Last time
- Recursion theorem
- Measuring complexity
- Asymptotic notation
- Relationship between models

Today
- Relationship between models: deterministic/nondeterministic
- Class P
Let $t(n)$ be a function, where $t(n) \geq n$.

Every 3-tape TM that runs in time $O(t(n))$ can be simulated by a 1-tape TM that runs in time

A. $O(t(n))$
B. $O(t(n^2))$
C. $O(t(n^3))$
D. $O\left(\left(t(n)\right)^2\right)$
E. Some 3-tape TMs can’t be simulated by 1-tape TMs
Theorem. Let $t(n)$ be a function, where $t(n) \geq n$. Every $t(n)$ time multitape TM has an equivalent $O\left((t(n))^2\right)$ time 1-tape TM.
The running time a nondeterministic decider $N$ is $t(n)$ if on all inputs of length $n$, NTM $N$ takes at most $t(n)$ steps on the longest nondeterministic branch.
Time complexity of NTMs

- Length of the longest computational branch, even if accepts before
Complexity relationships between models: nondeterminism

**Theorem.** Let \( t(n) \) be a function, where \( t(n) \geq n \). Every \( t(n) \) time nondeterministic TM has an equivalent \( 2^{O(t(n))} \) time 1-tape deterministic TM.

**Proof:** Simulate an NTM by a 3-tape TM.
- # of leaves \( \leq b^{t(n)} \)
- # of nodes \( \leq 2b^{t(n)} \)

Time
- increment the address and simulate from the root to a node: \( O(t(n)) \)
- Total: \( O(t(n)b^{t(n)}) = 2^{O(t(n))} \)
Complexity relationships between models: nondeterminism

**Theorem.** Let $t(n)$ be a function, where $t(n) \geq n$. Every $t(n)$ time nondeterministic TM has an equivalent $2^{O(t(n))}$ time 1-tape deterministic TM.

**Proof:** So, a 3-tape TM can simulate an NTM in $2^{O(t(n))}$ time. Converting to a 1-tape TM at most squares the running time:

$$(2^{O(t(n))})^2 = 2^{O(2t(n))} = 2^{O(t(n))}$$
Difference in time complexity

At most \textit{polynomial} difference between \textit{all reasonable} deterministic models.

At most \textit{exponential} difference between deterministic and nondeterministic models.
The class $P$ is the class of languages decidable in polynomial time on a *deterministic* 1-tape TM:

$$P = \bigcup_{k} TIME(n^k).$$

- The same class even if we substitute another reasonable deterministic model.
- Roughly the class of problems realistically solvable on a computer.
Examples of languages in P

- **PATH** = \{ ⟨G, s, t⟩ | G is a directed graph that has a directed path from s to t \}
- **RELPRIME** = \{ ⟨x, y⟩ | x and y are relatively prime \}
- **PRIMES** = \{ x | x is a prime number \} [2002]

- Every context-free language (On the board)
Recall: Chomsky Normal Form for CFGs

- Can have a rule $S \rightarrow \varepsilon$.
- All remaining rules are of the form $A \rightarrow BC$ where $A, B, C \in V$
  $A \rightarrow a$ where $a \in \Sigma$
- Cannot have $S$ on the RHS of any rule.

**Lemma.** Any CFG can be converted into an equivalent CFG in Chomsky normal form.

**Lemma.** If $G$ is in Chomsky normal form, any derivation of string $w$ of length $n$ in $G$ has $2n - 1$ steps.
A decider for a CFL

- Let L be a CFL generated by a CFG G in CNF

\[ M = \text{``On input } \langle w \rangle \text{, where } w \text{ is a string:} \]
1. Let \( n = |w| \).
2. Test all derivations with \( 2n - 1 \) steps.
3. Accept if any derived \( w \). O.w. reject.”

- How long does it take? (Exponential time)
- Idea: use dynamic programming

(in the book)