Intro to Theory of Computation

CS 332

LECTURE 21

Last time

- Recursion theorem
- Measuring complexity
- Asymptotic notation

Today

- Measuring complexity
- Relationship between models
- Class P

Sofya Raskhodnikova

Exercise

In the future we will find algorithms for all computational problems, that is, problems with well-defined inputs and desired outputs.

A. True. I am an optimist.

B. It is difficult to make predictions, especially about the future. (*K.K. Steincke*)

C. False. Finitely many people will be able to design only finitely many algorithms.

D. False. There are more computational problems than algorithms.

Predictions

It is not hard to make predictions, it is hard to make *interesting* predictions

(of unpredictable events you don't control).

- It will be dark tonight at 11pm.
- Most people in this room will have another meal today.
- The exercise from the previous slide will appear on the final.

Running time analysis

If M is a TM and $f : \mathbb{N} \to \mathbb{N}$ then

"M runs in time $f(n)$" means

for **every** input $w \in \Sigma^*$ of length n,

M on w halts within $f(n)$ steps

- Focus on worst case:
  - upper bound on running time for all inputs of given length
- Exact time depends on computer
  - instead measure **asymptotic growth**

CS 332

Time complexity classes

TIME($f(n)$) is a class of languages.

$A \in$ **TIME($f(n)$)** means that

some 1-tape TM M

that runs in time O($f(n)$) decides A.

How much time/memory needed to decide a language?

Example: Consider $A = \{0^m 1^m | m \geq 0\}$.

- $M_1 =$ "
  1. Scan input and reject if it is not of the form 0^*1^*.
  2. Repeat while both 0s and 1s remain on the tape:
  3. Cross off one 0 and one 1
  4. **Accept** if no 0s and no 1s left; otherwise **reject**."
- M_1 runs in time $O(n^2)$.
- $A \in TIME(n^2)$.
- Is there a faster algorithm?

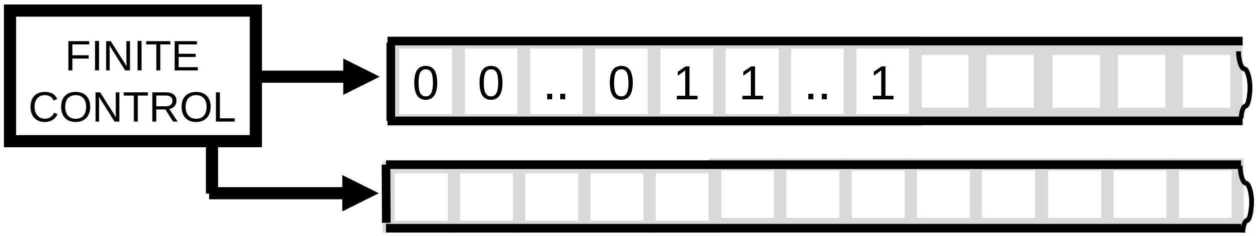
CS 332 How much time/memory needed to decide a language?

Example: Consider $A = \{0^m 1^m | m \geq 0\}$.

- M_2=“ 1. Scan input and reject if it is not of the form $0^* 1^*$.
  2. Repeat while both 0s and 1s remain on the tape:
  3. **Reject** if total number of 0s and 1s remaining is odd.
  4. Cross off every other 0 starting from the first 0 and every other 1 starting from the first 1
  5. **Accept** if no 0s and no 1s left; otherwise **reject**.”
- M_2 runs in time $O(n \log n)$, so $A \in TIME(n \log n)$.
- Sipser, Problem 7.49: If language L can be decided in $o(n \log n)$ time on a 1-tape TM then L is regular.
- 1-tape TM need $\Omega(n \log n)$ time to decide A.

Two-tape TM can do it faster

Example: Consider $A = \{0^m 1^m | m \geq 0\}$.



- M_3=" 1. Scan input and reject if it is not of the form $0^* 1^*$.
  2. Copy 0s on tape 2.
  3. Scan tape 1. For each 1 read, cross off a 0 on tape 2.
  4. **Accept** if no 0s remain on tape 2; otherwise **reject**."
- A is decided in $O(n)$ time (linear time) on a 2-tape TM.

Unlike decidability,

the complexity of the language depends on the model.

CS 332 Complexity relationships between models: number of tapes

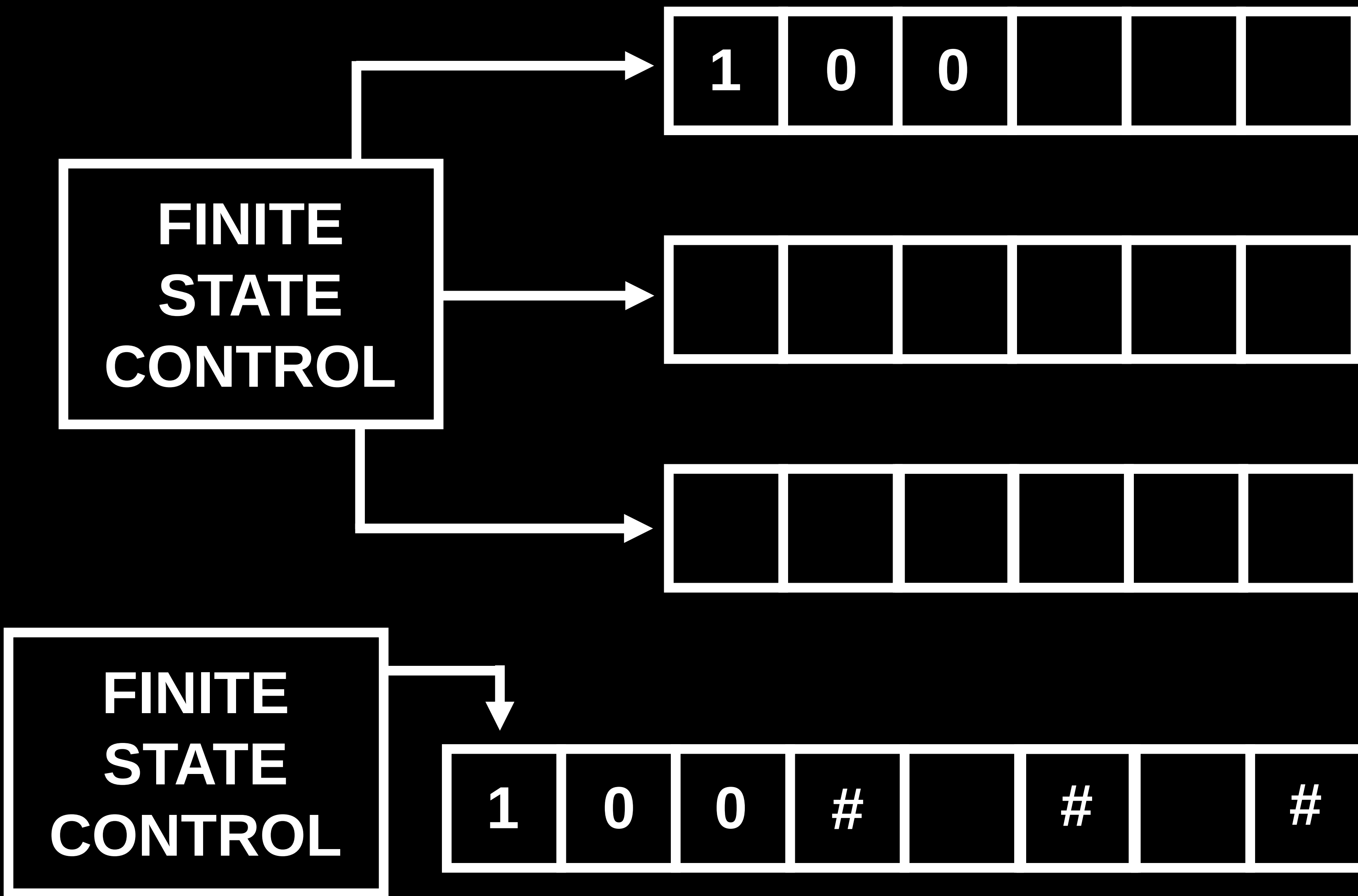
Theorem. Let $t(n)$ be a function, where $t(n) \geq n$.
Every $t(n)$ time multitape TM has

an equivalent $O\left(\left(t(n)\right)^2\right)$ time 1-tape TM.

Proof:

- Recall: we already showed
  how to simulate multitape TMs by 1-tape TMs.
- Need time analysis of the simulation.

Theorem: Every Multitape Turing Machine can be transformed into a single-tape Turing Machine



SIMULATING MULTIPLE TAPES

L#100# 0 # 1 #R

q_{jRSS} q_{i1} $q_{i1\square}$ $q_{i1\square}$ $q_{j101RSS}$

1. "Format" tape.

2. For each move of the k-tape TM:

   Scan left-to-right, finding current symbols

   Scan left-to-right, writing new symbols

   Scan left-to-right, moving each tape head.

3. If a tape head goes off right end, insert blank.

   If tape head goes off left end, move back right.

Complexity relationships between models: number of tapes

Theorem. Let $t(n)$ be a function, where $t(n) \geq n$.
Every $t(n)$ time multitape TM has

an equivalent $O\left(\left(t(n)\right)^2\right)$ time 1-tape TM.

Proof: Time analysis of the simulation.

- Time initialize tape: $O(n+k) = O(n)$
- Time to simulate one step of the multitape TM: $O(t(n))$
  (at any point $\leq t(n)$ nonblank squares on each tape)
- Number of steps to simulate: $t(n)$

Total time: $O(n) + O\big(t(n)\big)t(n) = O((t(n)^2)$

Exercise

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**

CS 332

The class P

P is the class of languages decidable in polynomial time on a ***deterministic*** 1-tape TM:

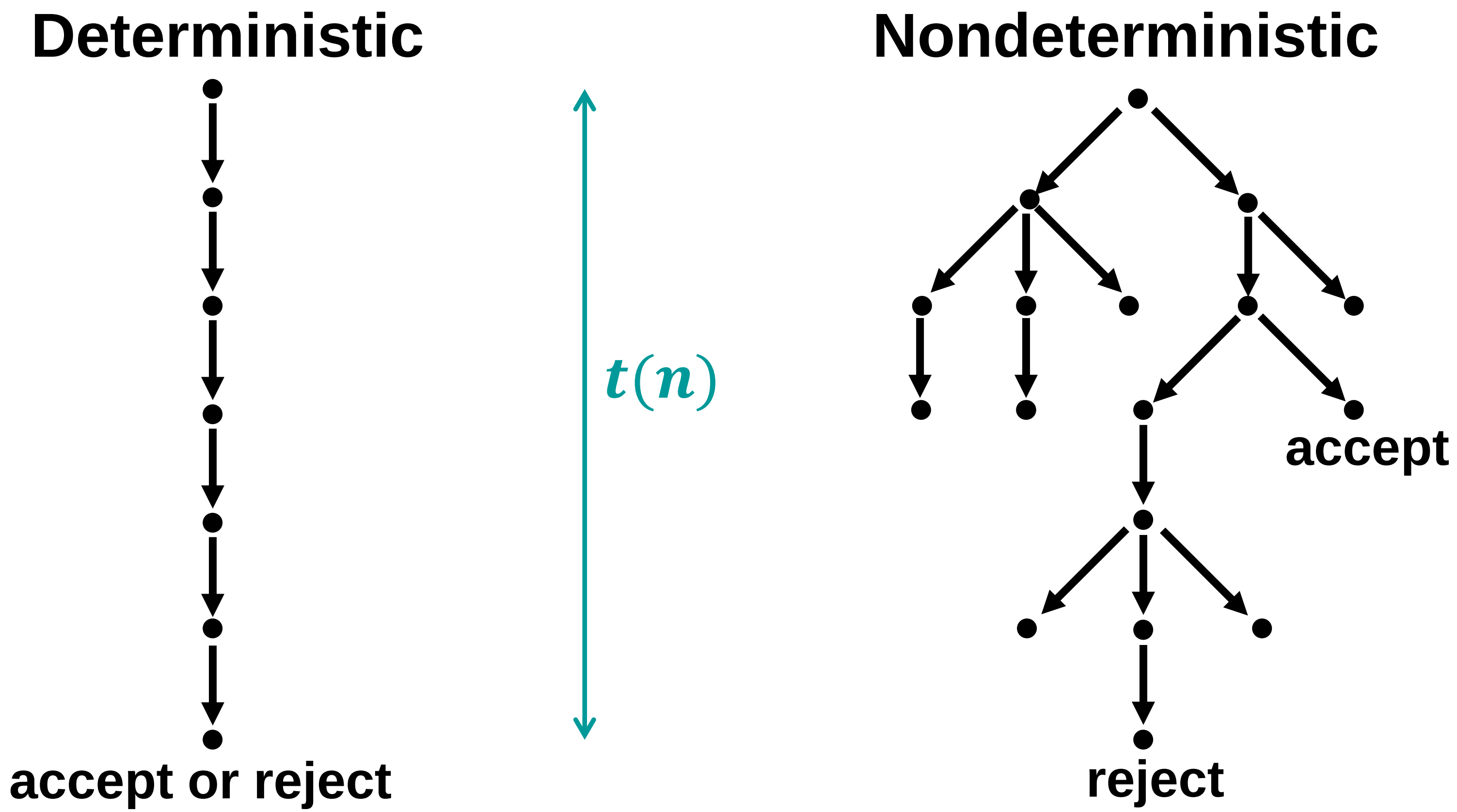
$$\boldsymbol{P} = \bigcup_{k} TIME(n^k).$$

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

Time complexity of NTMs

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

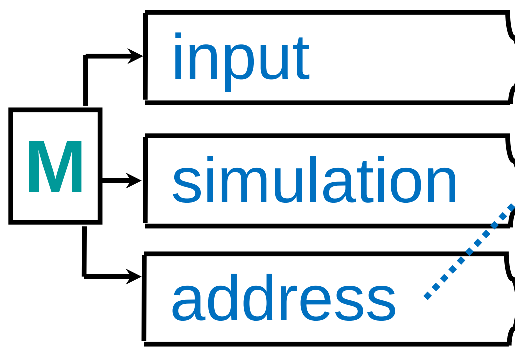
CS 332

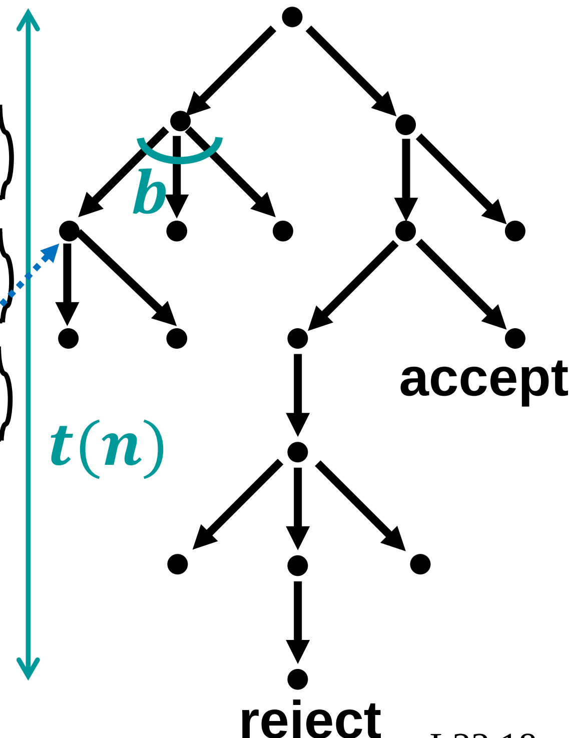
Complexity relationships between models: nondeterminism

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 $\leq \boldsymbol{b^{t(n)}}$
- # of nodes $\leq \boldsymbol{2b^{t(n)}}$





Time

- increment the address and

simulate from the root to a node: $\boldsymbol{O(t(n))}$

- Total: $\boldsymbol{O(t(n) b^{t(n)}) = 2^{O(t(n))}}$

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:

$$\left(2^{O(t(n))}\right)^{\mathbf{2}} = 2^{O(\mathbf{2}\,t(n))} = 2^{O(t(n))}$$

Difference in time complexity

At most ***polynomial*** difference between ***all reasonable*** deterministic models.

At most ***exponential*** difference between deterministic and nondeterministic models.

The class P

P is the class of languages decidable in polynomial time on a ***deterministic*** 1-tape TM:

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

- 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 $=\{\langle G, s, t\rangle \mid G$ is a directed graph that has a directed path from s to $t\}$
- RELPRIME $= \{\langle x, y\rangle \mid x$ and y are relatively prime$\}$
- PRIMES $=\{x \mid x$ is a prime number$\}$ [2002]

- Every context-free language **(On the board)**

Recall: Chomsky Normal Form for CFGs

- Can have a rule $S \to \varepsilon$.
- All remaining rules are of the form

$$A \to BC \qquad\qquad A, B, C \in V$$

$$A \to a \qquad\qquad a \in \Sigma$$

- 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 = `` **On input $\langle w \rangle$, where w 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 **(on the board)**
  - Solve smaller subproblems
  - Record results in a table
  - Construct solution for each subproblem from smaller solved instances