



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen



CSC 6011 - Theory of Computation

Lecture 10 - Time Complexity

Time: Mon & Wed, 4:00 - 5:20 PM

Location: Teaching A, 107

Instructor: Tao LIN (林涛)

TA: Yefan GAO (高叶繁)

Fall 2026

Lecture 10 - Time Complexity

Last time:

- The Reduction method for proving undecidability and T-unrecognizability.
- General reduction
- Mapping reduction

Finishes the Computability Theory part.

Today:

- Introduction to Complexity Theory
- Complexity classes; the Class P

Intro to Complexity Theory

Computability theory (1930s - 1950s):

Is A decidable?

Complexity theory (1960s - present):

*Is A decidable with restricted resources?
(time/memory/...)*

Example: Let $A = \{a^k b^k \mid k \geq 0\}$.

Q: How many steps are needed to decide A ?

Depends on the input.

We give an upper bound on the number of steps for **all inputs** of length n .

Called “**worst-case complexity**”.

steps to decide $A = \{a^k b^k \mid k \geq 0\}$

Theorem: A 1-tape TM M can decide A where, on inputs of length n , M uses at most cn^2 steps, for some fixed constant c .

Terminology: M uses $O(n^2)$ steps.

Proof: $M =$ "On input w

1. Scan input to check if $w \in a^*b^*$, *reject* if not.
2. Repeat until all crossed off:
 - Scan tape, crossing off one a and one b .
 - *Reject* if only a 's or only b 's remain.
3. *Accept* if all crossed off. "

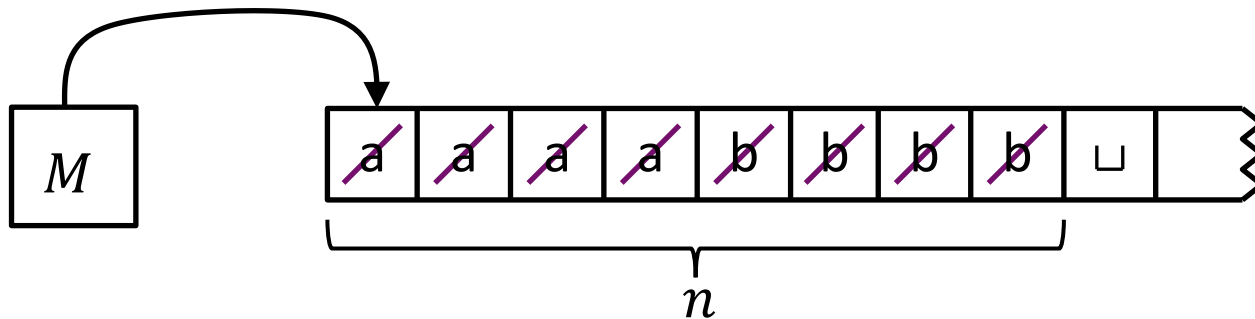
Analysis:

$$\left. \begin{array}{l} O(n) \text{ steps} \\ + O(n) \text{ iterations} \\ \times O(n) \text{ steps} \end{array} \right\} O(n) + O(n^2) = O(n^2) \text{ steps}$$

Big O and little o

Defn: $f(n)$ is $O(g(n))$ if $f(n) \leq cg(n)$ for some fixed c independent of n .

Defn: $f(n)$ is $o(g(n))$ if $f(n) \leq \epsilon g(n)$ for any $\epsilon > 0$ and large n .



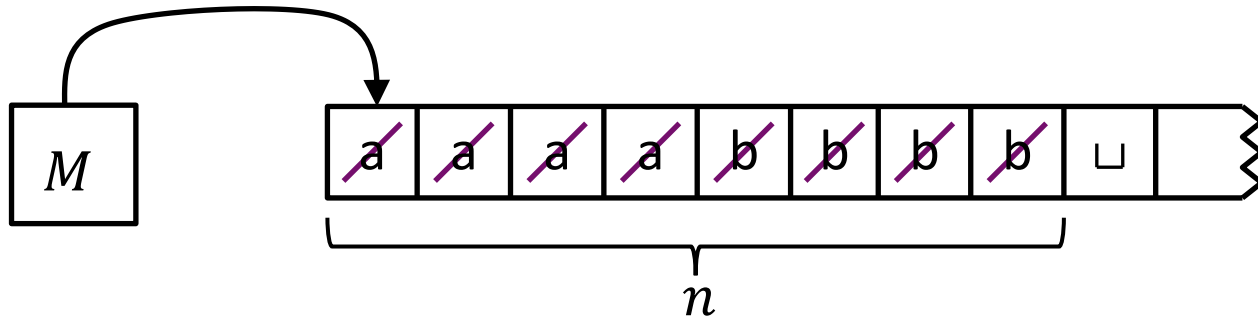
steps to decide $A = \{a^k b^k \mid k \geq 0\}$

Theorem: A 1-tape TM M can decide A where, on inputs of length n , M uses at most cn^2 steps, for some fixed constant c .

Terminology: M uses $O(n^2)$ steps.

Proof: $M =$ "On input w

1. Scan input to check if $w \in a^*b^*$, *reject* if not.
2. Repeat until all crossed off:
 - Scan tape, crossing off one a and one b .
 - *Reject* if only a 's or only b 's remain.
3. *Accept* if all crossed off. "



Analysis:

$O(n)$ steps
+ $O(n)$ iterations
× n

$O(n) + O(n^2)$
 $= O(n^2)$ steps

Mini-Quiz 10.1

Can this $O(n^2)$ bound be improved?
(Is there a 1-tape TM that decides A using a fewer number of steps?)

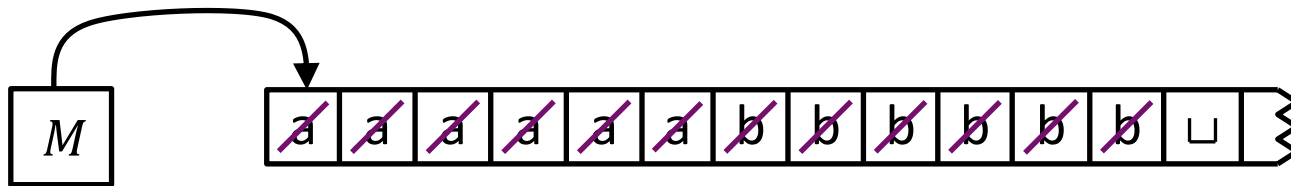
- (a) $O(n^2)$ is best possible.
- (b) $O(n \log n)$ is possible.
- (c) $O(n)$ is possible.

Deciding $A = \{a^k b^k \mid k \geq 0\}$ faster

Theorem: A 1-tape TM M can decide A by using $O(n \log n)$ steps.

Proof: M = "On input w

1. Scan tape to check if $w \in a^*b^*$. *Reject* if not.
2. Repeat until all crossed off:
 - Scan tape, crossing off every other a and b .
 - Record the even/odd parity of a and b .
 - *Reject* if the parities of a and b disagree.
3. *Accept* if all crossed off. "



Analysis:

$O(n)$ steps
+ $O(\log n)$ iterations
× $O(n)$ steps

} $O(n \log n)$ steps

	Parity		
a	even (6)	odd (3)	odd (1)
b	even (6)	odd (3)	odd (1)

Binary representations of #a and #b

Further improvement? Not possible.

Theorem: Any language decidable by a deterministic 1-tape TM in $o(n \log n)$ time is regular (and can be decided in $O(n)$ time).

Deciding $A = \{a^k b^k \mid k \geq 0\}$ even faster

Theorem: A multi-tape TM M can decide A using $O(n)$ steps.

M = “On input w

1. Scan input to check if $w \in a^*b^*$, *reject* if not.
2. Copy a ’s to second tape.
3. Match b ’s with a ’s on second tape.
4. *Accept* if match, else *reject*. ”

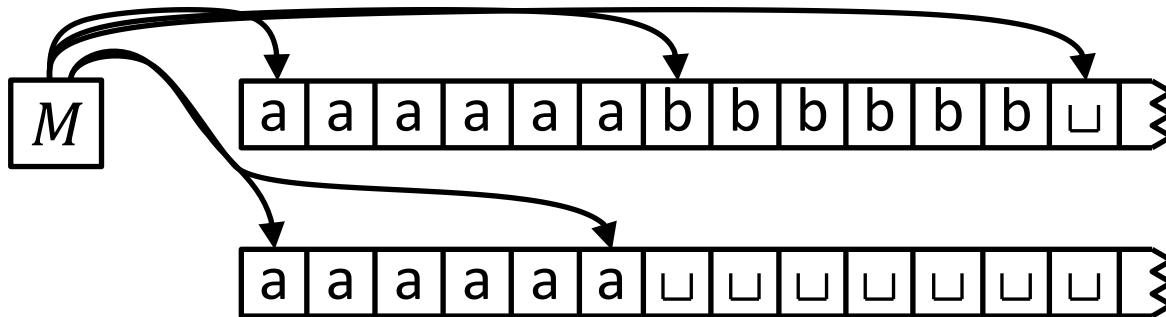
Analysis:

$O(n)$ steps

+ $O(n)$ steps

+ $O(n)$ steps

= $O(n)$ steps



Model Dependence

Number of steps to decide $A = \{a^k b^k \mid k \geq 0\}$ depends on the model:

- **1-tape TM:** $O(n \log n)$
- **Multi-tape TM:** $O(n)$

Computability theory: model independence (Church-Turing Thesis)

Therefore model choice doesn't matter. Mathematically nice.

Complexity Theory: model dependence

But dependence is low (polynomial) for reasonable deterministic models.

We will focus on questions that do not depend on the model choice.

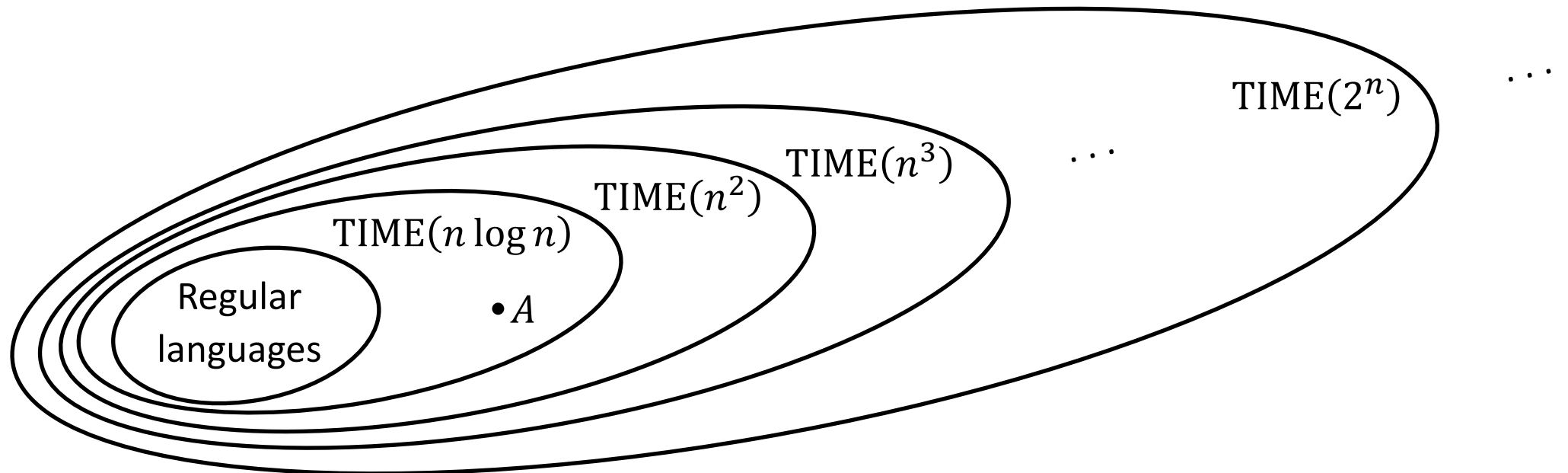
So... we will continue to use the 1-tape TM as the basic model for complexity.

TIME Complexity Classes

Defn: Let $t: \mathbb{N} \rightarrow \mathbb{N}$. Say TM M runs in time $t(n)$ if M always halts within $t(n)$ steps on all inputs of length n .

Defn: $\text{TIME}(t(n)) = \{ \text{language } B \mid \text{some deterministic 1-tape TM } M \text{ decides } B \text{ and } M \text{ runs in time } O(t(n)) \}$

Example: $A = \{a^k b^k \mid k \geq 0\} \in \text{TIME}(n \log n)$

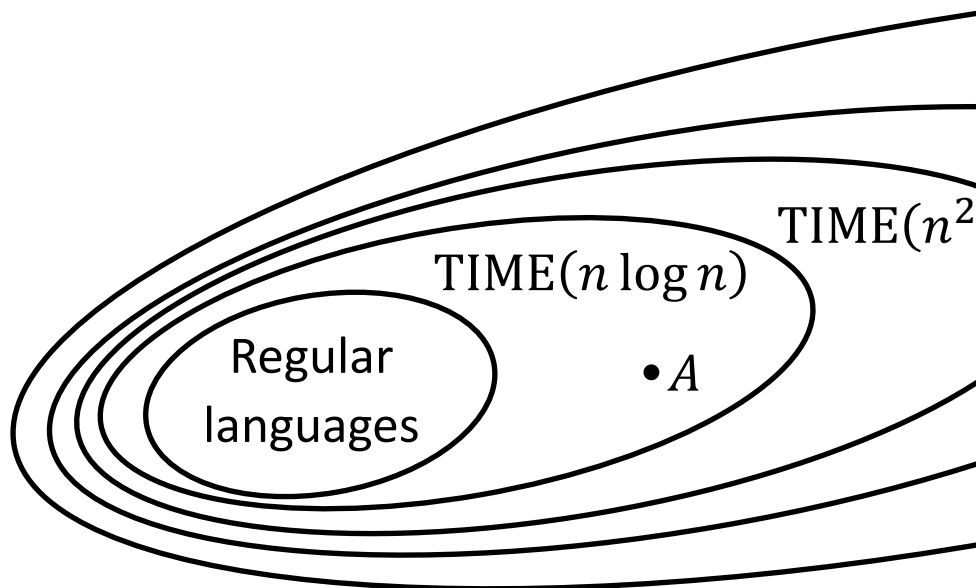


TIME Complexity Classes

Defn: Let $t: \mathbb{N} \rightarrow \mathbb{N}$. Say TM M runs in time $t(n)$ if M always halts within $t(n)$ steps on all inputs of length n .

Defn: $\text{TIME}(t(n)) = \{ \text{language } B \mid \text{some deterministic 1-tape TM } M \text{ decides } B \text{ and } M \text{ runs in time } O(t(n)) \}$

Example: $A = \{a^k b^k \mid k \geq 0\} \in \text{TIME}(n \log n)$



Mini-Quiz 10.2

Let $B = \{ww^R \mid w \in \{a, b\}^*\}$ be the language of Palindromes. What is the smallest function t such that $B \in \text{TIME}(t(n))$?

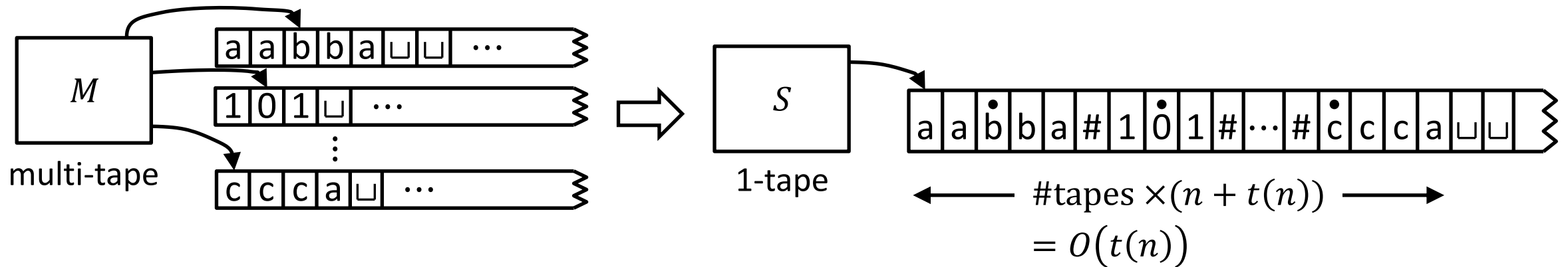
- (a) $O(n)$
- (b) $O(n \log n)$
- (c) $O(n^2)$
- (d) $O(n^3)$

Multi-tape vs 1-tape time

Theorem: Let $t(n) \geq n$.

If a multi-tape TM decides B in time $t(n)$, then $B \in \text{TIME}(t^2(n))$.

Proof: Analyze the conversion of multi-tape to 1-tape TMs.



To simulate 1 step of M 's computation, S uses $O(t(n))$ steps.

So total simulation time is $t(n) \times O(t(n)) = O(t^2(n))$.

Similar results can be shown for other reasonable deterministic models.

Relationships among models

Informal Definition: Two models of computation are polynomially related if each can simulate the other with a polynomial overhead:

So $t(n)$ time on one model $\rightarrow t^k(n)$ time on the other model, for some k .

All reasonable deterministic models are polynomially related.

- 1-tape TMs
- multi-tape TMs
- multi-dimensional TMs
- random access machine (RAM)
- cellular automata

2-dimensional TM

0	0	1	0	□
1	0	0	1	1
0	1	1	0	1
1	0	1	1	□
1	0	0	□	□
⋮				

...

Cellular automaton



“Game of Life”

The Class P

Defn: $P = \bigcup_k \text{TIME}(n^k)$
= polynomial time decidable languages

- Invariant for all reasonable deterministic models
- Corresponds roughly to realistically solvable problems

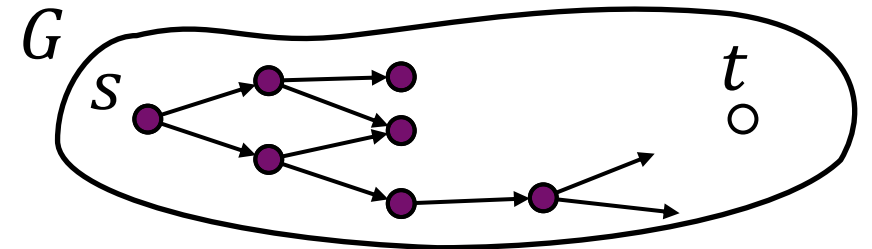
Example: $\text{PATH} = \{ \langle G, s, t \rangle \mid G \text{ is a directed graph with a path from } s \text{ to } t \}$

Theorem: $\text{PATH} \in P$

Proof: $M =$ “On input $\langle G, s, t \rangle$

1. Mark s
2. Repeat until nothing new is marked:
For each marked node x :
Scan G to mark all y where (x, y) is an edge
3. *Accept* if t is marked. *Reject* if not.

To show polynomial time: Each stage should be clearly polynomial and the total number of stages polynomial.



$$\begin{array}{l} \leq n \text{ rounds} \\ \times \leq n \text{ iterations} \\ \times O(n^2) \text{ steps} \\ \hline O(n^4) \text{ steps} \end{array}$$

PATH and HAMPATH

Example: $\text{HAMPATH} = \{ \langle G, s, t \rangle \mid G \text{ is a directed graph with a path from } s \text{ to } t \text{ and the path goes through every node exactly once in } G \}$

Called a Hamiltonian path

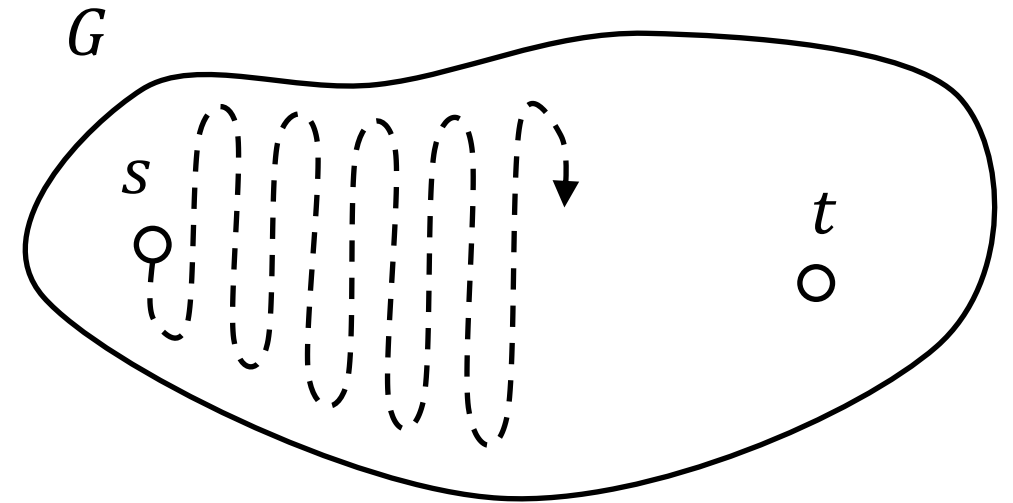
Recall Theorem: $\text{PATH} \in \text{P}$

Question: $\text{HAMPATH} \in \text{P} ?$

“On input $\langle G, s, t \rangle$

1. Let m be the number of nodes in G .
2. For each path with m nodes in G :
test if it is a Hamiltonian path from s to t .
Accept if yes.
3. *Reject* if all paths fail.”

May be $m! > 2^m$ paths with m nodes, so the algorithm is exponential time, not polynomial time.



PATH and HAMPATH

Example: $\text{HAMPATH} = \{ \langle G, s, t \rangle \mid G \text{ is a directed graph with a path from } s \text{ to } t \text{ and the path goes through every node exactly once in } G \}$

Called a Hamiltonian path

Recall Theorem: $\text{PATH} \in \text{P}$

Question: $\text{HAMPATH} \in \text{P} ?$

“On input $\langle G, s, t \rangle$

1. Let m be the number of nodes in G .
2. For each path with m nodes in G :
test if it is a Hamiltonian path from s to t .
Accept if yes.
3. *Reject* if all paths fail.”

May be $m! > 2^m$ paths with m nodes, so

Mini-Quiz 10.3

Is $\text{HAMPATH} \in \text{P} ?$

- (a) Definitely Yes. You have a polynomial-time algorithm.
- (b) Probably Yes. It should be similar to showing $\text{PATH} \in \text{P}$.
- (c) Toss a coin.
- (d) Probably No. Hard to beat the exponential algorithm.
- (e) Definitely No. You can prove it!

Open problem: no proved answer is known.

Quick review of today

1. Introduction to Complexity Theory
2. Which model to use? 1-tape-TMs
3. $\text{TIME}(t(n))$ complexity classes
4. The class P
5. $\text{PATH} \in \text{P}$

These slides are adapted from Michael Sipser, *18.404J Theory of Computation*, Fall 2020.
Massachusetts Institute of Technology:

- Source: MIT OpenCourseWare <https://ocw.mit.edu>

Except for separately identified third-party material, this adapted presentation is licensed under [CC BY-NC-SA 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/).