بسم الله الرحمن الرحیم

نظریه علوم کامپیوتر

نظریه علوم کامپیوتر - بهار ۱۴۰۰-۱۴۰۱ - جلسه دهم: پیچیدگی محاسبات

Theory of computation - 002 - S10 - computational complexity

Intro to Complexity Theory

Intro to Complexity Theory

Computability theory (1930s - 1950s):

Intro to Complexity Theory

Computability theory (1930s - 1950s):

Is A decidable?

Intro to Complexity Theory

Computability theory (1930s - 1950s):

Is A decidable?

Complexity theory (1960s - present):

Intro to Complexity Theory

Computability theory (1930s - 1950s):

Is A decidable?

Complexity theory (1960s - present):

Is A decidable with restricted resources?

Intro to Complexity Theory

Computability theory (1930s - 1950s):

Is A decidable?

Complexity theory (1960s - present):

Is A decidable with restricted resources?

(time/memory/…)

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 $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$.

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 $A = \left\{ \mathrm{a}^k\mathrm{b}^k \,\middle|\, k \geq 0 \right\}$.

Q: How many steps are needed to decide A?

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 $A = \left\{ \mathrm{a}^k\mathrm{b}^k \middle| \; k \geq 0 \right\}$.

Q: How many steps are needed to decide A?

Depends on the input.

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 $A = \{ \mathrm{a}^k \mathrm{b}^k \mid k \geq 0 \}$.

Q: How many steps are needed to decide A?

Depends on the input.

We give an <u>upper bound</u> for all inputs of length n.

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 $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$.

Q: How many steps are needed to decide A?

Depends on the input.

We give an <u>upper bound</u> for all inputs of length n.

Called “worst-case complexity”.

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 $A = \left\{ \mathrm{a}^k \mathrm{b}^k \middle| \ k \geq 0 \right\}$.

Q: How many steps are needed to decide A?

Depends on the input.

We give an <u>upper bound</u> for all inputs of length n.

Called "worst-case complexity".

steps to decide

$$A = \left\{ \mathsf{a}^k \mathsf{b}^k \,\middle|\, k \geq 0 \right\}$$

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.

steps to decide

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

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.

steps to decide

$$A = \left\{ \mathsf{a}^k \mathsf{b}^k \,\middle|\, k \geq 0 \right\}$$

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.

steps to decide

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

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.

Big O and little o

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

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

steps to decide

$$A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$$

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

Big O and little o

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

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

steps to decide

$$A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$$

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.

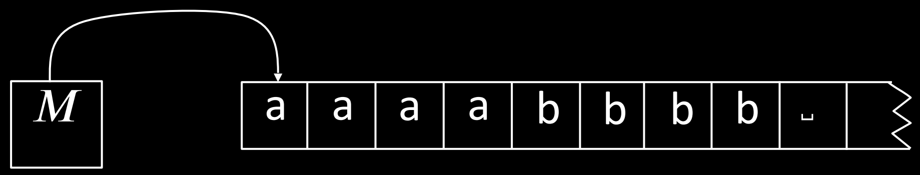
Terminology: M uses $O(n^2)$ steps.

Proof: $M =$ "On input w

Big O and little o

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

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



steps to decide

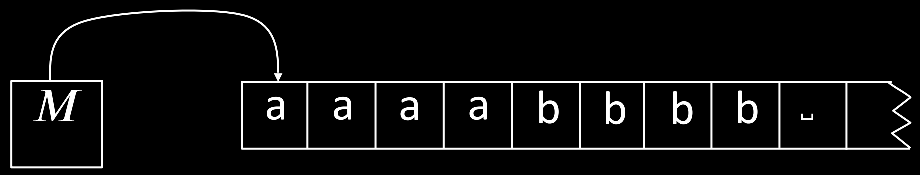
$$A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$$

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 $w \in \mathrm{a}^*\mathrm{b}^*$, *reject* if not.



Big O and little o

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

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

steps to decide

$$A = \left\{ \text{a}^k\text{b}^k \,\middle|\, k \geq 0 \right\}$$

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.

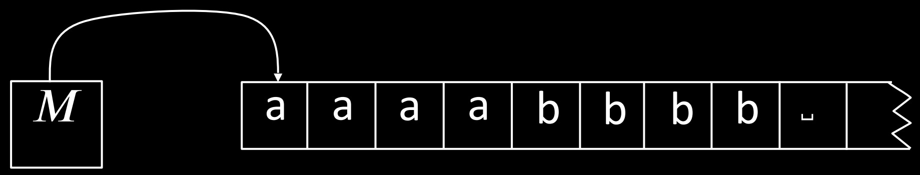
Proof: $M =$ "On input w

1. Scan input to check if $w \in$ a*b*, *reject* if not.
2. Repeat until all crossed off.

Big O and little o

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

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



steps to decide

$$A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$$

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.

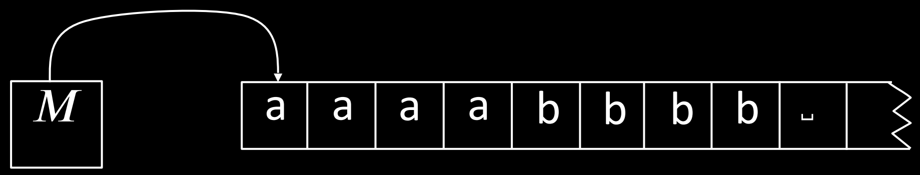
Proof: $M =$ "On input w

1. Scan input to check if $w \in \mathrm{a}^*\mathrm{b}^*$, *reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off one a and one b.

Big O and little o

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

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



steps to decide

$$A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$$

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.

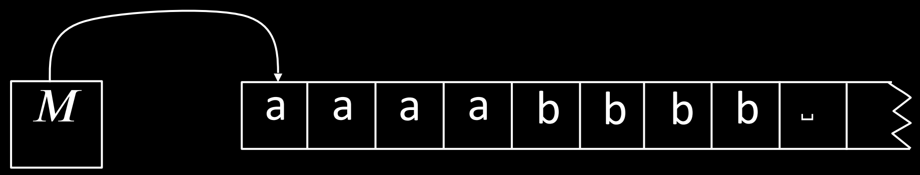
Terminology: M uses $O(n^2)$ steps.

Proof: $M =$ "On input w

1. Scan input to check if $w \in \mathrm{a}^*\mathrm{b}^*$, *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.



Big O and little o

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

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

steps to decide

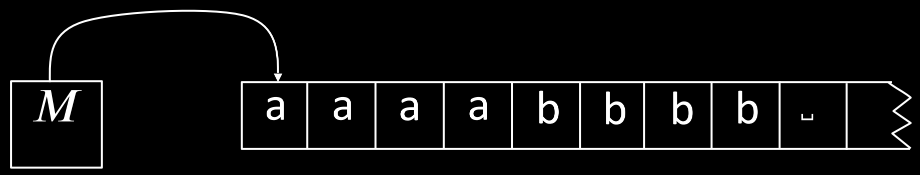
$$A = \{\mathrm{a}^k\mathrm{b}^k \mid k \geq 0\}$$

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 $w \in \mathrm{a}^*\mathrm{b}^*$, *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. "



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 all $\epsilon > 0$ and large n.

steps to decide

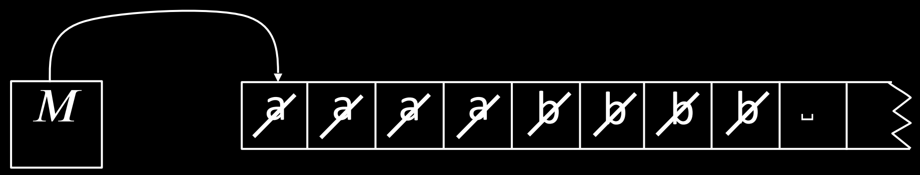
$$A = \{ \mathrm{a}^k \mathrm{b}^k \mid k \geq 0 \}$$

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 $w \in \mathrm{a}^*\mathrm{b}^*$, *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. "



Big O and little o

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

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

steps to decide

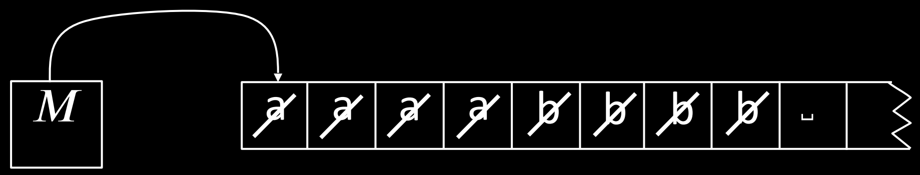
$$A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$$

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

Terminology: M uses $\underline{O(n^2)}$ steps.

Proof: $M =$ "On input w

1. Scan input to check if $w \in \mathrm{a}^*\mathrm{b}^*$, *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:

Big O and little o

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

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

steps to decide

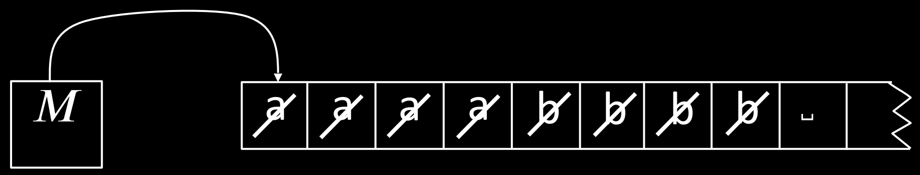
$$A = \{\mathrm{a}^k\mathrm{b}^k \mid k \geq 0\}$$

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 $w \in \mathrm{a}^*\mathrm{b}^*$, *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



Big O and little o

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

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

steps to decide

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

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.

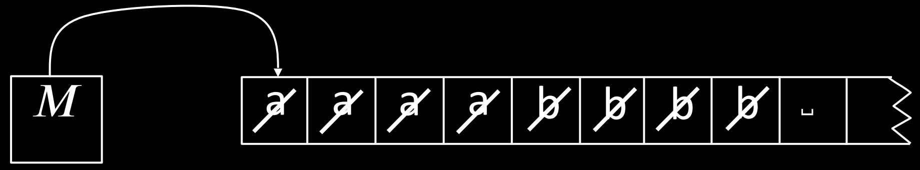
Proof: $M =$ "On input w

1. Scan input to check if $w \in \mathrm{a}^*\mathrm{b}^*$, *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



Big O and little o

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

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

steps to decide

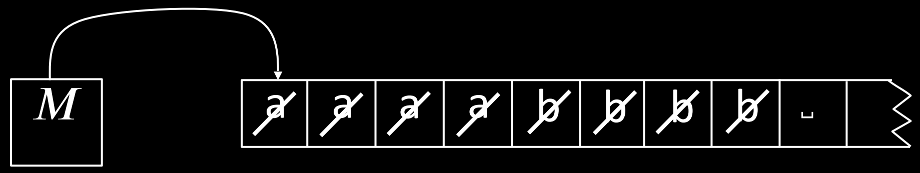
$$A = \{ \mathrm{a}^k \mathrm{b}^k \mid k \geq 0 \}$$

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 $w \in \mathrm{a}^*\mathrm{b}^*$, *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

$\times O(n)$ 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 all $\epsilon > 0$ and large n.

steps to decide

$$A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$$

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 $w \in \mathrm{a}^*\mathrm{b}^*$, *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

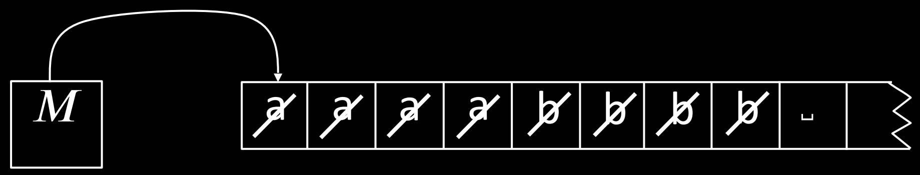
$+O(n)$ iterations

$\times\, O(n)$ steps

Big O and little o

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

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



steps to decide

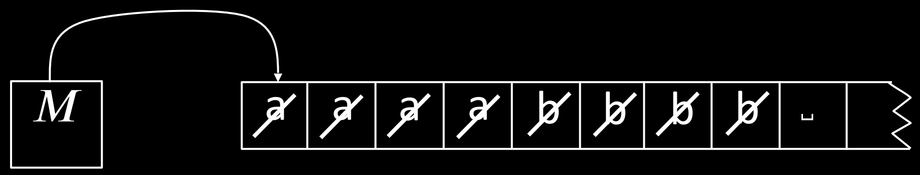
$$A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$$

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 $w \in \mathrm{a}^*\mathrm{b}^*$, *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

$\times O(n)$ steps

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

$= O(n^2)$ steps

Big O and little o

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

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

steps to decide

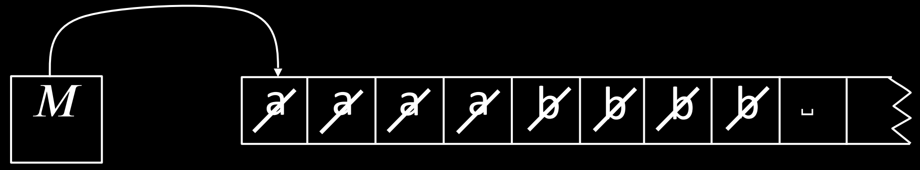
$$A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$$

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 $w \in \mathrm{a}^*\mathrm{b}^*$, *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

$\times O(n)$ steps

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

$= O(n^2)$ steps

Big O and little o

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

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

Check-in 12.1

steps to decide

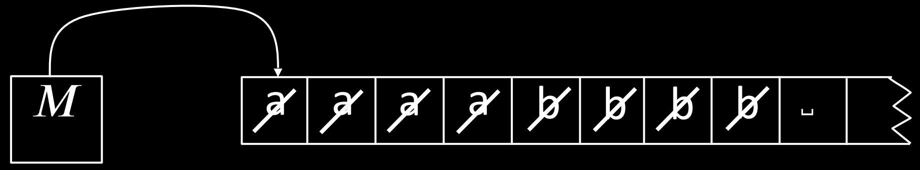
$$A = \{ \mathrm{a}^k \mathrm{b}^k \mid k \geq 0 \}$$

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 $w \in \mathrm{a}^*\mathrm{b}^*$, *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

$\times\, O(n)$ steps

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

$= O(n^2)$ steps

Check-in 12.1

How much improvement is possible in the bound for this theorem about 1-tape TMs deciding A?

(a) $O(n^2)$ is best possible.

(b) $O(n \log n)$ is possible.

(c) $O(n)$ is possible.

Check-in 12.1

Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ faster

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

Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \middle| \ k \geq 0 \right\}$ faster

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

Proof:

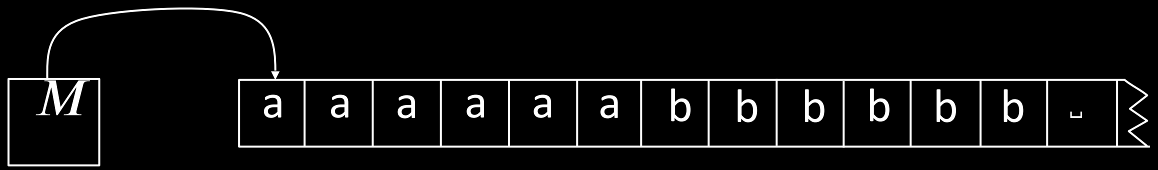
$M =$ “On input w

Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ faster

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

Proof:

$M =$ "On input w



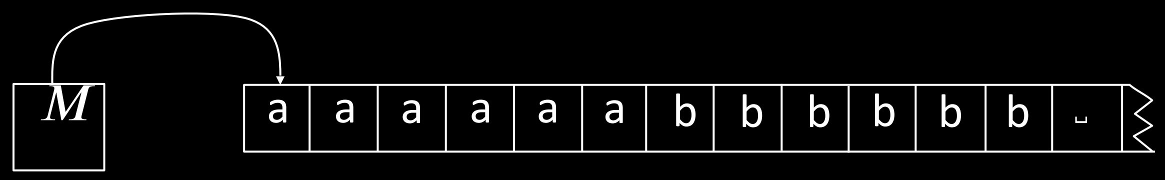
Deciding $A = \{a^k b^k \mid k \geq 0\}$ faster

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.



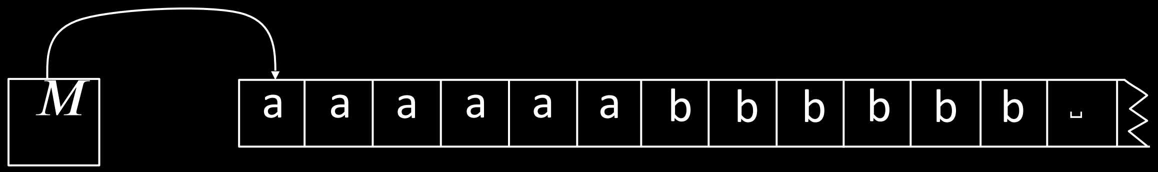
Deciding $A = \{a^k b^k \mid k \geq 0\}$ faster

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.



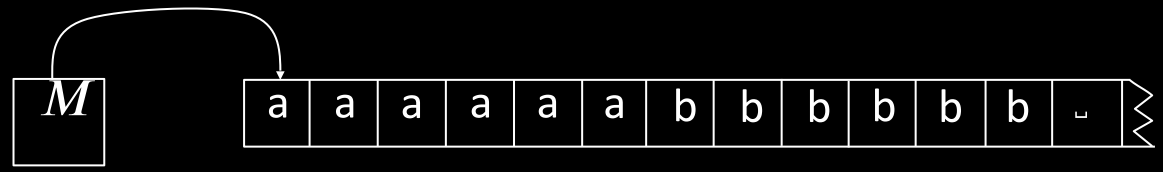
Deciding $A = \left\{ \mathrm{a}^k\mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ faster

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 $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.



Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ faster

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

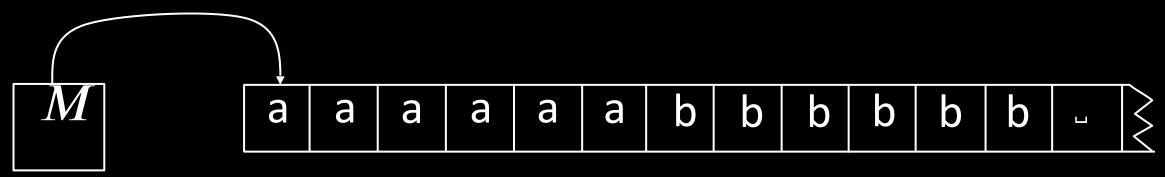
Proof:

$M =$ "On input w

1. Scan tape to check if $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.

    Scan tape, crossing off every other a and b.

    Reject if even/odd parities disagree.



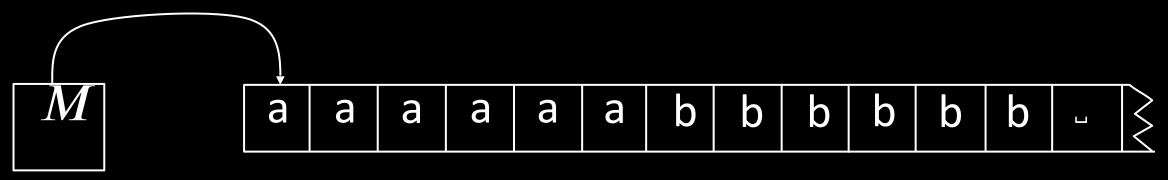
Deciding $A = \left\{ a^k b^k \,\middle|\, k \geq 0 \right\}$ faster

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.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "



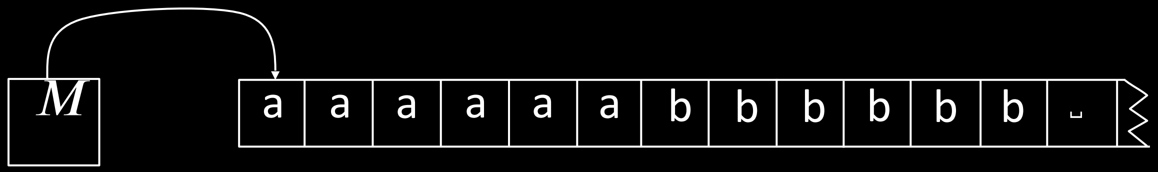
Deciding $A = \{a^k b^k \mid k \geq 0\}$ faster

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.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "



	Parities
a's	
b's	

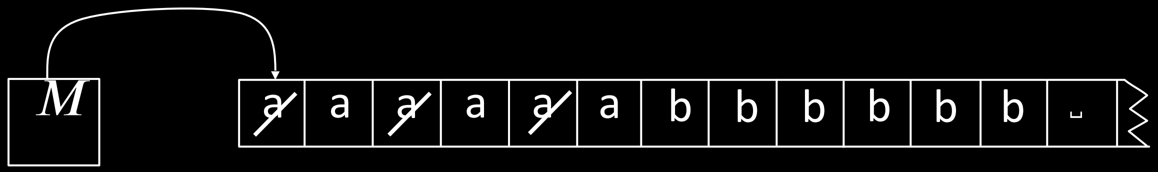
Deciding $A = \{\mathrm{a}^k\mathrm{b}^k \mid k \geq 0\}$ faster

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 $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "



	Parities
a's	
b's	

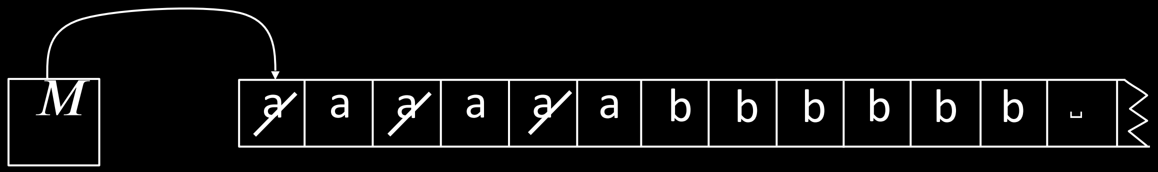
Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ faster

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 $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "



	Parities
a's	even (6)
b's	

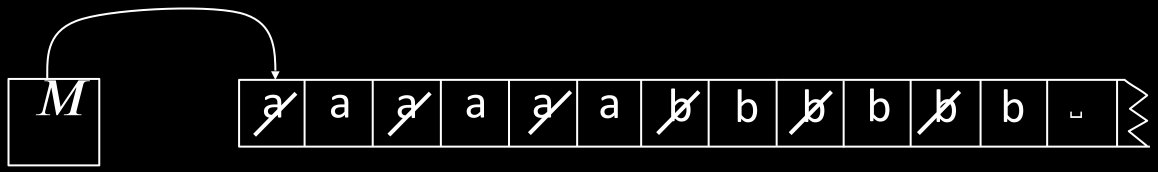
Deciding $A = \{a^k b^k \mid k \geq 0\}$ faster

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.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "



	Parities
a's	even (6)
b's	

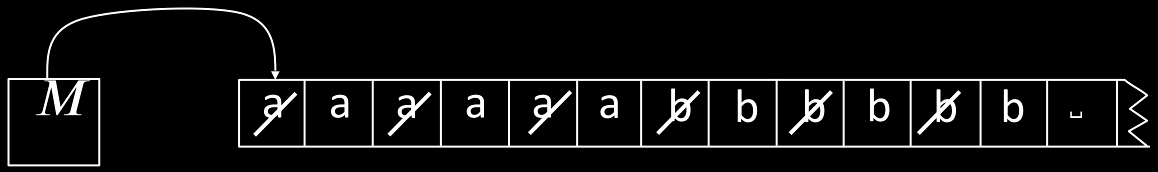
Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ faster

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 $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "



	Parities
a's	even (6)
b's	even (6)

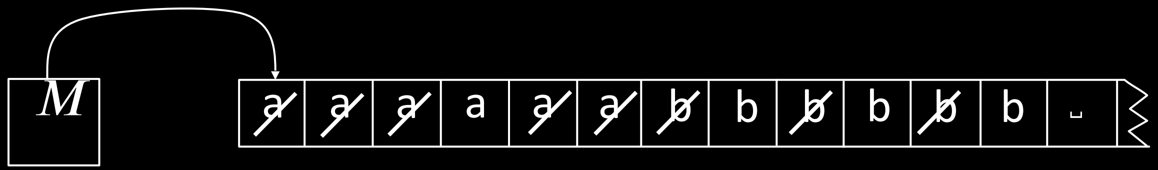
Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ faster

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 $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "



	Parities
a's	even (6)
b's	even (6)

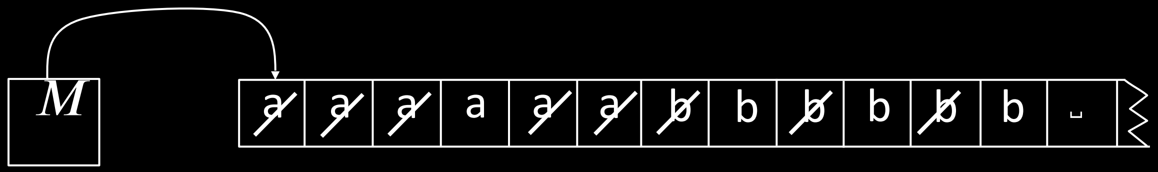
Deciding $A = \left\{ \mathrm{a}^k\mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ faster

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 $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "



	Parities	
a's	even (6)	odd (3)
b's	even (6)	

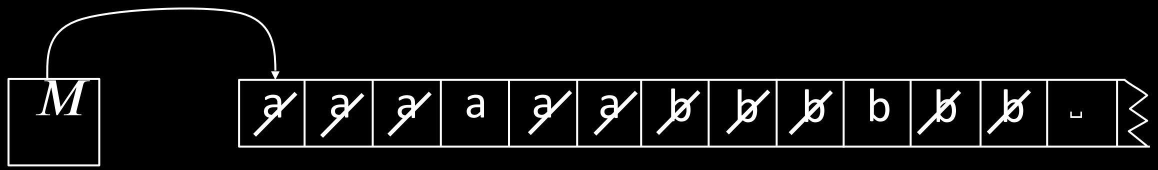
Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ faster

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 $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "



	Parities	
a's	even (6)	odd (3)
b's	even (6)	

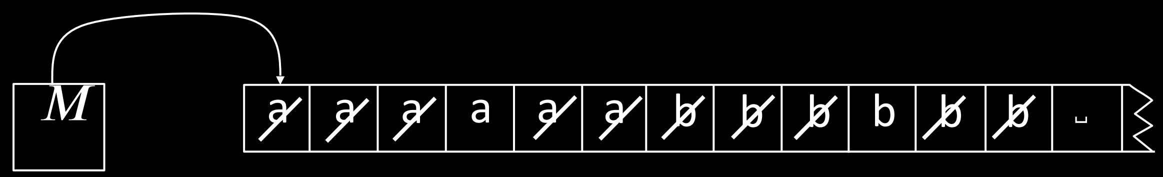
Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \middle| \; k \geq 0 \right\}$ faster

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 $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "



	Parities	
a's	even (6)	odd (3)
b's	even (6)	odd (3)

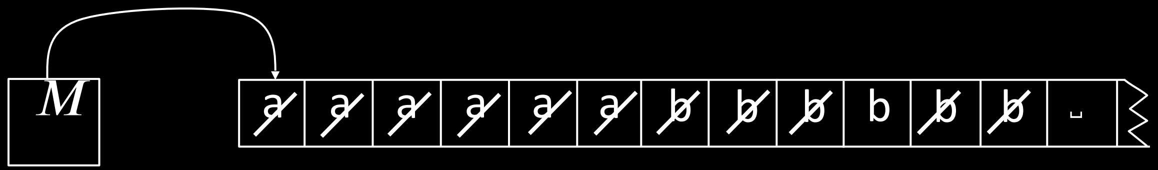
Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ faster

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 $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "



	Parities	
a's	even (6)	odd (3)
b's	even (6)	odd (3)

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.

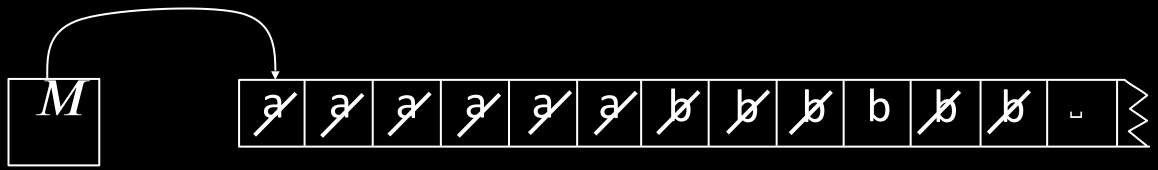
Proof:

$M =$ "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.

    Reject if even/odd parities disagree.
3. Accept if all crossed off. "



	Parities		
a's	even (6)	odd (3)	odd (1)
b's	even (6)	odd (3)	

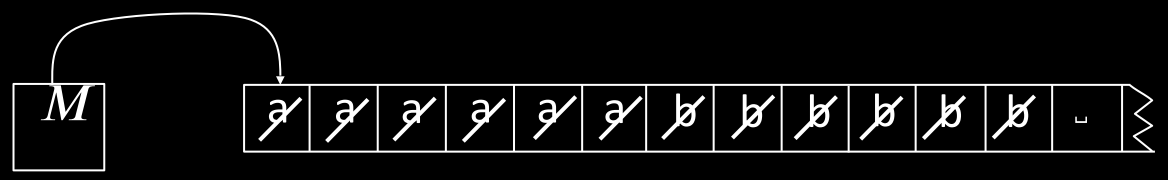
Deciding $A = \{a^k b^k \mid k \geq 0\}$ faster

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.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "



	Parities		
a's	even (6)	odd (3)	odd (1)
b's	even (6)	odd (3)	

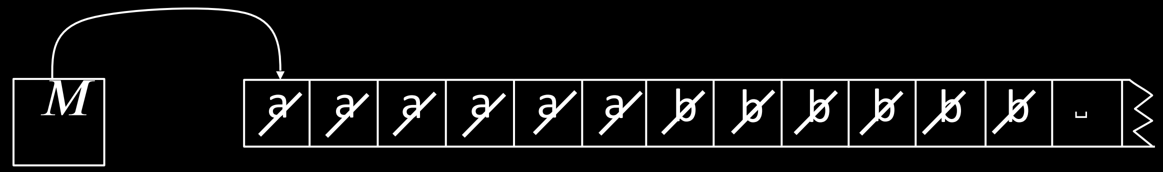
Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ faster

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 $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "



	Parities		
a's	even (6)	odd (3)	odd (1)
b's	even (6)	odd (3)	odd (1)

Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ faster

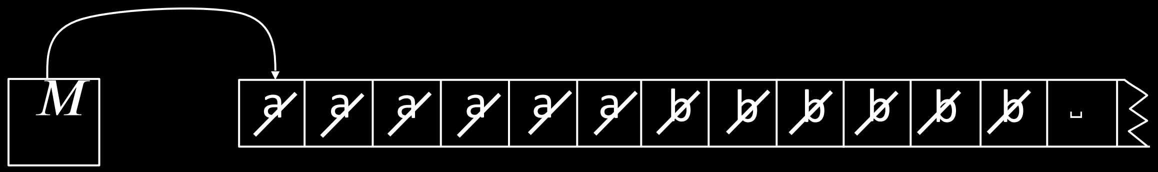
Theorem: A 1-tape TM M can decide A by using $O(n\log n)$ steps.

Proof:

$M =$ "On input w

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

Analysis:



	Parities		
a's	even (6)	odd (3)	odd (1)
b's	even (6)	odd (3)	odd (1)

Deciding $A = \left\{ \mathrm{a}^k\mathrm{b}^k \middle|\ k \geq 0 \right\}$ faster

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

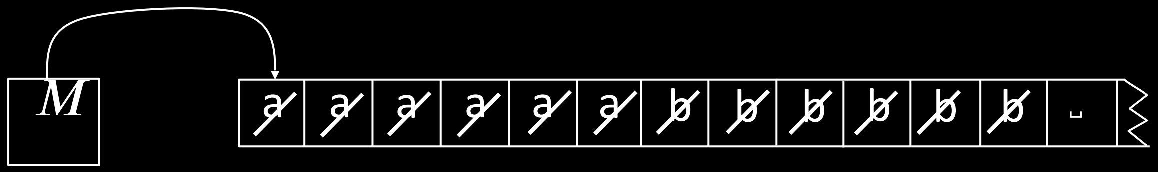
Proof:

$M =$ "On input w

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

Analysis:

$O(n)$ steps



	Parities		
a's	even (6)	odd (3)	odd (1)
b's	even (6)	odd (3)	odd (1)

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:

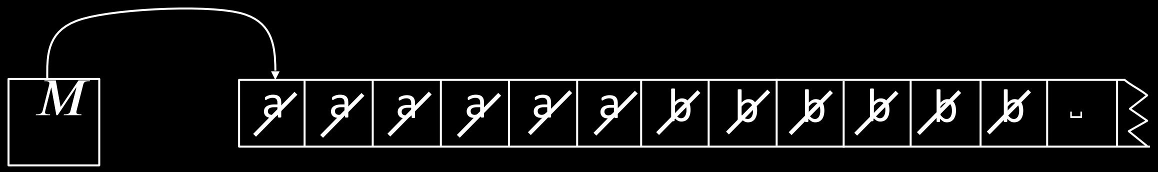
$M =$ "On input w

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.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "

Analysis:

$O(n)$ steps

$+O(\log n)$ iterations



	Parities		
a's	even (6)	odd (3)	odd (1)
b's	even (6)	odd (3)	odd (1)

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

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

Proof:

$M =$ "On input w

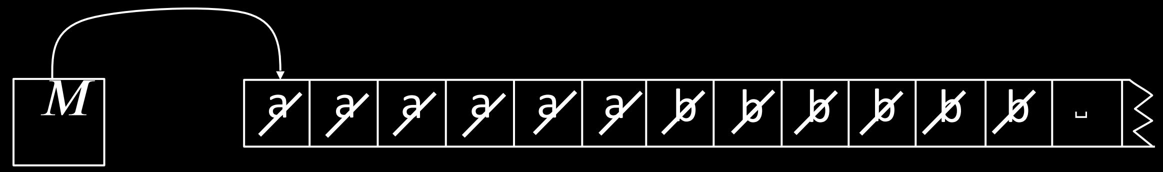
1. Scan tape to check if $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "

Analysis:

$O(n)$ steps

$+O(\log n)$ iterations

$\times O(n)$ steps



	Parities		
a's	even (6)	odd (3)	odd (1)
b's	even (6)	odd (3)	odd (1)

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

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

Proof:

$M =$ "On input w

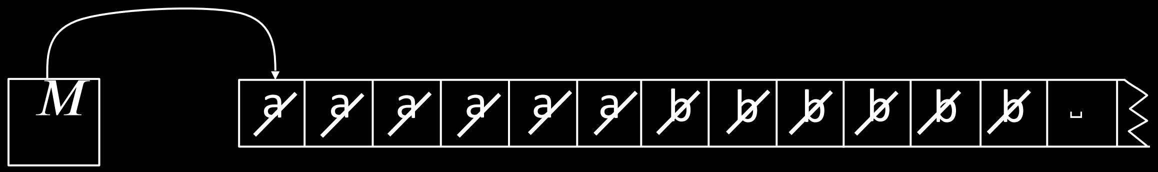
1. Scan tape to check if $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "

Analysis:

$O(n)$ steps

$+O(\log n)$ iterations

$\times O(n)$ steps



	Parities		
a's	even (6)	odd (3)	odd (1)
b's	even (6)	odd (3)	odd (1)

Deciding $A = \left\{ \text{a}^k\text{b}^k \,\middle|\, k \geq 0 \right\}$ 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 $w \in \text{a}^*\text{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "

Analysis:

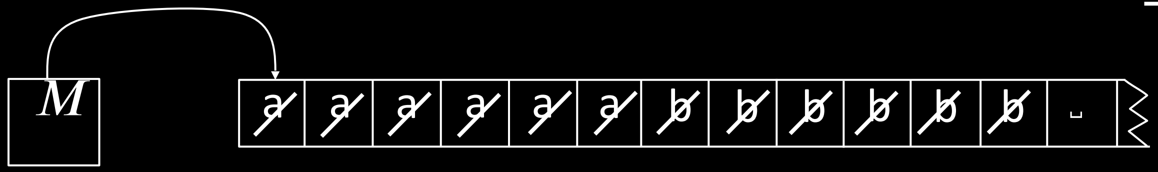
$O(n)$ steps

$+O(\log n)$ iterations

$\times O(n)$ steps

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

$= O(n\log n)$ steps



	Parities		
a's	even (6)	odd (3)	odd (1)
b's	even (6)	odd (3)	odd (1)

Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ 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 $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "

Analysis:

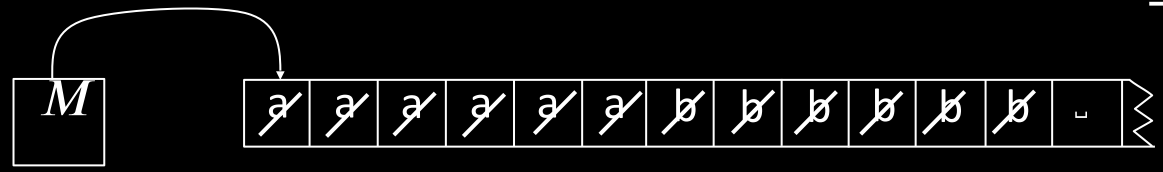
$O(n)$ steps

$+O(\log n)$ iterations

$\times\, O(n)$ steps

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

$= O(n\log n)$ steps



	Parities		
a's	even (6)	odd (3)	odd (1)
b's	even (6)	odd (3)	odd (1)

Further improvement? Not possible.

Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ 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 $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "

Analysis:

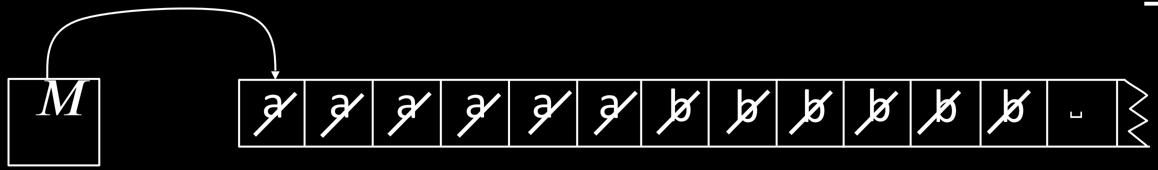
$O(n)$ steps

$+O(\log n)$ iterations

$\times\, O(n)$ steps

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

$= O(n\log n)$ steps



	Parities		
a's	even (6)	odd (3)	odd (1)
b's	even (6)	odd (3)	odd (1)

Further improvement? Not possible.

Theorem: A 1-tape TM M cannot decide A by using $o(n\log n)$ steps.

Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ 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 $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "

Analysis:

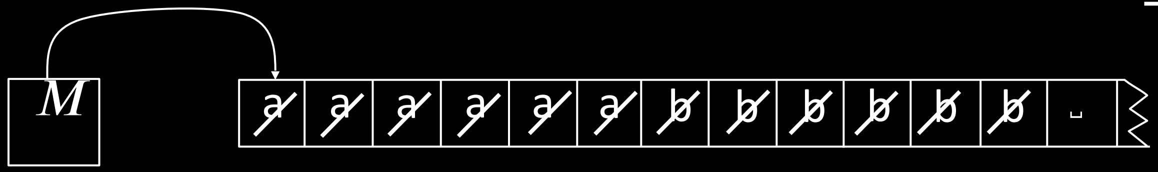
$O(n)$ steps

$+O(\log n)$ iterations

$\times\, O(n)$ steps

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

$= O(n\log n)$ steps



	Parities		
a's	even (6)	odd (3)	odd (1)
b's	even (6)	odd (3)	odd (1)

Further improvement? Not possible.

Theorem: A 1-tape TM M cannot decide A by using $o(n\log n)$ steps.

You are not responsible for knowing the proof.

Deciding $A = \{ \mathrm{a}^k \mathrm{b}^k \mid k \geq 0 \}$ 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 $w \in \mathrm{a}^*\mathrm{b}^*$. *Reject* if not.
2. Repeat until all crossed off.
   Scan tape, crossing off every other a and b.
   Reject if even/odd parities disagree.
3. Accept if all crossed off. "

Analysis:

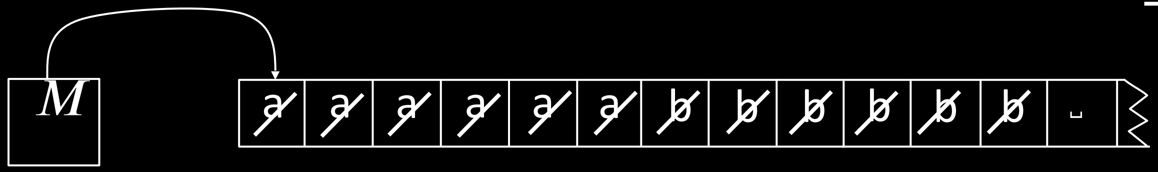
$O(n)$ steps

$+O(\log n)$ iterations

$\times O(n)$ steps

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

$= O(n \log n)$ steps



	Parities		
a's	even (6)	odd (3)	odd (1)
b's	even (6)	odd (3)	odd (1)

Further improvement? Not possible.

Theorem: A 1-tape TM M cannot decide A by using $o(n \log n)$ steps.

You are not responsible for knowing the proof.

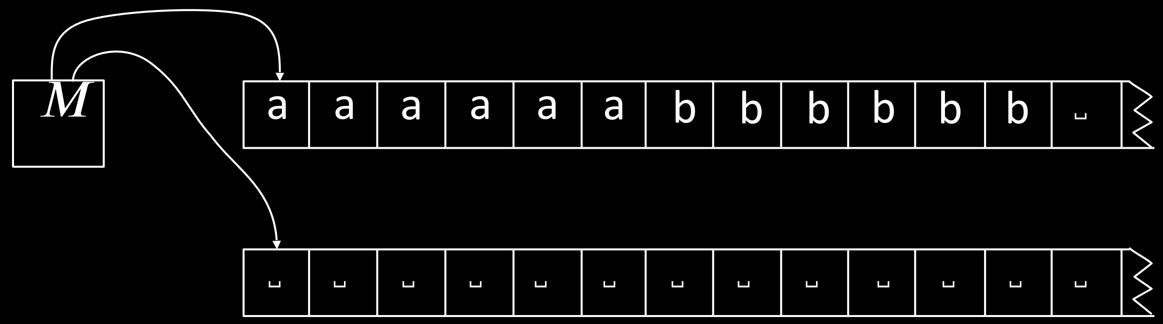
Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ even faster

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

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

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

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

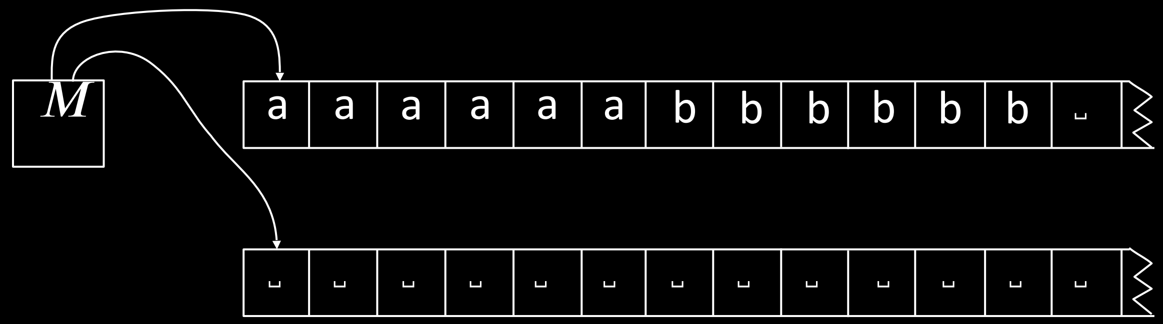


Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ even faster

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

$M =$ "On input w

1. Scan input to check if $w \in \mathrm{a}{*}\mathrm{b}^*$, *reject* if not.

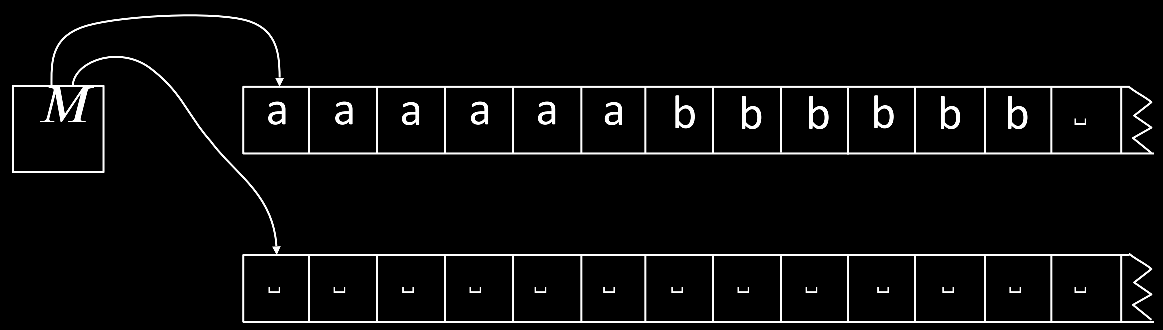


Deciding $A = \left\{ \mathrm{a}^k\mathrm{b}^k \middle|\ k \geq 0 \right\}$ even faster

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

$M =$ "On input w

1. Scan input to check if $w \in \mathrm{a}^*\mathrm{b}^*$, *reject* if not.
2. Copy a's to second tape.
3. Match b's with a's on second tape.

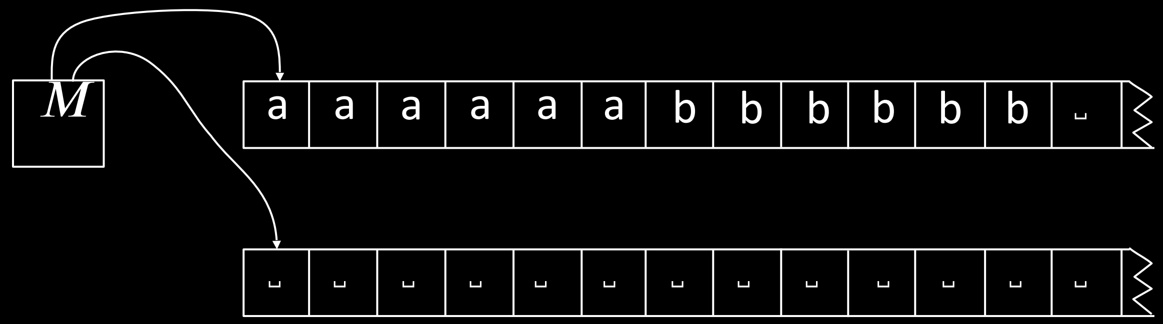


Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ even faster

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

$M =$ "On input w

1. Scan input to check if $w \in \mathrm{a}{*}\mathrm{b}^*$, *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*. "

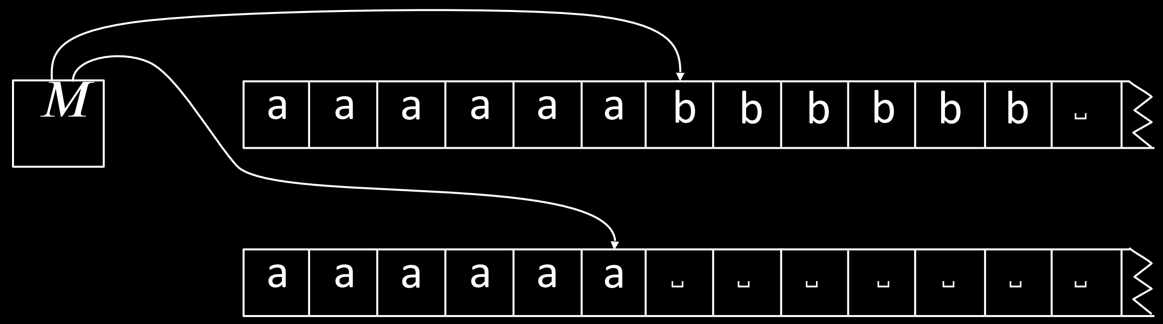


Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ even faster

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

$M =$ "On input w

1. Scan input to check if $w \in \mathrm{a}^*\mathrm{b}^*$, *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*. "

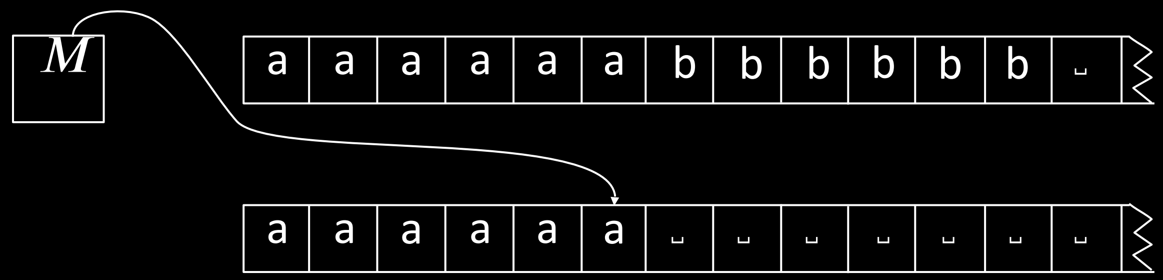


Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ even faster

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

$M =$ "On input w

1. Scan input to check if $w \in \mathrm{a}^*\mathrm{b}^*$, *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*. "

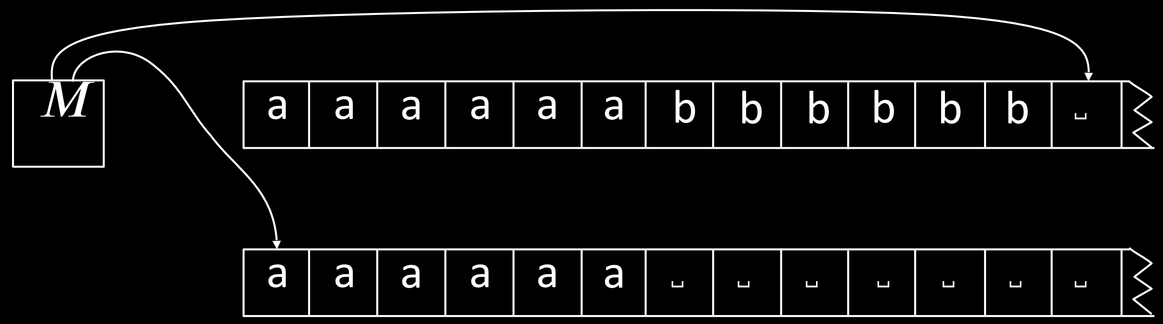


Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ even faster

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

$M =$ "On input w

1. Scan input to check if $w \in \mathrm{a}{*}\mathrm{b}^*$, *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*. "



Deciding $A = \left\{ \mathrm{a}^k \mathrm{b}^k \middle|\ k \geq 0 \right\}$ 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 $w \in \mathrm{a}{*}\mathrm{b}^*$, *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*. "

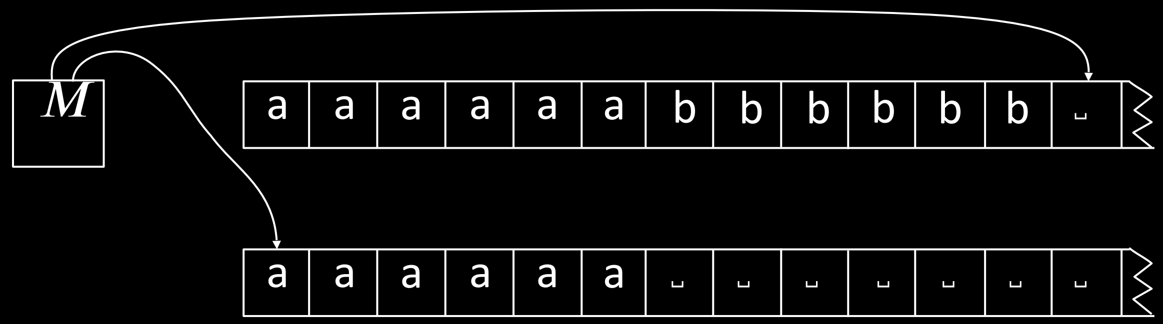
Analysis:

$O(n)$ steps

$+O(n)$ steps

$+O(n)$ steps

$= O(n)$ steps



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*."

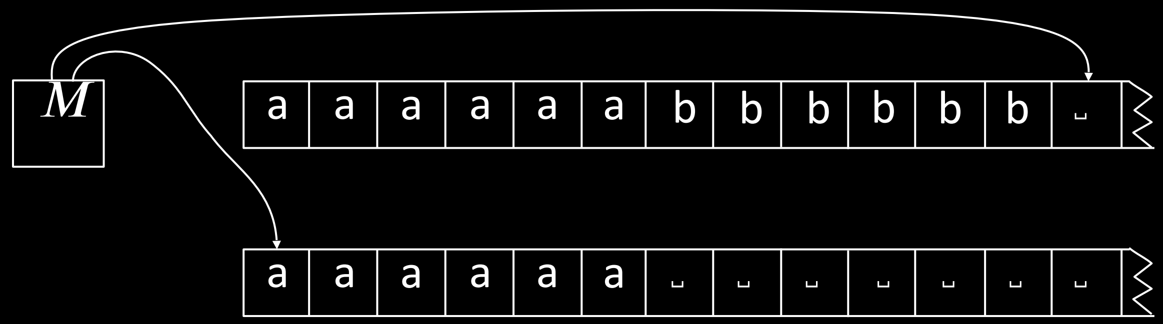
Analysis:

$O(n)$ steps

$+O(n)$ steps

$+O(n)$ steps

$= O(n)$ steps



Model Dependence

Number of steps to decide $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ depends on the model.

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

Model Dependence

Number of steps to decide $A = \left\{ \mathrm{a}^k \mathrm{b}^k \middle|\ k \geq 0 \right\}$ 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.

Model Dependence

Number of steps to decide $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ 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

Model Dependence

Number of steps to decide $A = \left\{ \mathrm{a}^k \mathrm{b}^k \middle| \ k \geq 0 \right\}$ 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.

Model Dependence

Number of steps to decide $A = \left\{ \mathrm{a}^k \mathrm{b}^k \,\middle|\, k \geq 0 \right\}$ 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.

Model Dependence

Number of steps to decide $A = \left\{ \text{a}^k\text{b}^k \,\middle|\, k \geq 0 \right\}$ 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.

Model Dependence

Number of steps to decide $A = \left\{ \mathrm{a}^k \mathrm{b}^k \middle|\ k \geq 0 \right\}$ 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

TIME Complexity Classes

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

TIME Complexity Classes

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

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

TIME Complexity Classes

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

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

Example:

$A = \left\{ \mathbf{a}^k \mathbf{b}^k \,\middle|\, k \geq 0 \right\} \in$ **TIME**$\big(n \log n\big)$

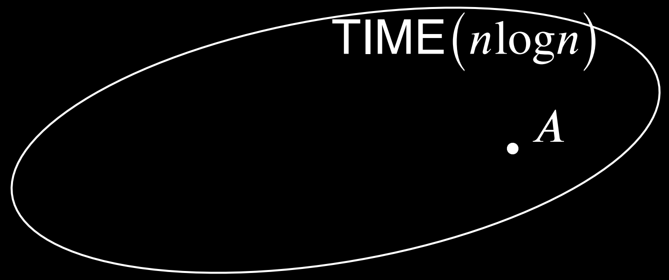
TIME Complexity Classes

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

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

Example:

$$A = \{\mathbf{a}^k\mathbf{b}^k \mid k \geq 0\} \in \textbf{TIME}(n\log n)$$



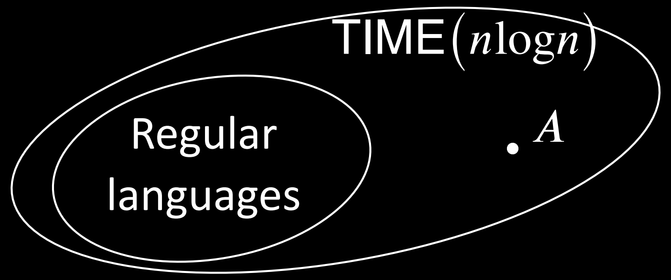
TIME Complexity Classes

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

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

Example:

$$A = \left\{\mathbf{a}^k\mathbf{b}^k \,\middle|\, k \geq 0\right\} \in \textbf{TIME}\big(n\log n\big)$$



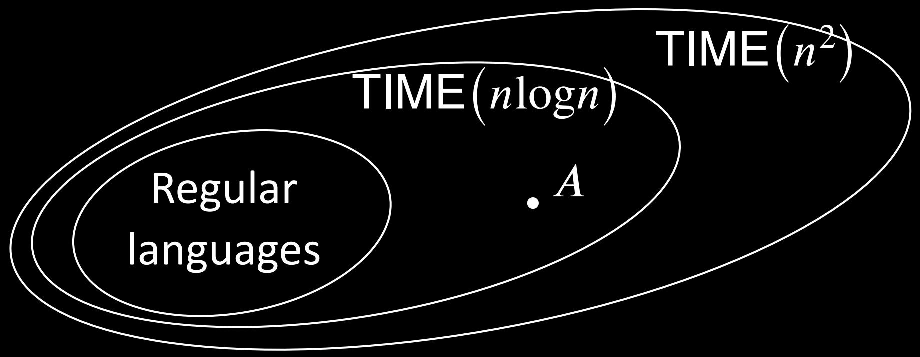
TIME Complexity Classes

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

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

Example:

$$A = \left\{ \mathbf{a}^k \mathbf{b}^k \,\middle|\, k \geq 0 \right\} \in \textbf{TIME}\big(n\log n\big)$$



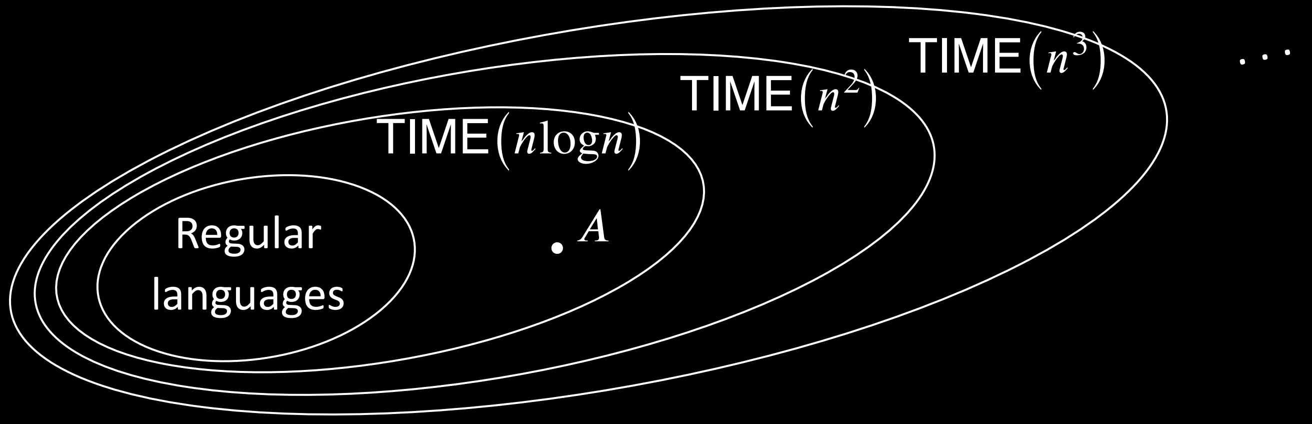
TIME Complexity Classes

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

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

Example:

$$A = \{\mathbf{a}^k\mathbf{b}^k \mid k \geq 0\} \in \textbf{TIME}\big(n\log n\big)$$



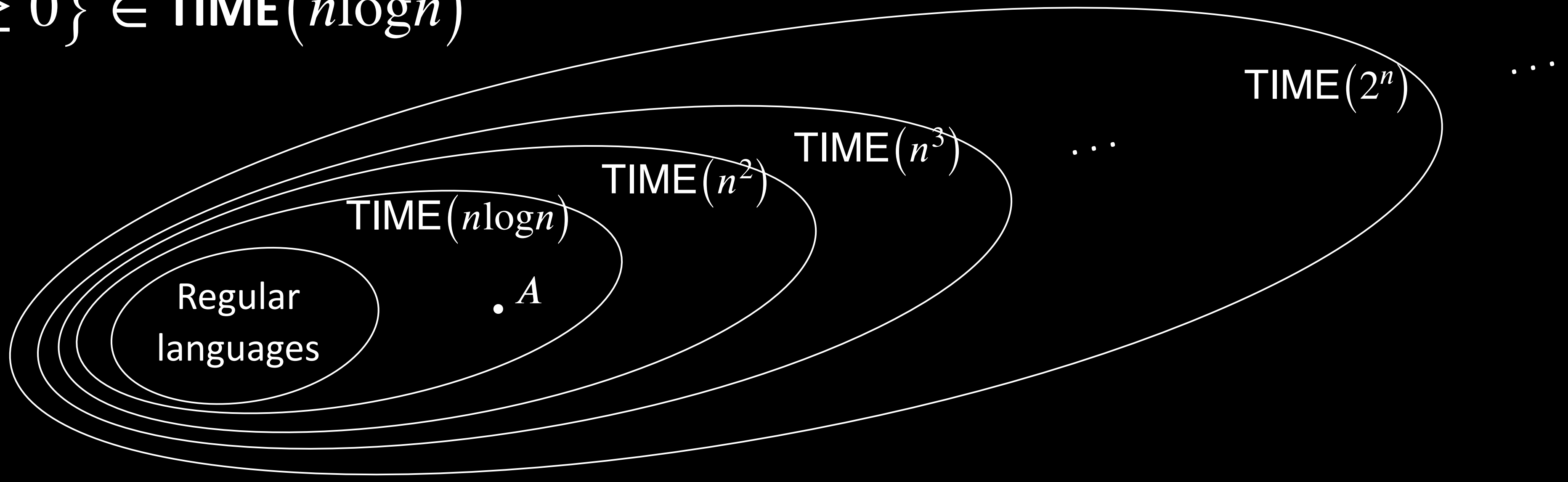
TIME Complexity Classes

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

Example:

$$A = \left\{\mathbf{a}^k\mathbf{b}^k \,\middle|\, k \geq 0\right\} \in \textbf{TIME}\big(n\log n\big)$$



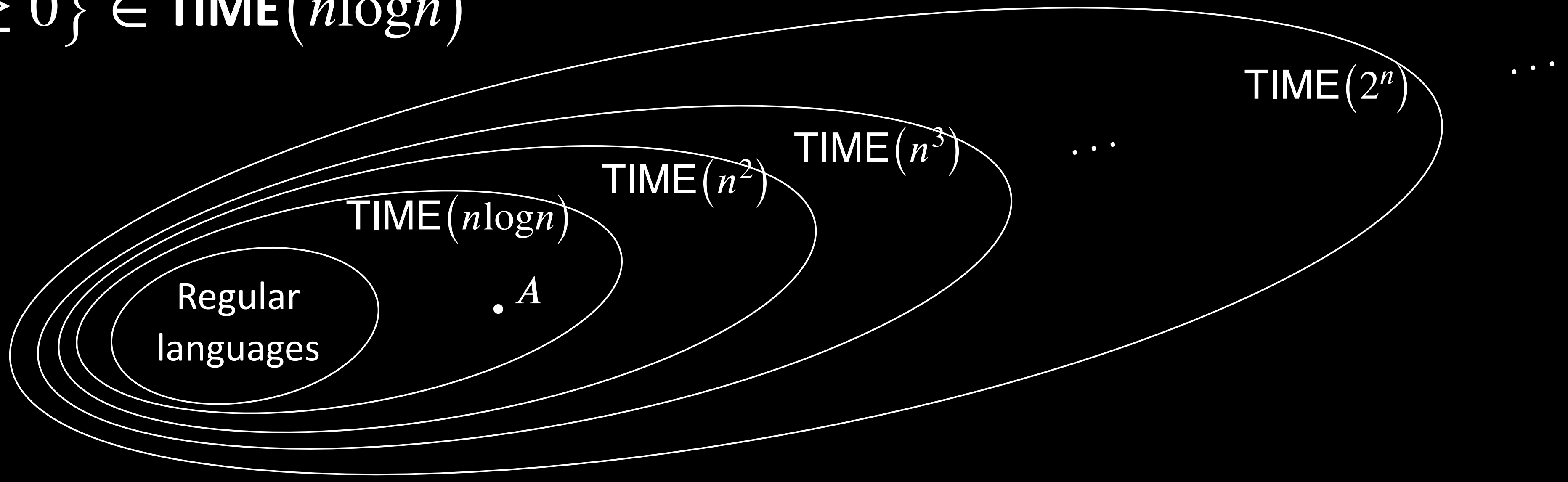
TIME Complexity Classes

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

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

Example:

$A = \left\{ \mathbf{a}^k\mathbf{b}^k \,\middle|\, k \geq 0 \right\} \in \textbf{TIME}\big(n\log n\big)$



Check-in 12.2

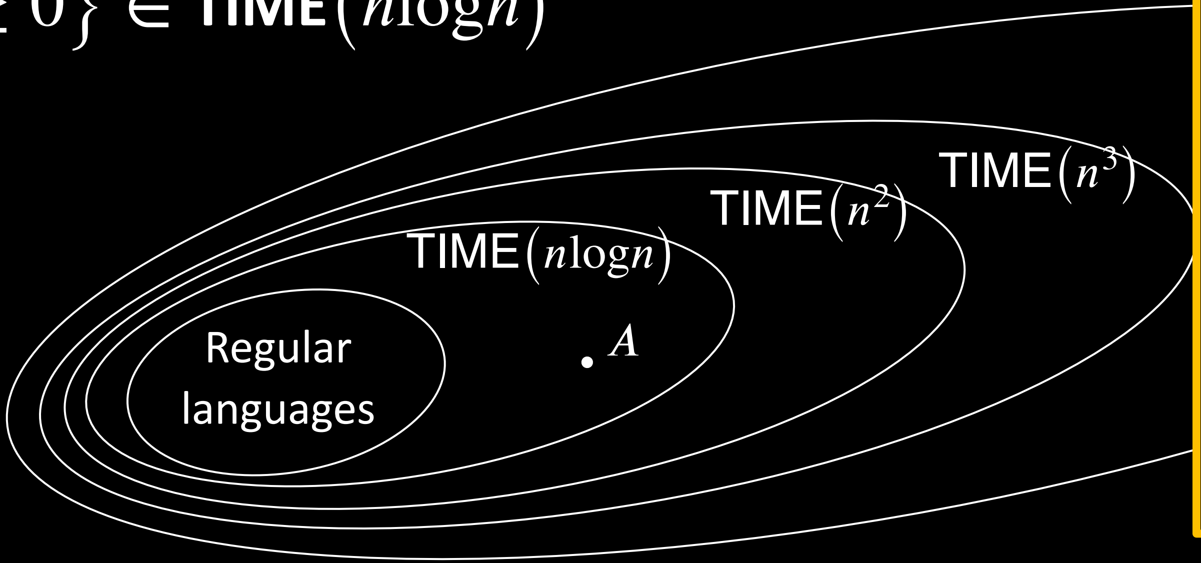
TIME Complexity Classes

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

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

Example:

$A = \{\mathbf{a}^k\mathbf{b}^k \mid k \geq 0\} \in \textbf{TIME}\big(n\log n\big)$



Check-in 12.2

Let

$B = \{\, ww^{\mathcal{R}} \mid w \in \{\mathrm{a}, \mathrm{b}\}^* \,\}$.

What is the smallest function t such that $B \in \text{TIME}\big(t(n)\big)$?

(a) $O(n)$

(b) $O\big(n\log n\big)$

(c) $O(n^2)$

(d) $O\big(n^3\big)$

Check-in 12.2

Multi-tape vs 1-tape time

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

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

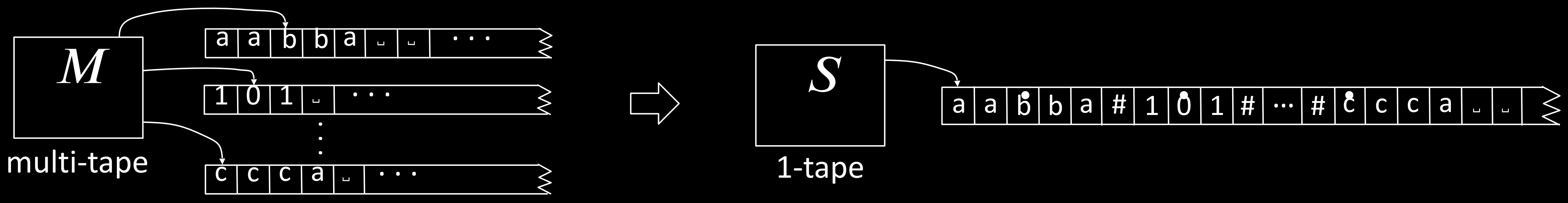
Proof: Analyze conversion of multi-tape to 1-tape TMs.

Multi-tape vs 1-tape time

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

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

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

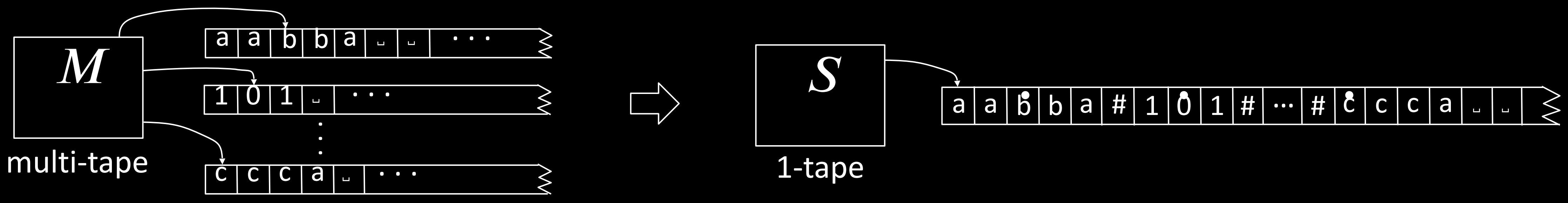


Multi-tape vs 1-tape time

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

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

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



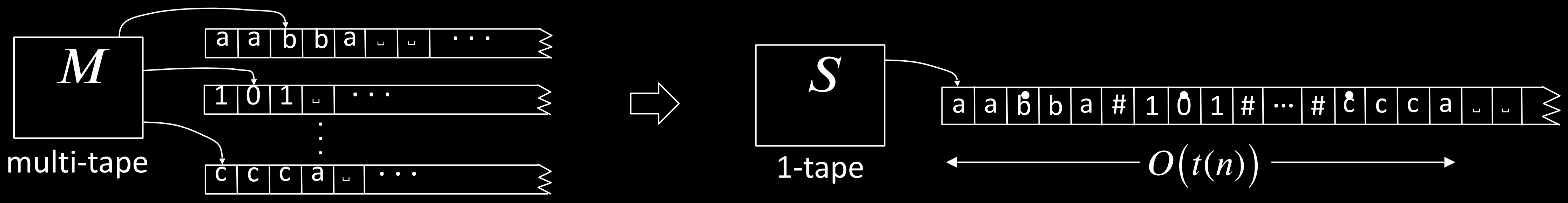
To simulate 1 step of M's computation, S uses $O\left(t(n)\right)$ steps.

Multi-tape vs 1-tape time

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

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

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



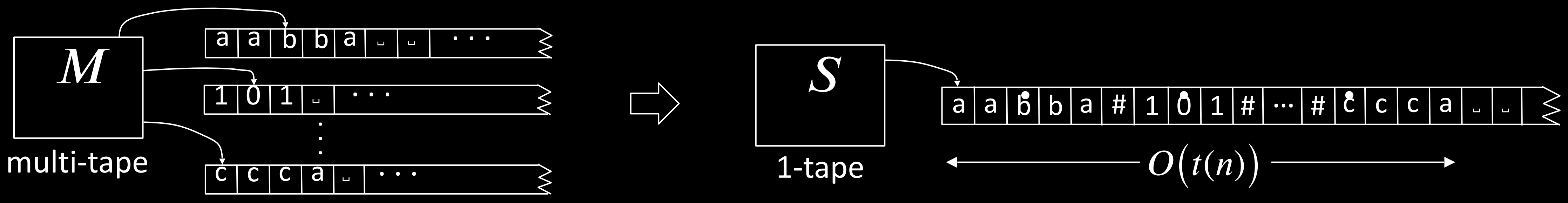
To simulate 1 step of M's computation, S uses $O\left(t(n)\right)$ steps.

Multi-tape vs 1-tape time

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

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

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



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

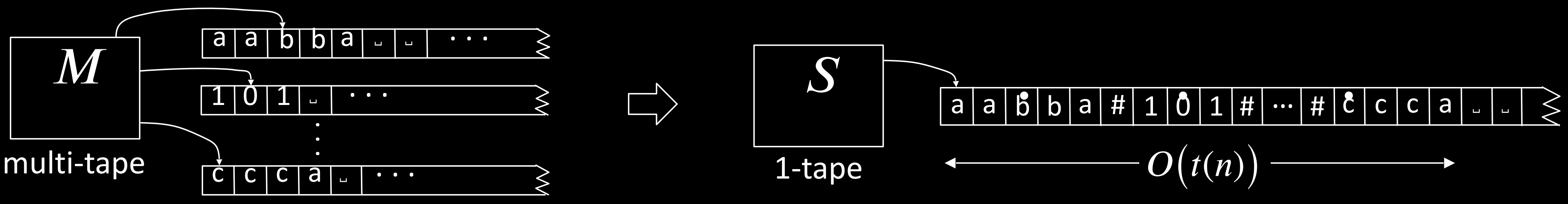
So total simulation time is $O\left(t(n) \times\ t(n)\right)\ =\ O\left(t^2(n)\right)$.

Multi-tape vs 1-tape time

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

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

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



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

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

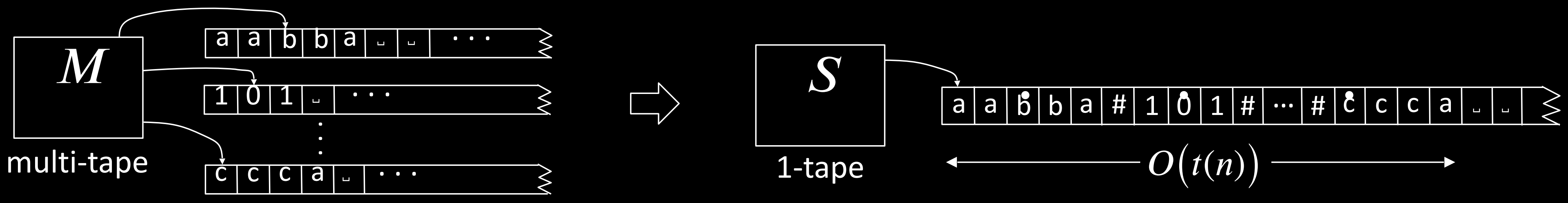
Similar results can be shown for other reasonable deterministic models.

Multi-tape vs 1-tape time

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

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

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



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

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

Similar results can be shown for other reasonable deterministic models.

Relationships among models

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

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

Relationships among models

Informal Defn: Two models of computation are <u>polynomially related</u> if each can simulate the other with a polynomial overhead:

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

All reasonable deterministic models are polynomially related.

Relationships among models

Informal Defn: Two models of computation are polynomially related if each can simulate the other with a polynomial overhead:
So $t(n)$ time $\rightarrow t^k(n)$ time on the other model, for some k.

All reasonable deterministic models are polynomially related.

- 1-tape TMs
- multi-tape TMs

Relationships among models

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

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

All reasonable deterministic models are polynomially related.

- 1-tape TMs
- multi-tape TMs
- multi-dimensional TMs

Relationships among models

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

So $t(n)$ time $\rightarrow t^k(n)$ 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)

Relationships among models

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

So $t(n)$ time $\to t^k(n)$ 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

Relationships among models

Informal Defn: Two models of computation are <u>polynomially related</u> if each can simulate the other with a polynomial overhead:

So $t(n)$ time $\rightarrow t^k(n)$ 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

The Class P

Defn: $\mathrm{P} = \bigcup_k \mathrm{TIME}(n^k)$

$= \text{polynomial time decidable languages}$

The Class P

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

$= \text{polynomial time decidable languages}$

- Invariant for all reasonable deterministic models

The Class P

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

$= \text{ polynomial time decidable languages}$

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

The Class P

Defn: $\mathrm{P} = \bigcup_k \mathrm{TIME}(n^k)$

$= $ polynomial time decidable languages

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

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

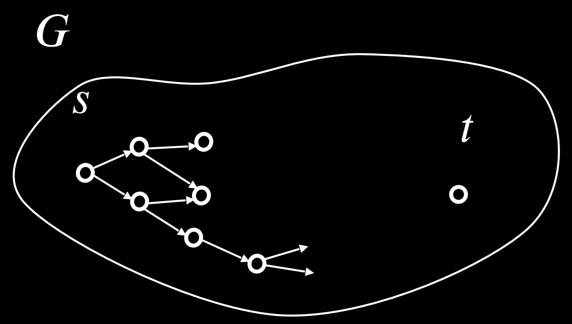
The Class P

Defn: $\mathrm{P} = \bigcup_k \mathrm{TIME}(n^k)$

$= $ polynomial time decidable languages

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

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



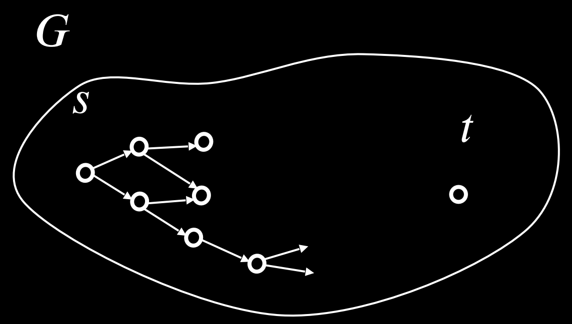
The Class P

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

$= \text{ polynomial time decidable languages}$

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

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



Theorem: $PATH \in$ **P**

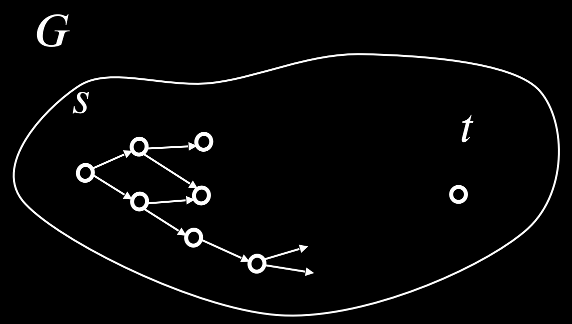
Proof: $M =$ "On input $\langle G, s, t \rangle$

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: $PATH = \left\{ \langle G, s, t \rangle \,\middle|\, G \text{ is a directed graph with a path from } s \text{ to } t \right\}$

Theorem: $PATH \in$ **P**

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

1. Mark s

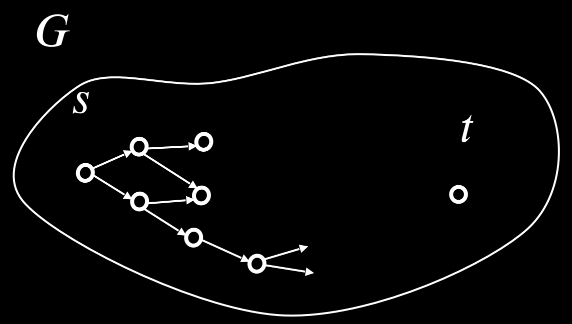
The Class P

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



Theorem: $PATH \in$ **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

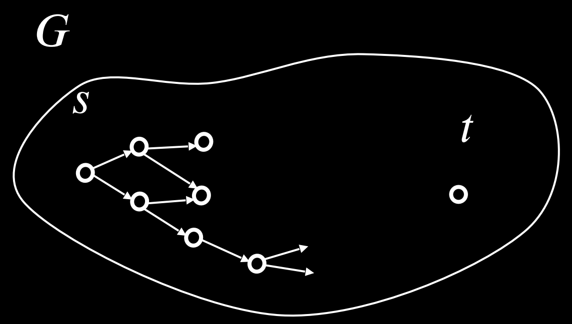
The Class P

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



Theorem: $PATH \in$ **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.

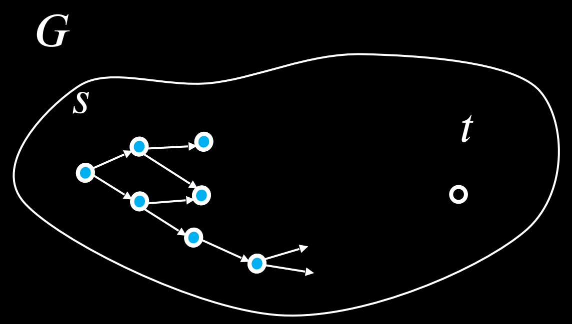
The Class P

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



Theorem: $PATH \in$ **P**

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

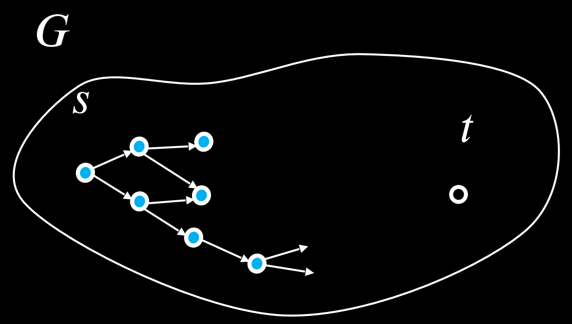
1. Mark s
2. Repeat until nothing new is marked:
   For each marked node x:
   Scan G to mark all y where (x, y) is an edge
3. *Accept* if t is marked. *Reject* if not.

The Class P

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

$= $ polynomial time decidable languages

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



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

Theorem: $PATH \in$ **P**

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

1. Mark s
2. Repeat until nothing new is marked: $\quad \leq n$ iterations
   For each marked node x: $\quad \times \quad \leq n$ iterations
   Scan G to mark all y where (x, y) is an edge $\quad \times \quad O(n^2)$ steps
3. *Accept* if t is marked. *Reject* if not.

$O(n^4)$ steps

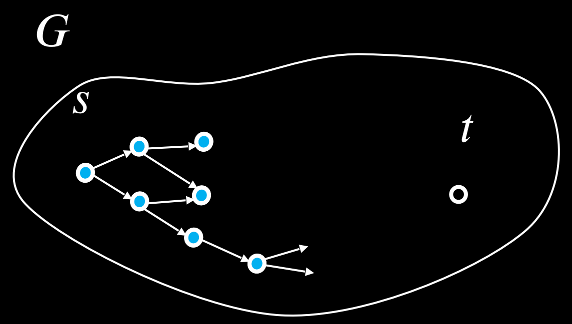
The Class P

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



Theorem: $PATH \in$ **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.

$\leq n$ iterations

$\times$ $\leq n$ iterations

$\times$ $O(n^2)$ steps

$O(n^4)$ steps

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

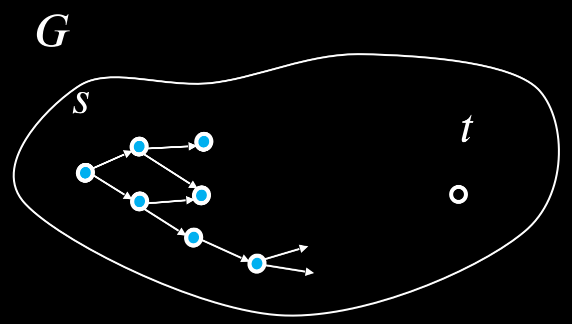
The Class P

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



Theorem: $PATH \in$ **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.

$\leq n$ iterations
$\times \quad \leq n$ iterations
$\times \quad O(n^2)$ steps

$O(n^4)$ steps

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

PATH and *HAMPATH*

$PATH$ and $HAMPATH$

Example: $HAMPATH = \Big\{ \langle G, s, t \rangle \,\Big|\,$ G is a directed graph with a path from s to t and the path goes through every node of G $\Big\}$

$PATH$ and $HAMPATH$

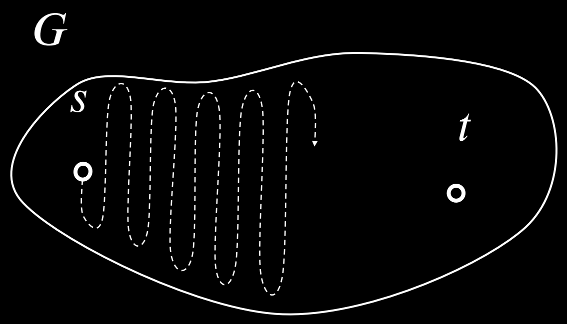
Example: $HAMPATH = \{\langle G, s, t\rangle \mid G$ is a directed graph with a path from s to t and $\underline{\text{the path goes through every node of}}$ G $\}$

Called a Hamiltonian path

$PATH$ and $HAMPATH$

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

Called a Hamiltonian path

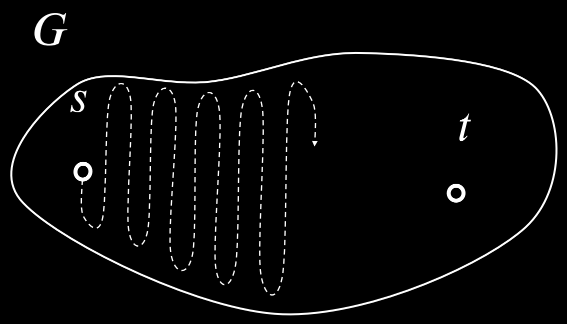


$PATH$ and $HAMPATH$

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

Called a Hamiltonian path

Recall Theorem: $PATH \in$ **P**



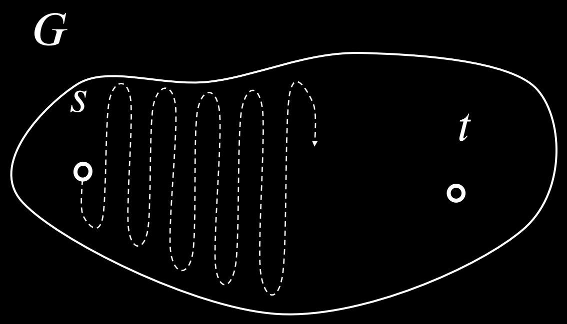
$PATH$ and $HAMPATH$

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

Called a Hamiltonian path

Recall Theorem: $PATH \in$ **P**

Question: $HAMPATH \in$ P ?



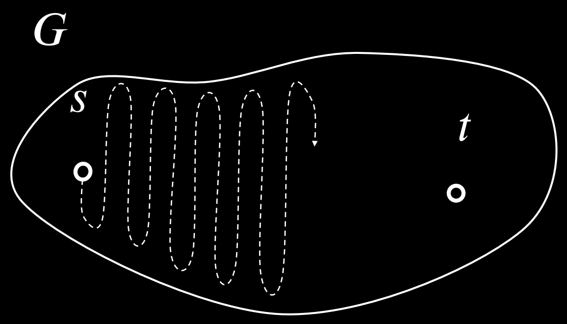
$PATH$ and $HAMPATH$

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

Called a Hamiltonian path

Recall Theorem: $PATH \in$ **P**

Question: $HAMPATH \in$ P ?



"On input $\langle G, s, t\rangle$

1. Let m be the number of nodes in G.

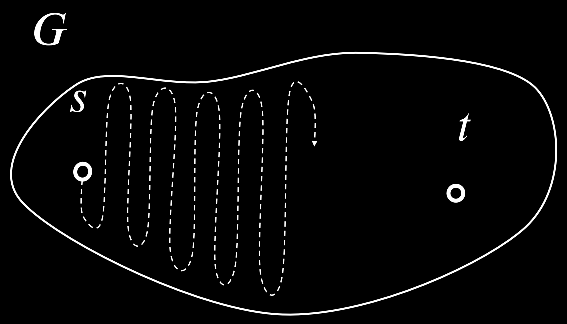
$PATH$ and $HAMPATH$

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

Called a Hamiltonian path

Recall Theorem: $PATH \in$ **P**

Question: $HAMPATH \in$ P ?



"On input $\langle G, s, t\rangle$

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

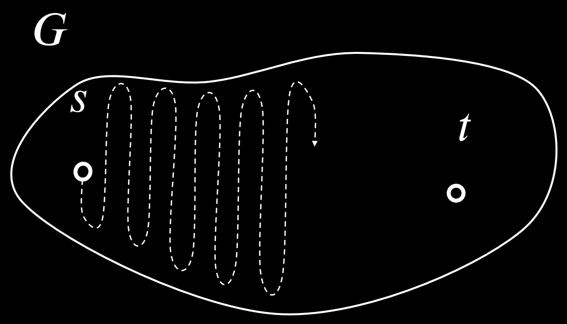
$PATH$ and $HAMPATH$

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

Called a Hamiltonian path

Recall Theorem: $PATH \in$ **P**

Question: $HAMPATH \in$ P ?



"On input $\langle G, s, t\rangle$

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

May be $m! > 2^m$ paths of length m
so algorithm is exponential time
not polynomial time.

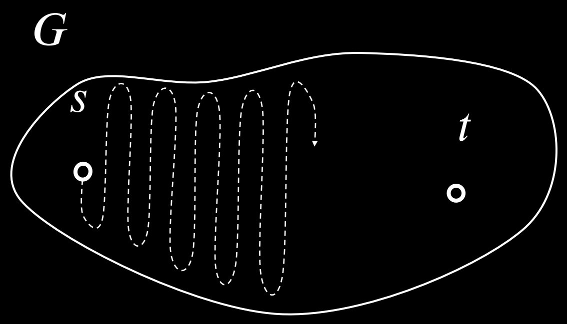
$PATH$ and $HAMPATH$

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

Called a Hamiltonian path

Recall Theorem: $PATH \in$ **P**

Question: $HAMPATH \in$ P ?



"On input $\langle G, s, t \rangle$

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

May be $m! > 2^m$ paths of length m
so algorithm is exponential time
not polynomial time.

Check-in 12.3

$PATH$ and $HAMPATH$

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

Called a Hamiltonian path

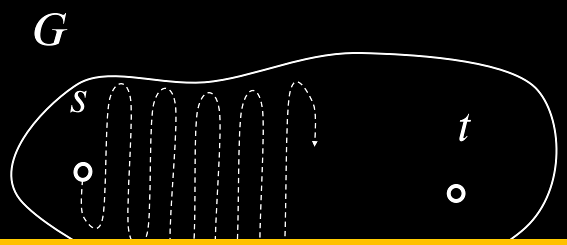
Recall Theorem: $PATH \in$ **P**

Question: $HAMPATH \in$ P ?

"On input $\langle G, s, t\rangle$

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

May be $m! > 2^m$ paths of length m
so algorithm is exponential time
not polynomial time.



Check-in 12.3

Is $HAMPATH \in$ P ?

(a) Definitely Yes. You have a polynomial-time algorithm.

(b) Probably Yes. It should be similar to showing $PATH \in$ P.

(c) Toss up.

(d) Probably No. Hard to beat the exponential algorithm.

(e) Definitely No. You can prove it!

Quick Review

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

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

$=$ polynomial time decidable languages

Quick Review

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

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

$=$ polynomial time decidable languages

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

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

Quick Review

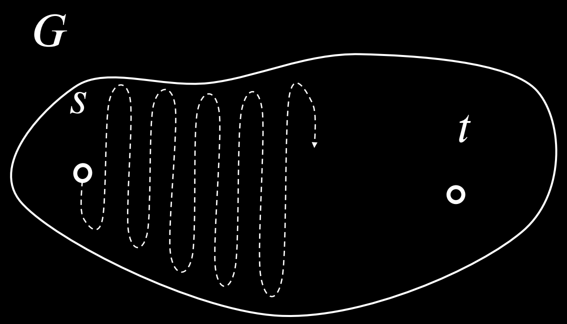
Defn: $\text{TIME}\big(t(n)\big) = \{B \mid$ some deterministic 1-tape TM M decides B and M runs in time $O\big(t(n)\big)\}$

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

$=$ polynomial time decidable languages

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

Theorem: $PATH \in$ **P**



$HAMPATH = \big\{\langle G, s, t\rangle \,\big|\, G$ is a directed graph with a path from s to t that goes through every node of $G\ \big\}$

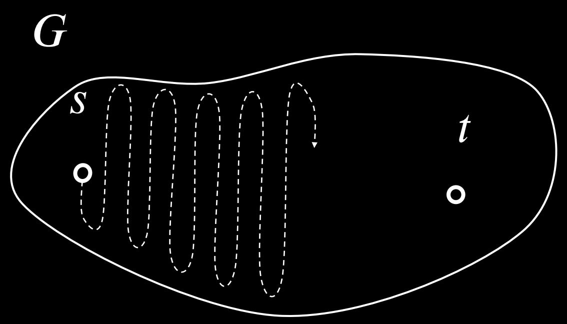
Quick Review

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

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

$= $ polynomial time decidable languages

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



Theorem: $PATH \in$ **P**

$HAMPATH = \big\{\langle G, s, t\rangle \big|\ G$ is a directed graph with a path from s to t that goes through every node of G $\}$

$HAMPATH \in$ **P** ?

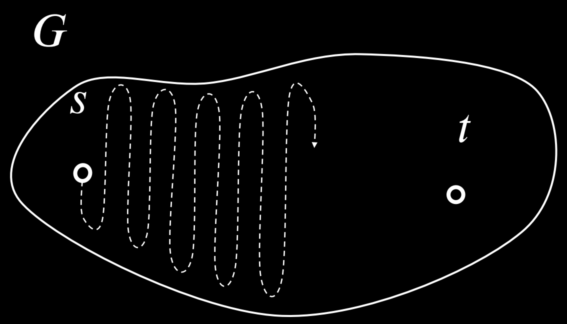
Quick Review

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

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

$=$ polynomial time decidable languages

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



Theorem: $PATH \in$ **P**

$HAMPATH = \{\langle G, s, t\rangle \mid G$ is a directed graph with a path from s to t that goes through every node of G $\}$

$HAMPATH \in$ **P** ? Unsolved Problem

[connection to factoring]

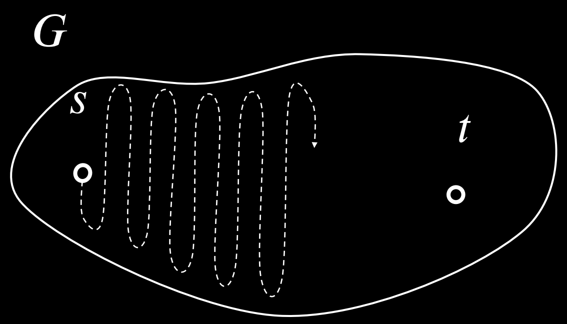
Quick Review

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

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

$=$ polynomial time decidable languages

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



Theorem: $PATH \in$ **P**

$HAMPATH = \big\{\langle G, s, t\rangle \big|\ G$ is a directed graph with a path from s to t that goes through every node of G $\}$

$HAMPATH \in$ **P** ? Unsolved Problem

[connection to factoring]

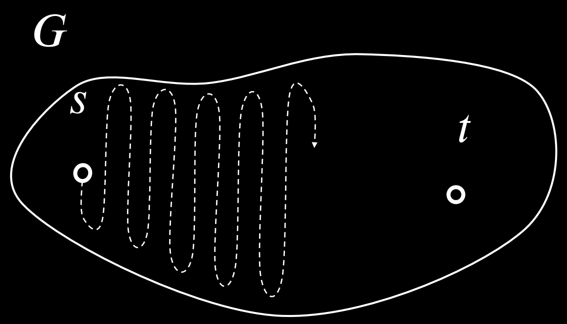
Quick Review

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

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

$=$ polynomial time decidable languages

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



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

$HAMPATH = \big\{\langle G, s, t\rangle \,\big|\, G$ is a directed graph with a path from s to t that goes through every node of $G\ \}$

$HAMPATH \in \mathbf{P}$? Unsolved Problem

[connection to factoring]

Nondeterministic Complexity

In a nondeterministic TM (NTM) decider, all branches halt on all inputs.

Nondeterministic Complexity

In a nondeterministic TM (NTM) decider, all branches halt on all inputs.

Defn: An NTM runs in time $t(n)$ if all branches halt within $t(n)$ steps on all inputs of length n.

Nondeterministic Complexity

In a nondeterministic TM (NTM) decider, all branches halt on all inputs.

Defn: An NTM <u>runs in time</u> $t(n)$ if all branches halt within $t(n)$ steps on all inputs of length n.

Defn: NTIME$\big(t(n)\big) = \{ B |$ some 1-tape NTM decides B and runs in time $O\big(t(n)\big)$ $\}$

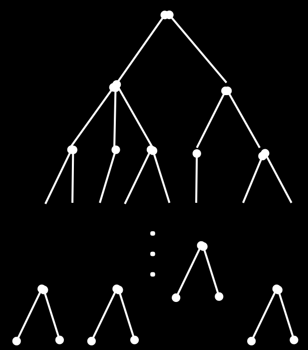
Nondeterministic Complexity

In a nondeterministic TM (NTM) decider, all branches halt on all inputs.

Defn: An NTM <u>runs in time</u> $t(n)$ if all branches halt within $t(n)$ steps on all inputs of length n.

Defn: NTIME$\big(t(n)\big) = \{\, B \mid$ some 1-tape NTM decides B and runs in time $O\big(t(n)\big)\,\}$

Computation tree for NTM on input w.



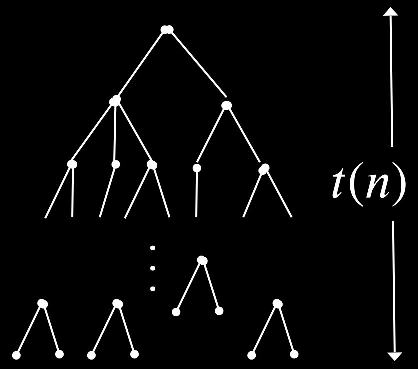
Nondeterministic Complexity

In a nondeterministic TM (NTM) decider, all branches halt on all inputs.

Defn: An NTM <u>runs in time</u> $t(n)$ if all branches halt within $t(n)$ steps on all inputs of length n.

Defn: $\text{NTIME}\big(t(n)\big) = \{ B |$ some 1-tape NTM decides B and runs in time $O\big(t(n)\big) \}$

Computation tree for NTM on input w.



<u>all branches halt</u>
within $t(n)$ steps

Nondeterministic Complexity

In a nondeterministic TM (NTM) decider, all branches halt on all inputs.

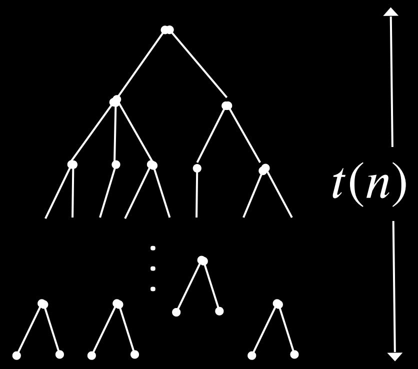
Defn: An NTM <u>runs in time</u> $t(n)$ if all branches halt within $t(n)$ steps on all inputs of length n.

Defn: $\text{NTIME}\big(t(n)\big) = \{ B \mid$ some 1-tape NTM decides B and runs in time $O\big(t(n)\big)$ $\}$

Defn: $\text{NP} = \bigcup_k \text{NTIME}(n^k)$

$=$ nondeterministic polynomial time decidable languages

Computation tree for NTM on input w.



<u>all branches halt</u>
within $t(n)$ steps

Nondeterministic Complexity

In a nondeterministic TM (NTM) decider, all branches halt on all inputs.

Defn: An NTM <u>runs in time</u> $t(n)$ if all branches halt within $t(n)$ steps on all inputs of length n.

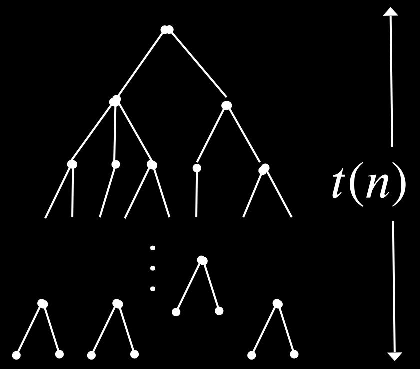
Defn: $\text{NTIME}\big(t(n)\big) = \{ B \mid$ some 1-tape NTM decides B and runs in time $O\big(t(n)\big)$ $\}$

Defn: **NP** $= \bigcup_k \text{NTIME}(n^k)$

$=$ nondeterministic polynomial time decidable languages

- Invariant for all reasonable nondeterministic models

Computation tree for NTM on input w.



<u>all branches halt</u> within $t(n)$ steps

Nondeterministic Complexity

In a nondeterministic TM (NTM) decider, all branches halt on all inputs.

Defn: An NTM <u>runs in time</u> $t(n)$ if all branches halt within $t(n)$ steps on all inputs of length n.

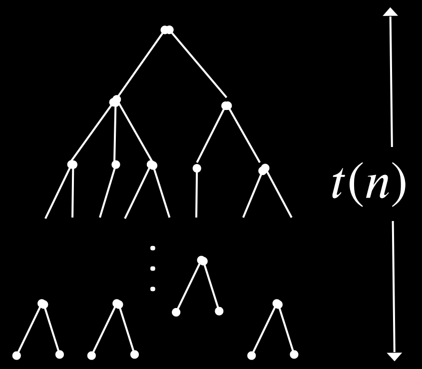
Defn: $\text{NTIME}\big(t(n)\big) = \{\, B \mid$ some 1-tape NTM decides B and runs in time $O\big(t(n)\big)\ \}$

Defn: $\textbf{NP} = \bigcup_k \text{NTIME}(n^k)$

$=$ nondeterministic polynomial time decidable languages

- Invariant for all reasonable nondeterministic models
- Corresponds roughly to easily verifiable problems

Computation tree for NTM on input w.



<u>all branches halt</u> within $t(n)$ steps

$HAMPATH \in$ NP

$HAMPATH \in$ NP

Theorem: $HAMPATH \in$ NP

Proof:

"On input $\langle G, s, t \rangle$ (Say G has m nodes.)

1. Nondeterministically write a sequence $v_1, v_2, \ldots, v_m$ of m nodes.
2. *Accept* if $v_1 = s$
   $v_m = t$
   each (v_i, v_{i+1}) is an edge
   and no v_i repeats.
3. *Reject* if any condition fails."

$HAMPATH \in$ NP

Computation of
M on $\langle G, s, t \rangle$

Theorem: $HAMPATH \in$ NP

Proof:

"On input $\langle G, s, t \rangle$ (Say G has m nodes.)

1. Nondeterministically write a sequence
   $v_1, v_2, \ldots, v_m$ of m nodes.
2. *Accept* if $v_1 = s$
   $v_m = t$
   each (v_i, v_{i+1}) is an edge
   and no v_i repeats.
3. *Reject* if any condition fails."

$HAMPATH \in$ NP

Computation of
M on $\langle G, s, t \rangle$

Theorem: $HAMPATH \in$ NP

Proof:

"On input $\langle G, s, t \rangle$ (Say G has m nodes.)

1. Nondeterministically write a sequence
   $v_1, v_2, \ldots, v_m$ of m nodes.
2. *Accept* if $v_1 = s$
   $v_m = t$
   each (v_i, v_{i+1}) is an edge
   and no v_i repeats.
3. *Reject* if any condition fails."

$HAMPATH \in$ NP

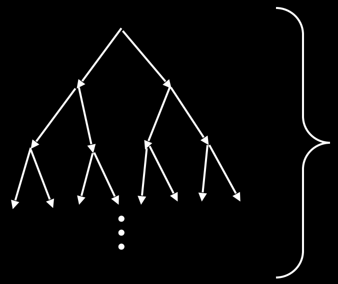
Theorem: $HAMPATH \in$ NP

Proof:

"On input $\langle G, s, t\rangle$ (Say G has m nodes.)

1. Nondeterministically write a sequence $v_1, v_2, \ldots, v_m$ of m nodes.
2. *Accept* if $v_1 = s$
   $v_m = t$
   each (v_i, v_{i+1}) is an edge
   and no v_i repeats.
3. *Reject* if any condition fails."

Computation of M on $\langle G, s, t\rangle$



Guess bits of v_1

$HAMPATH \in$ NP

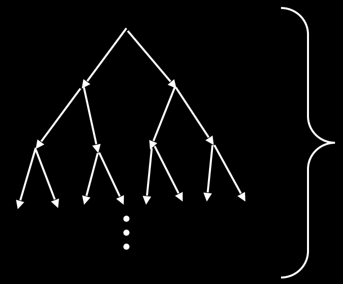
Theorem: $HAMPATH \in$ NP

Proof:

"On input $\langle G, s, t\rangle$ (Say G has m nodes.)

1. Nondeterministically write a sequence $v_1, v_2, \ldots, v_m$ of m nodes.
2. *Accept* if $v_1 = s$
   $v_m = t$
   each (v_i, v_{i+1}) is an edge
   and no v_i repeats.
3. *Reject* if any condition fails."

Computation of M on $\langle G, s, t\rangle$



Guess bits of v_1

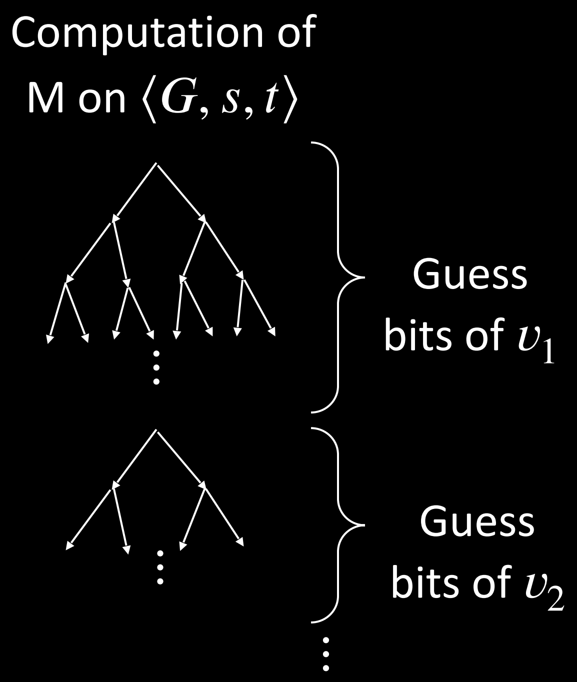
$HAMPATH \in$ NP

Theorem: $HAMPATH \in$ NP

Proof:

"On input $\langle G, s, t\rangle$ (Say G has m nodes.)

1. Nondeterministically write a sequence $v_1, v_2, \ldots, v_m$ of m nodes.
2. *Accept* if $v_1 = s$
   $v_m = t$
   each (v_i, v_{i+1}) is an edge
   and no v_i repeats.
3. *Reject* if any condition fails."



$HAMPATH \in$ NP

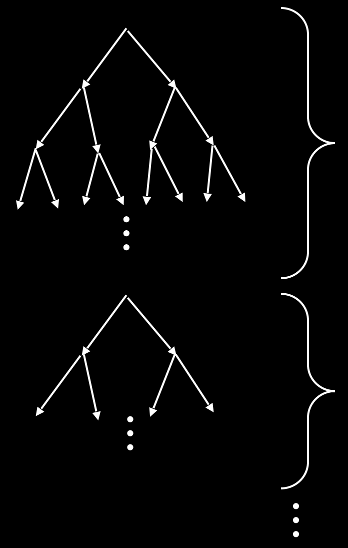
Theorem: $HAMPATH \in$ NP

Proof:

"On input $\langle G, s, t \rangle$ (Say G has m nodes.)

1. Nondeterministically write a sequence $v_1, v_2, \ldots, v_m$ of m nodes.
2. *Accept* if $v_1 = s$
   $v_m = t$
   each (v_i, v_{i+1}) is an edge
   and no v_i repeats.
3. *Reject* if any condition fails."

Computation of M on $\langle G, s, t \rangle$



Guess bits of v_1

Guess bits of v_2

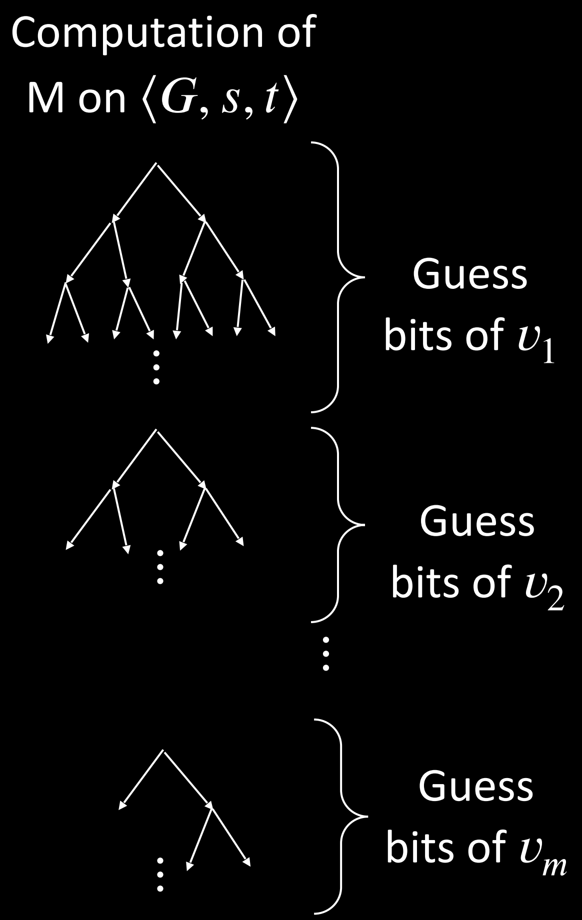
$HAMPATH \in$ NP

Theorem: $HAMPATH \in$ NP

Proof:

"On input $\langle G, s, t \rangle$ (Say G has m nodes.)

1. Nondeterministically write a sequence $v_1, v_2, \ldots, v_m$ of m nodes.
2. *Accept* if $v_1 = s$
   $v_m = t$
   each (v_i, v_{i+1}) is an edge
   and no v_i repeats.
3. *Reject* if any condition fails."



$HAMPATH \in$ NP

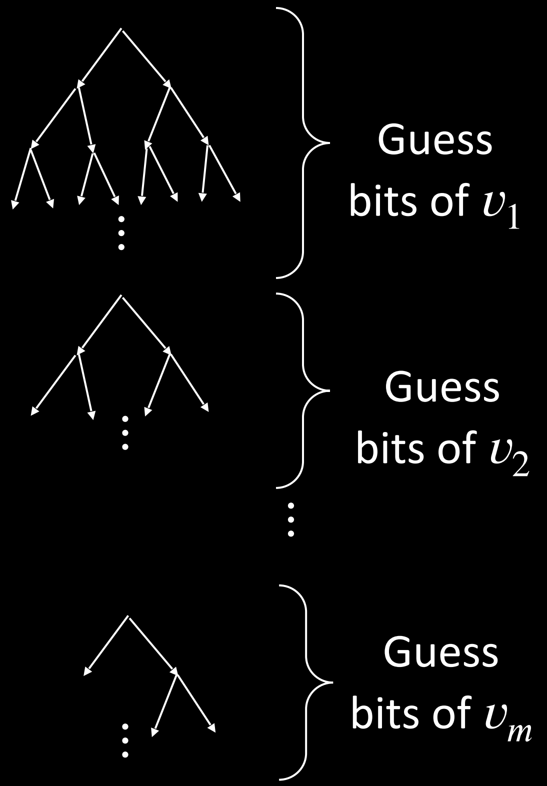
Theorem: $HAMPATH \in$ NP

Proof:

"On input $\langle G, s, t\rangle$ (Say G has m nodes.)

1. Nondeterministically write a sequence $v_1, v_2, \ldots, v_m$ of m nodes.
2. *Accept* if $v_1 = s$
   $v_m = t$
   each (v_i, v_{i+1}) is an edge
   and no v_i repeats.
3. *Reject* if any condition fails."

Computation of M on $\langle G, s, t\rangle$



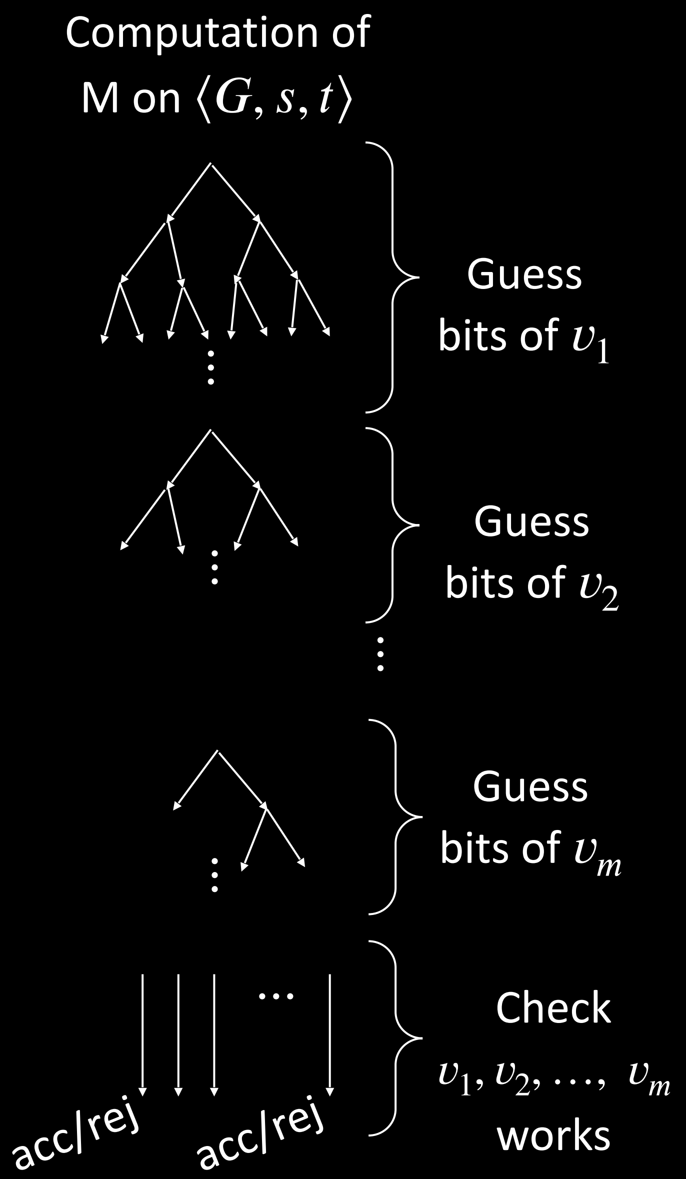
$HAMPATH \in$ NP

Theorem: $HAMPATH \in$ NP

Proof:

"On input $\langle G, s, t \rangle$ (Say G has m nodes.)

1. Nondeterministically write a sequence $v_1, v_2, \ldots, v_m$ of m nodes.
2. *Accept* if $v_1 = s$
   $v_m = t$
   each (v_i, v_{i+1}) is an edge
   and no v_i repeats.
3. *Reject* if any condition fails."



$COMPOSITES \in$ NP

Defn: $COMPOSITES = \{x \mid x$ is not prime and x is written in binary$\}$

$= \{x \mid x = yz$ for integers $y, z > 1,\ x$ in binary$\}$

$COMPOSITES \in$ NP

Defn: $COMPOSITES = \{x \mid x \text{ is not prime and } x \text{ is written in binary}\}$

$$= \{x \mid x = yz \text{ for integers } y, z > 1,\ x \text{ in binary}\}$$

Theorem: $COMPOSITES \in$ NP

$COMPOSITES \in$ NP

Defn: $COMPOSITES = \{x \mid x \text{ is not prime and } x \text{ is written in binary}\}$
$= \{x \mid x = yz \text{ for integers } y, z > 1,\ x \text{ in binary}\}$

Theorem: $COMPOSITES \in$ NP

Proof: "On input x

$COMPOSITES \in$ NP

Defn: $COMPOSITES = \{x \mid x$ is not prime and x is written in binary$\}$

$= \{x \mid x = yz$ for integers $y, z > 1,\ x$ in binary$\}$

Theorem: $COMPOSITES \in$ NP

Proof: “On input x

1. Nondeterministically write y where $1 < y < x$.

$COMPOSITES \in$ NP

Defn: $COMPOSITES = \{x \mid x \text{ is not prime and } x \text{ is written in binary}\}$

$= \{x \mid x = yz \text{ for integers } y, z > 1,\ x \text{ in binary}\}$

Theorem: $COMPOSITES \in$ NP

Proof: "On input x

1. Nondeterministically write y where $1 < y < x$.
2. *Accept* if y divides x with remainder 0.
   Reject if not."

$COMPOSITES \in$ NP

Defn: $COMPOSITES = \{x \mid x$ is not prime and x is written in binary$\}$

$= \{x \mid x = yz$ for integers $y, z > 1,\ x$ in binary$\}$

Theorem: $COMPOSITES \in$ NP

Proof: "On input x

1. Nondeterministically write y where $1 < y < x$.
2. *Accept* if y divides x with remainder 0.
   Reject if not."

Note: Using base 10 instead of base 2 wouldn't matter because can convert in polynomial time.

$COMPOSITES \in$ NP

Defn: $COMPOSITES = \{x \mid x$ is not prime and x is written in binary$\}$

$= \{x \mid x = yz$ for integers $y, z > 1,\ x$ in binary$\}$

Theorem: $COMPOSITES \in$ NP

Proof: "On input x

1. Nondeterministically write y where $1 < y < x$.
2. *Accept* if y divides x with remainder 0.
   Reject if not."

Note: Using base 10 instead of base 2 wouldn't matter because can convert in polynomial time.

Bad encoding: write number k in unary: $1^k = \overbrace{111\cdots1}^{k}$, exponentially longer.

$COMPOSITES \in$ NP

Defn: $COMPOSITES = \{x \mid x$ is not prime and x is written in binary$\}$

$= \{x \mid x = yz$ for integers $y, z > 1,\ x$ in binary$\}$

Theorem: $COMPOSITES \in$ NP

Proof: "On input x

1. Nondeterministically write y where $1 < y < x$.
2. *Accept* if y divides x with remainder 0.
   Reject if not."

Note: Using base 10 instead of base 2 wouldn't matter because can convert in polynomial time.

Bad encoding: write number k in unary: $1^k = \overbrace{111\cdots1}^{k}$, exponentially longer.

Theorem (2002): $COMPOSITES \in$ P

We won't cover this proof.

$COMPOSITES \in$ NP

Defn: $COMPOSITES = \{x \mid x \text{ is not prime and } x \text{ is written in binary}\}$

$= \{x \mid x = yz \text{ for integers } y, z > 1,\ x \text{ in binary}\}$

Theorem: $COMPOSITES \in$ NP

Proof: "On input x

1. Nondeterministically write y where $1 < y < x$.
2. *Accept* if y divides x with remainder 0.
   Reject if not."

Note: Using base 10 instead of base 2 wouldn't matter because can convert in polynomial time.

Bad encoding: write number k in unary: $1^k = \overbrace{111\cdots1}^{k}$, exponentially longer.

Theorem (2002): $COMPOSITES \in$ P

We won't cover this proof.

Intuition for P and NP

NP = All languages where can <u>verify</u> membership quickly
P = All languages where can <u>test</u> membership quickly

Intuition for P and NP

NP = All languages where can <u>verify</u> membership quickly
P = All languages where can <u>test</u> membership quickly

Examples of quickly verifying membership:

Intuition for P and NP

NP = All languages where can <u>verify</u> membership quickly
P = All languages where can <u>test</u> membership quickly

Examples of quickly verifying membership:

- $HAMPATH$: Give the Hamiltonian path.

Intuition for P and NP

NP = All languages where can <u>verify</u> membership quickly

P = All languages where can <u>test</u> membership quickly

Examples of quickly verifying membership:

- $HAMPATH$: Give the Hamiltonian path.
- $COMPOSITES$: Give the factor.

Intuition for P and NP

NP = All languages where can <u>verify</u> membership quickly
P = All languages where can <u>test</u> membership quickly

Examples of quickly verifying membership:

- $HAMPATH$: Give the Hamiltonian path.
- $COMPOSITES$: Give the factor.

The <u>Hamiltonian path</u> and the <u>factor</u> are called ***short certificates*** of membership.

Intuition for P and NP

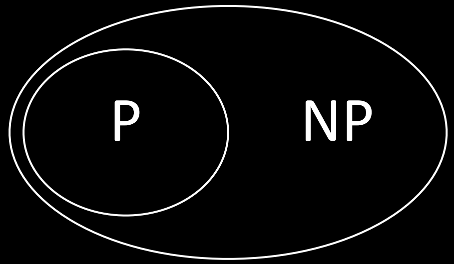
NP = All languages where can verify membership quickly
P = All languages where can test membership quickly

Examples of quickly verifying membership:

- $HAMPATH$: Give the Hamiltonian path.
- $COMPOSITES$: Give the factor.

The Hamiltonian path and the factor are called ***short certificates*** of membership.

P $\subseteq$ NP



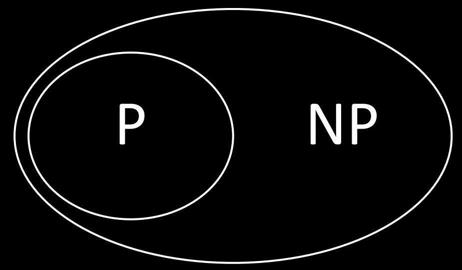
Intuition for P and NP

NP = All languages where can <u>verify</u> membership quickly
 P = All languages where can <u>test</u> membership quickly

Examples of quickly verifying membership:
- $HAMPATH$: Give the Hamiltonian path.
- $COMPOSITES$: Give the factor.

The <u>Hamiltonian path</u> and the <u>factor</u> are called ***short certificates*** of membership.

P $\subseteq$ NP
Question: P = NP?



Intuition for P and NP

NP = All languages where can verify membership quickly
P = All languages where can test membership quickly

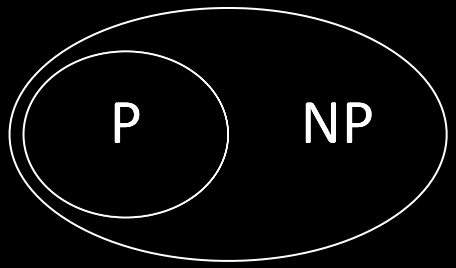
Examples of quickly verifying membership:

- $HAMPATH$: Give the Hamiltonian path.
- $COMPOSITES$: Give the factor.

The Hamiltonian path and the factor are called ***short certificates*** of membership.

P $\subseteq$ NP

Question: P = NP? Famous unsolved problem (Cook 1971).



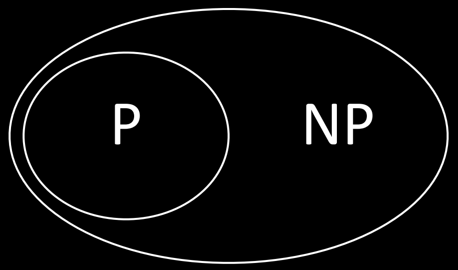
Intuition for P and NP

NP = All languages where can <u>verify</u> membership quickly
P = All languages where can <u>test</u> membership quickly

Examples of quickly verifying membership:
- $HAMPATH$: Give the Hamiltonian path.
- $COMPOSITES$: Give the factor.

The <u>Hamiltonian path</u> and the <u>factor</u> are called ***short certificates*** of membership.

P $\subseteq$ NP



Question: P = NP? Famous unsolved problem (Cook 1971).
Conjecture: P ≠ NP.

Intuition for P and NP

NP = All languages where can verify membership quickly
P = All languages where can test membership quickly

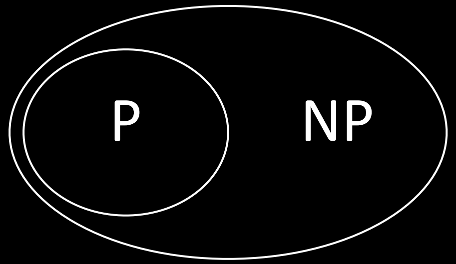
Examples of quickly verifying membership:
- $HAMPATH$: Give the Hamiltonian path.
- $COMPOSITES$: Give the factor.

The Hamiltonian path and the factor are called ***short certificates*** of membership.

P $\subseteq$ NP

Question: P = NP? Famous unsolved problem (Cook 1971).

Conjecture: P ≠ NP. Some problems are NP and not in P.



Intuition for P and NP

NP = All languages where can verify membership quickly
P = All languages where can test membership quickly

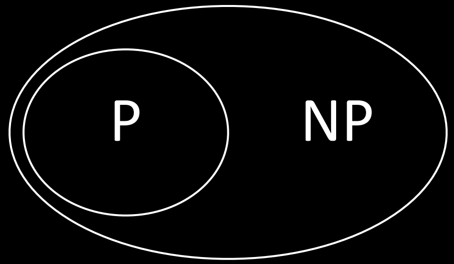
Examples of quickly verifying membership:
- $HAMPATH$: Give the Hamiltonian path.
- $COMPOSITES$: Give the factor.

The Hamiltonian path and the factor are called ***short certificates*** of membership.

P $\subseteq$ NP

Question: P = NP? Famous unsolved problem (Cook 1971).

Conjecture: P ≠ NP. Some problems are NP and not in P.



Hard to prove the conjecture because polynomial-time algorithms are powerful.

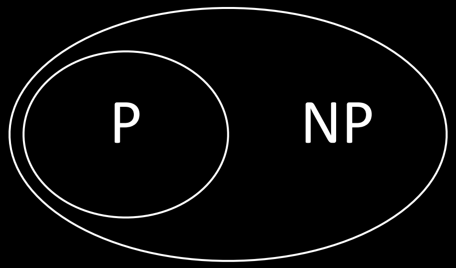
Intuition for P and NP

NP = All languages where can verify membership quickly
P = All languages where can test membership quickly

Examples of quickly verifying membership:
- $HAMPATH$: Give the Hamiltonian path.
- $COMPOSITES$: Give the factor.

The Hamiltonian path and the factor are called ***short certificates*** of membership.

$\mathrm{P} \subseteq \mathrm{NP}$



Question: P = NP? Famous unsolved problem (Cook 1971).
Conjecture: P ≠ NP. Some problems are NP and not in P.

Hard to prove the conjecture because polynomial-time algorithms are powerful.

Example: Show ACFG $\in$ P.

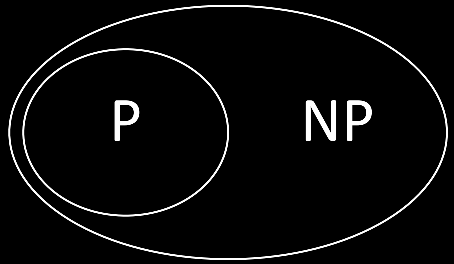
Intuition for P and NP

NP = All languages where can verify membership quickly
P = All languages where can test membership quickly

Examples of quickly verifying membership:
- $HAMPATH$: Give the Hamiltonian path.
- $COMPOSITES$: Give the factor.

The Hamiltonian path and the factor are called ***short certificates*** of membership.

P $\subseteq$ NP



Question: P = NP? Famous unsolved problem (Cook 1971).
Conjecture: P ≠ NP. Some problems are NP and not in P.

Hard to prove the conjecture because polynomial-time algorithms are powerful.

Example: Show ACFG $\in$ P.

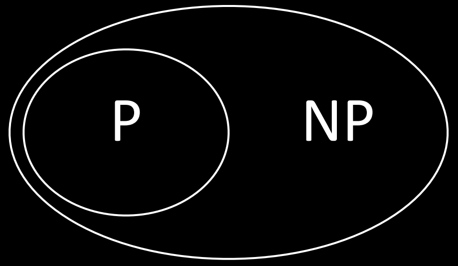
Check-in 14.1

Intuition for P and NP

NP = All languages where can <u>verify</u> membership quickly
P = All languages where can <u>test</u> membership quickly

Examples of quickly verifying membership:
- $HAMPATH$: Give the Hamiltonian path.
- $COMPOSITES$: Give the factor.

The <u>Hamiltonian path</u> and the <u>factor</u> are called ***short certificates*** of membership.



Check-in 14.1

Let $\overline{HAMPATH}$ be the complement of $HAMPATH$.

So $\langle G, s, t\rangle \in \overline{HAMPATH}$ if G does <u>not</u> have a Hamiltonian path from s to t.

Is $\overline{HAMPATH} \in$ NP?

(a) Yes, we can invert the accept/reject output of the NTM for $HAMPATH$.

(b) No, we cannot give a short certificate for a graph not to have a Hamiltonian path.

(c) I don't know.