BU CS 332 – Theory of Computation

Lecture 21:

- Complexity Class P
- Nondeterministic time, NP

Reading:
Sipser Ch 7.2, 7.3

Mark Bun
April 12, 2021
Complexity class $P$

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

$$P = \bigcup_{k=1}^{\infty} \text{TIME}(n^k) = \text{TIME}(n) \cup \text{TIME}(n') \cup \text{TIME}(n^3) \cup ...$$

- 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.
  \[ \Rightarrow \text{Can freely use algorithms we’ve seen before as subroutines if we’ve analyzed their runtime} \]

• Need to be careful about size of inputs! (Assume inputs represented in binary unless otherwise stated.)
Examples of languages in $\mathbb{P}$

$PATH = \{(G, s, t) | G$ is a directed graph with a directed path from $s$ to $t\}$

**Algorithm via BFS**

**On input $(G, s, t)$:**
1. Mark vertex $s$
2. Until no new vertices are being marked:
   - Mark all out-neighbors of currently marked vertices
3. Accept if $t$ is marked and reject otherwise

Runs in the $O(|E|)$ (all edges) polynomial in $|\langle G, s, t \rangle|$
Examples of languages in $\mathbf{P}$

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

Also note:

- Check at the end of BFS if any accept state has been marked.
Examples of languages in $P$

- $RELPRIME = \{ \langle x, y \rangle \mid x \text{ and } y \text{ are relatively prime} \}$
  - $x \text{ and } y \text{ relatively prime if } \gcd(x, y) = 1$
  - Euclid's algorithm
- $PRIMES = \{ \langle x \rangle \mid x \text{ is prime} \}$

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:

On input $\langle x \rangle$:

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

- Try to divide $x$ by $b$
- If $b$ divides $x$, accept

If all $b$ fail to divide $x$, reject

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) $2^{O(n)}$

4/12/2021  
CS332 - Theory of Computation 7
Beyond polynomial time

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

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

\[= \text{TIME}(2^n) \cup \text{TIME}(2^{n^2}) \cup \text{TIME}(2^{n^3}) \cup \ldots\]
Why study P?

Criticism of the Cobham-Edmonds Thesis:
- Algorithms running in time $n^{100}$ aren’t really efficient
  
  **Response:** Runtimes improve with more research
- Does not capture some physically realizable models using randomness, quantum mechanics
  
  **Response:** Randomness may not change P, useful principles

- \( \text{TIME}(n) \text{ vs. } \text{TIME}(n^2) \)
- \( P \text{ vs. } EXP \)
- Decidable vs. undecidable
Nondeterministic Time and NP
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.
Deterministic vs. nondeterministic time

Deterministic

accept or reject

Non-deterministic

accept

reject

accept

reject

\( t(n) \)
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

---

**Finite control**

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

**Proof:** Simulate NTM by 3-tape TM

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

Running time:

To simulate one root-to-leaf path:

$O(t(n))$

Total time:

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

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

↑

basic single-tape TM
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.

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

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

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

A language $A \in \text{NTIME}(f(n))$ 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

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

3. $w \notin A$ iff every computational branch of $M$ rejects on input $w$ (or dies with no applicable transitions)
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) \( P \subseteq \text{NP} \)

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

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

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

e) \( \text{EXP} \subseteq \text{NP} \)
Hamiltonian Path

\[ \text{HAMPATH} = \{ \langle G, s, t \rangle \mid G \text{ is a directed graph and there is a path from } s \text{ to } t \text{ that passes through every vertex exactly once} \} \]
**HAMPATH ∈ NP**

The following nondeterministic algorithm decides **HAMPATH** in polynomial time:

1. **Nondeterministically** guess a sequence $c_1, c_2, ..., c_k$ of numbers 1, ..., $k$
2. Check that $c_1, c_2, ..., c_k$ is a permutation: Every number 1, ..., $k$ appears exactly once
3. Check that $c_1 = s$, $c_k = t$, and there is an edge from every $c_i$ to $c_{i+1}$
4. **Accept** if all checks pass, otherwise, **reject**.

On input $\langle G, s, t \rangle$: (Vertices of $G$ are numbers 1, ..., $k$)
Analyzing the algorithm

Need to check:

1) Correctness
   - \( \langle 6, 5, t \rangle \in \text{HAMILTON} \Rightarrow \exists \text{ an accepting comp. branch} \)
   - \( \langle 6, 5, t \rangle \notin \text{HAMILTON} \Rightarrow \forall \text{ comp. branches reject} \)

2) Running time
   - Show poly time on NTM
   - (Check all comp. branches halt in poly time)