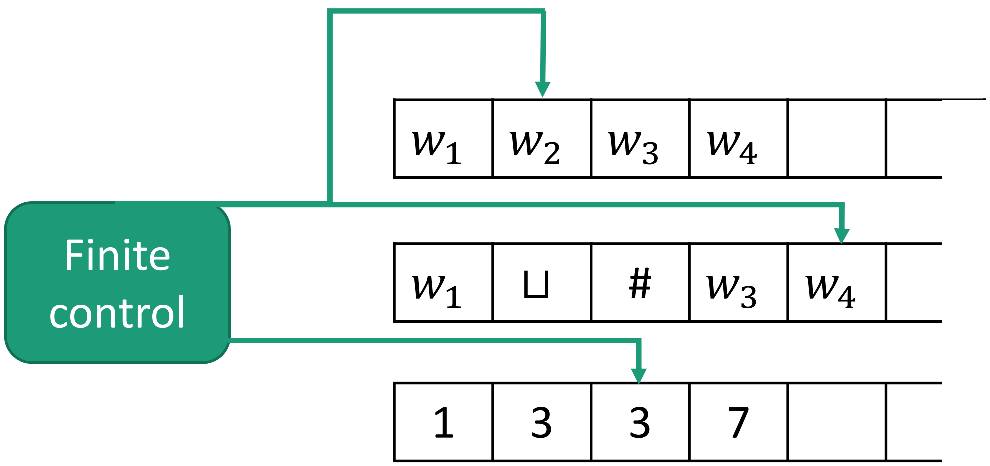
Deterministic vs. nondeterministic time



Deterministic vs. nondeterministic time

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

Proof: Simulate NTM by 3-tape TM



Input w to N (read-only)

Simulation tape (run N on w using nondeterministic choices from tape 3)

Address in computation tree

Counting leaves

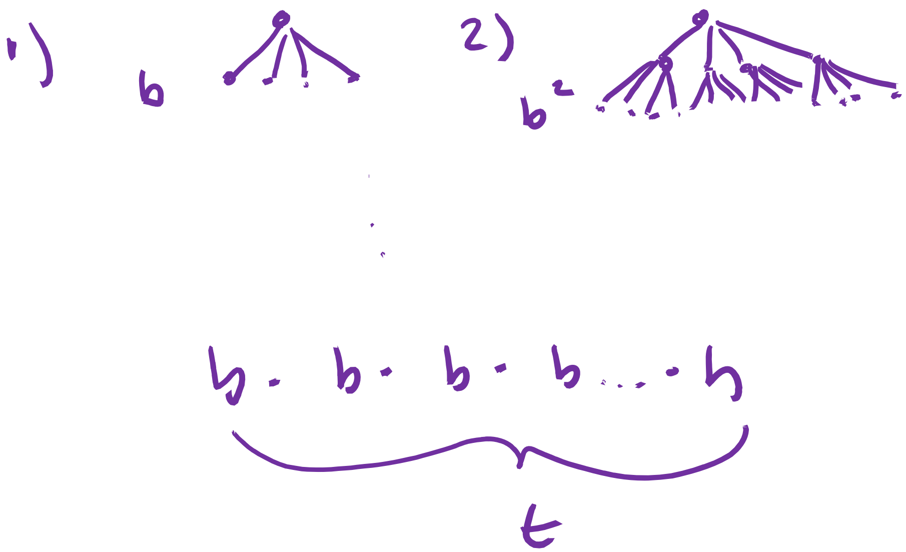
What is the maximum number of leaves in a tree with branching factor b and depth t?

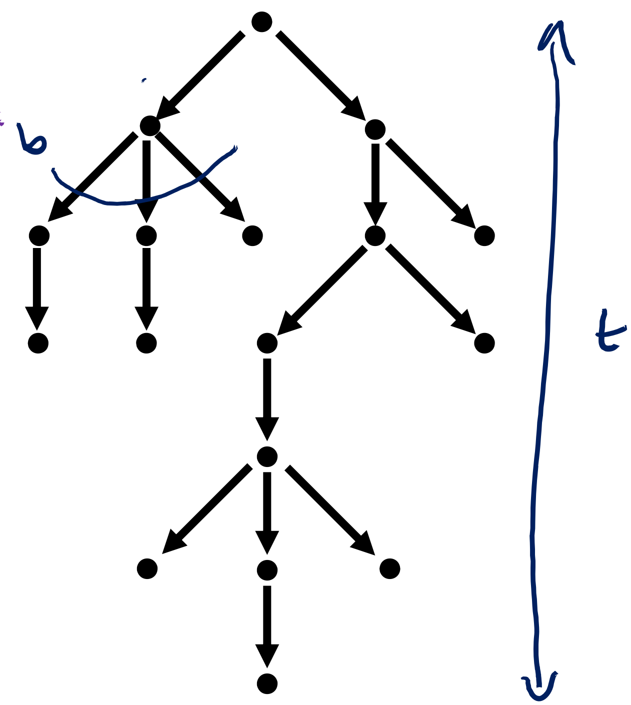
a) bt

b) b^t

c) t^b

d) 2^t



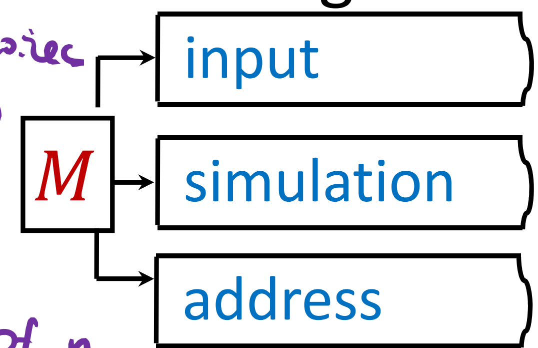


Deterministic vs. nondeterministic time

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

b: "branching factor" = max # of choices NTM can make in 1 step

b is constant independent of n



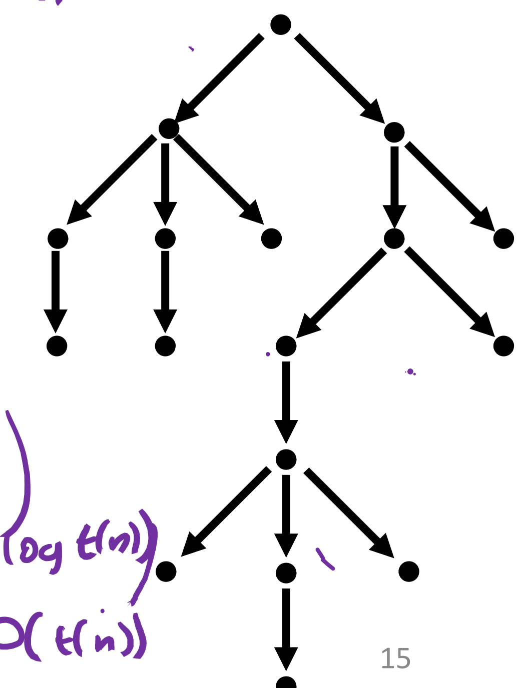
Proof: Simulate NTM by 3-tape TM

- # leaves: $b^{t(n)}$

<u>Running time:</u>

To simulate one root-to-leaf path:

Time $O(t(n))$



$2^{\log t(n)}$ =

Total time: $b^{t(n)} \cdot O(t(n)) = O(t(n) \cdot b^{t(n)})$

$= O(2^{t(n)\log b + \log t(n)})$

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

Deterministic vs. nondeterministic time

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

Proof: Simulate NTM by 3-tape TM in time $2^{O(t(n))}$

We know that a 3-tape TM can be simulated by a single-tape TM with quadratic overhead, hence we get running time

$$(2^{O(t(n))})^2 = 2^{2 \cdot O(t(n))} = 2^{O(t(n))}$$

Difference in time complexity

Extended Church-Turing Thesis:

At most polynomial difference in running time between all (reasonable) deterministic models

At most exponential difference in running time between deterministic and nondeterministic models

Nondeterministic time

Let $f : \mathbb{N} \rightarrow \mathbb{N}$

A NTM M runs in time $f(n)$ if on every input $w \in \Sigma^n$, M halts on w within at most $f(n)$ steps on every computational branch

$\mathrm{NTIME}(f(n))$ is a class (i.e., set) of languages:

A language $A \in \mathrm{NTIME}(f(n))$ if there exists an NTM M that

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

NTIME explicitly

A language $A \in \mathrm{NTIME}(f(n))$ if there exists an NTM M such that, on every input $w \in \Sigma^*$

1. Every computational branch of M halts in either the accept or reject state within $f(|w|)$ steps

   M is a decider running in time f(n)

2. If $w \in A$, then there exists an accepting computational branch of M on input w

   2+3 = M decides A

3. If $w \notin A$, then every computational branch of M rejects on input w

Complexity class NP

Definition: NP is the class of languages decidable in polynomial time on a nondeterministic TM

$$\text{NP} = \bigcup_{k=1}^{\infty} \text{NTIME}(n^k)$$

Which of the following are definitely true about NP?

a) $\text{P} \subseteq \text{NP}$

b) $\text{NP} \subseteq \text{P}$

c) $\text{NP} \not\subseteq \text{P}$

d) $\text{NP} \subseteq \text{EXP}$

e) $\text{EXP} \subseteq \text{NP}$

You win $1 m

???

P : deterministic poly time

EXP: deterministic exponential time

Nondet time can be simulated w/ det time w/ exponential blowup

$\text{NTIME}(t(n)) \subseteq \text{TIME}(2^{O(t(n))})$