

بسم الله الرحمن الرحيم

# نظريه علوم کامپیوتر

نظريه علوم کامپیوتر - بهار ۱۴۰۰-۱۴۰۱ - جلسه دوازدهم: پیچیدگی حافظه

Theory of computation - 002 - S12 - space complexity

# SPACE Complexity

**Defn:** Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  where  $f(n) \geq n$ . Say TM  $M$  runs in space  $f(n)$  if  $M$  always halts and uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

An NTM  $M$  runs in space  $f(n)$  if all branches halt and each branch uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

**Defn:**  $\text{SPACE}(f(n)) = \{B \mid \text{some deterministic 1-tape TM } M \text{ decides } B$   
and  $M \text{ runs in space } O(f(n))\}$

$\text{NSPACE}(f(n)) = \{B \mid \text{some nondeterministic 1-tape TM } M \text{ decides } B$   
and  $M \text{ runs in space } O(f(n))\}$

$\text{PSPACE} = \bigcup_k \text{SPACE}(n^k)$  “polynomial space”

$\text{NPSPACE} = \bigcup_k \text{NSPACE}(n^k)$  “nondeterministic polynomial space”

# SPACE Complexity

**Defn:** Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  where  $f(n) \geq n$ . Say TM  $M$  runs in space  $f(n)$  if  $M$  always halts and uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

An NTM  $M$  runs in space  $f(n)$  if all branches halt and each branch uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

**Defn:**  $\text{SPACE}(f(n)) = \{B \mid \text{some deterministic 1-tape TM } M \text{ decides } B$   
and  $M \text{ runs in space } O(f(n))\}$

$\text{NSPACE}(f(n)) = \{B \mid \text{some nondeterministic 1-tape TM } M \text{ decides } B$   
and  $M \text{ runs in space } O(f(n))\}$

$\text{PSPACE} = \bigcup_k \text{SPACE}(n^k)$  “polynomial space”

$\text{NPSPACE} = \bigcup_k \text{NSPACE}(n^k)$  “nondeterministic polynomial space”

# SPACE Complexity

**Defn:** Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  where  $f(n) \geq n$ . Say TM  $M$  runs in space  $f(n)$  if  $M$  always halts and uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

## Check-in 17.1

We define space complexity for multi-tape TMs by taking the sum of the cells used on all tapes.

Do we get the same class PSPACE for multi-tape TMs?

- (a) No.
- (b) Yes, converting a multi-tape TM to single-tape only squares the amount of space used.
- (c) Yes, converting a multi-tape TM to single-tape only increases the amount of space used by a constant factor.

# Relationships between Time and SPACE Complexity

**Theorem:** For  $t(n) \geq n$

1)  $\text{TIME}(t(n)) \subseteq \text{SPACE}(t(n))$

# Relationships between Time and SPACE Complexity

**Theorem:** For  $t(n) \geq n$

- 1)  $\text{TIME}(t(n)) \subseteq \text{SPACE}(t(n))$
- 2)  $\text{SPACE}(t(n)) \subseteq \text{TIME}\left(2^{O(t(n))}\right)$   
 $= \bigcup_c \text{TIME}(c^{t(n)})$

# Relationships between Time and SPACE Complexity

**Theorem:** For  $t(n) \geq n$

- 1)  $\text{TIME}(t(n)) \subseteq \text{SPACE}(t(n))$
- 2)  $\text{SPACE}(t(n)) \subseteq \text{TIME}\left(2^{O(t(n))}\right)$   
 $= \bigcup_c \text{TIME}(c^{t(n)})$

Proof:

# Relationships between Time and SPACE Complexity

**Theorem:** For  $t(n) \geq n$

- 1)  $\text{TIME}(t(n)) \subseteq \text{SPACE}(t(n))$
- 2)  $\text{SPACE}(t(n)) \subseteq \text{TIME}\left(2^{O(t(n))}\right)$   
 $= \bigcup_c \text{TIME}(c^{t(n)})$

Proof:

- 1) A TM that runs in  $t(n)$  steps cannot use more than  $t(n)$  tape cells.



# Relationships between Time and SPACE Complexity

**Theorem:** For  $t(n) \geq n$

- 1)  $\text{TIME}(t(n)) \subseteq \text{SPACE}(t(n))$
- 2)  $\text{SPACE}(t(n)) \subseteq \text{TIME}\left(2^{O(t(n))}\right)$   
 $= \bigcup_c \text{TIME}(c^{t(n)})$

**Proof:**

- 1) A TM that runs in  $t(n)$  steps cannot use more than  $t(n)$  tape cells.
- 2) A TM that uses  $t(n)$  tape cells cannot use more than  $c^{t(n)}$  time without repeating a configuration and looping (for some  $c$ ).

# Relationships between Time and SPACE Complexity

**Theorem:** For  $t(n) \geq n$

- 1)  $\text{TIME}(t(n)) \subseteq \text{SPACE}(t(n))$
- 2)  $\text{SPACE}(t(n)) \subseteq \text{TIME}\left(2^{O(t(n))}\right)$   
 $= \bigcup_c \text{TIME}(c^{t(n)})$

Proof:

- 1) A TM that runs in  $t(n)$  steps cannot use more than  $t(n)$  tape cells.
- 2) A TM that uses  $t(n)$  tape cells cannot use more than  $c^{t(n)}$  time without repeating a configuration and looping (for some  $c$ ).

Corollary:  $P \subseteq PSPACE$

# Relationships between Time and SPACE Complexity

**Theorem:** For  $t(n) \geq n$

- 1)  $\text{TIME}(t(n)) \subseteq \text{SPACE}(t(n))$
- 2)  $\text{SPACE}(t(n)) \subseteq \text{TIME}\left(2^{O(t(n))}\right)$   
 $= \bigcup_c \text{TIME}(c^{t(n)})$

**Proof:**

- 1) A TM that runs in  $t(n)$  steps cannot use more than  $t(n)$  tape cells.
- 2) A TM that uses  $t(n)$  tape cells cannot use more than  $c^{t(n)}$  time without repeating a configuration and looping (for some  $c$ ).

**Corollary:**  $P \subseteq PSPACE$

**Theorem:**  $NP \subseteq PSPACE$  [next slide]

# Relationships between Time and SPACE Complexity

**Theorem:** For  $t(n) \geq n$

- 1)  $\text{TIME}(t(n)) \subseteq \text{SPACE}(t(n))$
- 2)  $\text{SPACE}(t(n)) \subseteq \text{TIME}\left(2^{O(t(n))}\right)$   
 $= \bigcup_c \text{TIME}(c^{t(n)})$

Proof:

- 1) A TM that runs in  $t(n)$  steps cannot use more than  $t(n)$  tape cells.
- 2) A TM that uses  $t(n)$  tape cells cannot use more than  $c^{t(n)}$  time without repeating a configuration and looping (for some  $c$ ).

Corollary:  $P \subseteq PSPACE$

Theorem:  $NP \subseteq PSPACE$  [next slide]

$$\text{NP} \subseteq \text{PSPACE}$$

**Theorem:**  $\text{NP} \subseteq \text{PSPACE}$

$$\text{NP} \subseteq \text{PSPACE}$$

**Theorem:**  $\text{NP} \subseteq \text{PSPACE}$

Proof:

1.  $\text{SAT} \in \text{PSPACE}$

$$\text{NP} \subseteq \text{PSPACE}$$

**Theorem:**  $\text{NP} \subseteq \text{PSPACE}$

Proof:

1.  $\text{SAT} \in \text{PSPACE}$
2. If  $A \leq_p B$  and  $B \in \text{PSPACE}$  then  $A \in \text{PSPACE}$

$$\text{NP} \subseteq \text{PSPACE}$$

**Theorem:**  $\text{NP} \subseteq \text{PSPACE}$

Proof:

1.  $\text{SAT} \in \text{PSPACE}$
2. If  $A \leq_p B$  and  $B \in \text{PSPACE}$  then  $A \in \text{PSPACE}$

**Defn:**  $\text{coNP} = \{\overline{A} \mid A \in \text{NP}\}$



$$\text{NP} \subseteq \text{PSPACE}$$

**Theorem:**  $\text{NP} \subseteq \text{PSPACE}$

Proof:

1.  $\text{SAT} \in \text{PSPACE}$
2. If  $A \leq_p B$  and  $B \in \text{PSPACE}$  then  $A \in \text{PSPACE}$

**Defn:**  $\text{coNP} = \{\overline{A} \mid A \in \text{NP}\}$

$\overline{\text{HAMPATH}} \in \text{coNP}$

# NP $\subseteq$ PSPACE

**Theorem:** NP  $\subseteq$  PSPACE

Proof:

1.  $SAT \in \text{PSPACE}$
2. If  $A \leq_p B$  and  $B \in \text{PSPACE}$  then  $A \in \text{PSPACE}$

**Defn:**  $\text{coNP} = \{ \overline{A} \mid A \in \text{NP} \}$

$\overline{HAMPATH} \in \text{coNP}$

$TAUTOLOGY = \{ \langle \phi \rangle \mid \text{all assignments satisfy } \phi \} \in \text{coNP}$

# NP $\subseteq$ PSPACE

**Theorem:** NP  $\subseteq$  PSPACE

Proof:

1.  $SAT \in \text{PSPACE}$
2. If  $A \leq_p B$  and  $B \in \text{PSPACE}$  then  $A \in \text{PSPACE}$

**Defn:**  $\text{coNP} = \{\overline{A} \mid A \in \text{NP}\}$

$\overline{HAMPATH} \in \text{coNP}$

$TAUTOLOGY = \{\langle \phi \rangle \mid \text{all assignments satisfy } \phi\} \in \text{coNP}$

$\text{coNP} \subseteq \text{PSPACE}$  (because  $\text{PSPACE} = \text{coPSPACE}$ )

# $NP \subseteq PSPACE$

**Theorem:**  $NP \subseteq PSPACE$

Proof:

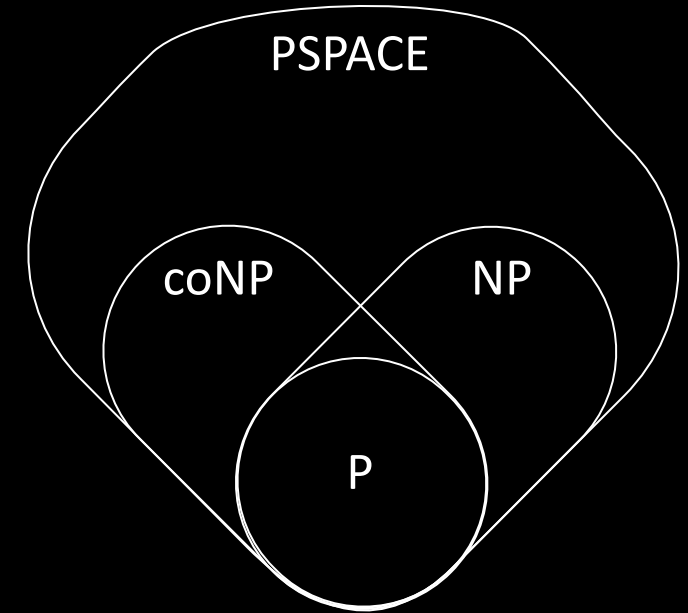
1.  $SAT \in PSPACE$
2. If  $A \leq_p B$  and  $B \in PSPACE$  then  $A \in PSPACE$

**Defn:**  $coNP = \{\overline{A} \mid A \in NP\}$

$\overline{HAMPATH} \in coNP$

$TAUTOLOGY = \{\langle \phi \rangle \mid \text{all assignments satisfy } \phi\} \in coNP$

$coNP \subseteq PSPACE$  (because  $PSPACE = coPSPACE$ )



# $NP \subseteq PSPACE$

**Theorem:**  $NP \subseteq PSPACE$

Proof:

1.  $SAT \in PSPACE$
2. If  $A \leq_p B$  and  $B \in PSPACE$  then  $A \in PSPACE$

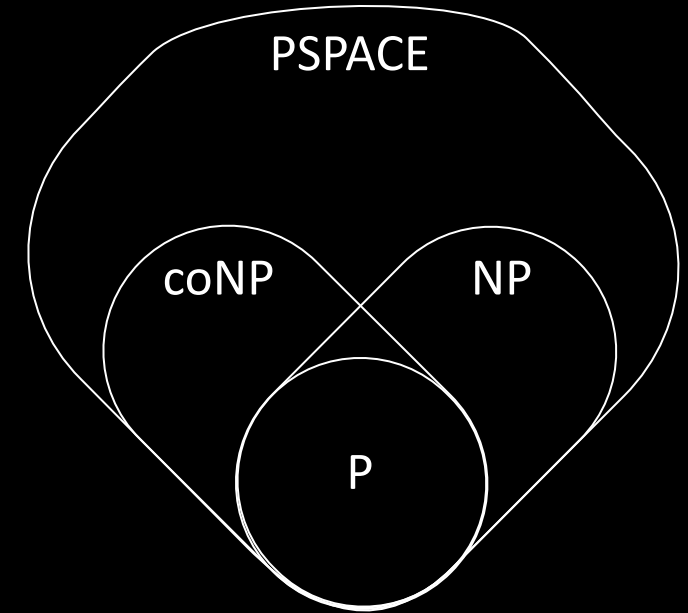
**Defn:**  $coNP = \{\overline{A} \mid A \in NP\}$

$\overline{HAMPATH} \in coNP$

$TAUTOLOGY = \{\langle \phi \rangle \mid \text{all assignments satisfy } \phi\} \in coNP$

$coNP \subseteq PSPACE$  (because  $PSPACE = coPSPACE$ )

$P = PSPACE$  ? *Not known.*



# $NP \subseteq PSPACE$

**Theorem:**  $NP \subseteq PSPACE$

Proof:

1.  $SAT \in PSPACE$
2. If  $A \leq_p B$  and  $B \in PSPACE$  then  $A \in PSPACE$

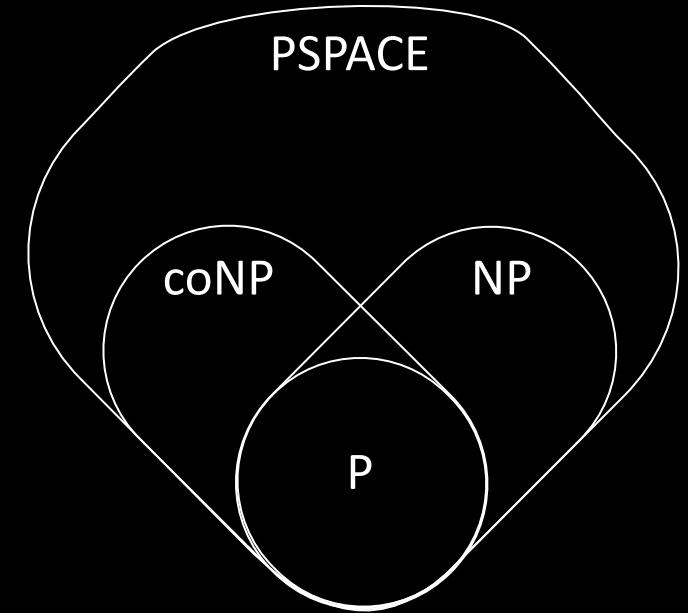
**Defn:**  $coNP = \{\overline{A} \mid A \in NP\}$

$\overline{HAMPATH} \in coNP$

$TAUTOLOGY = \{\langle \phi \rangle \mid \text{all assignments satisfy } \phi\} \in coNP$

$coNP \subseteq PSPACE$  (because  $PSPACE = coPSPACE$ )

$P = PSPACE$  ? *Not known.*



Or possibly:

$P = NP = coNP = PSPACE$

# $NP \subseteq PSPACE$

**Theorem:**  $NP \subseteq PSPACE$

Proof:

1.  $SAT \in PSPACE$
2. If  $A \leq_p B$  and  $B \in PSPACE$  then  $A \in PSPACE$

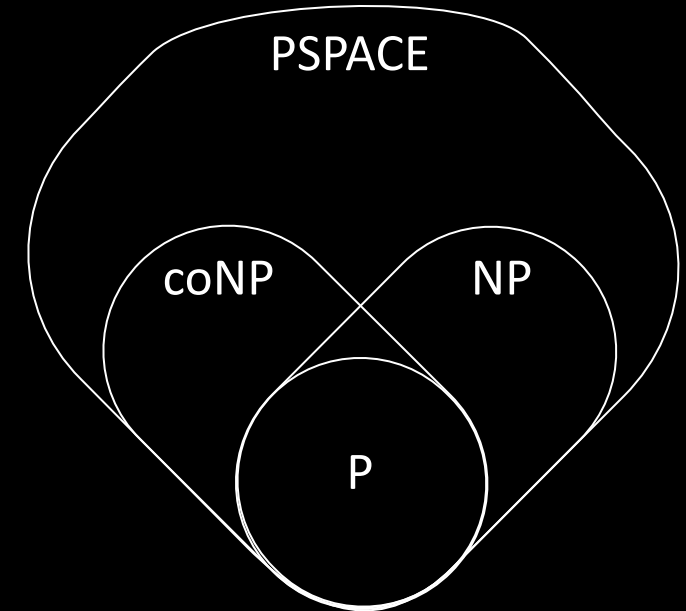
**Defn:**  $coNP = \{\overline{A} \mid A \in NP\}$

$\overline{HAMPATH} \in coNP$

$TAUTOLOGY = \{\langle \phi \rangle \mid \text{all assignments satisfy } \phi\} \in coNP$

$coNP \subseteq PSPACE$  (because  $PSPACE = coPSPACE$ )

$P = PSPACE$  ? *Not known.*



Or possibly:

$P = NP = coNP = PSPACE$

Example:  $TQBF$



# Example: *TQBF*

**Defn:** A quantified Boolean formula (QBF) is a Boolean formula with leading exists ( $\exists x$ ) and for all ( $\forall x$ ) quantifiers. All variables must lie within the scope of a quantifier.

# Example: *TQBF*

**Defn:** A quantified Boolean formula (QBF) is a Boolean formula with leading exists ( $\exists x$ ) and for all ( $\forall x$ ) quantifiers. All variables must lie within the scope of a quantifier.

A QBF is TRUE or FALSE.

# Example: *TQBF*

**Defn:** A quantified Boolean formula (QBF) is a Boolean formula with leading exists ( $\exists x$ ) and for all ( $\forall x$ ) quantifiers. All variables must lie within the scope of a quantifier.

A QBF is TRUE or FALSE.

**Examples:**  $\phi_1 = \forall x \exists y [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$

# Example: *TQBF*

**Defn:** A quantified Boolean formula (QBF) is a Boolean formula with leading exists ( $\exists x$ ) and for all ( $\forall x$ ) quantifiers. All variables must lie within the scope of a quantifier.

A QBF is TRUE or FALSE.

**Examples:**  $\phi_1 = \forall x \exists y [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$   
 $\phi_2 = \exists y \forall x [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$

# Example: *TQBF*

**Defn:** A quantified Boolean formula (QBF) is a Boolean formula with leading exists ( $\exists x$ ) and for all ( $\forall x$ ) quantifiers. All variables must lie within the scope of a quantifier.

A QBF is TRUE or FALSE.

**Examples:**  $\phi_1 = \forall x \exists y [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$  TRUE

$\phi_2 = \exists y \forall x [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$

# Example: *TQBF*

**Defn:** A quantified Boolean formula (QBF) is a Boolean formula with leading exists ( $\exists x$ ) and for all ( $\forall x$ ) quantifiers. All variables must lie within the scope of a quantifier.

A QBF is TRUE or FALSE.

**Examples:**  $\phi_1 = \forall x \exists y [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$  TRUE  
 $\phi_2 = \exists y \forall x [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$  FALSE

# Example: *TQBF*

**Defn:** A quantified Boolean formula (QBF) is a Boolean formula with leading exists ( $\exists x$ ) and for all ( $\forall x$ ) quantifiers. All variables must lie within the scope of a quantifier.

A QBF is TRUE or FALSE.

**Examples:**  $\phi_1 = \forall x \exists y [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$  TRUE

$\phi_2 = \exists y \forall x [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$  FALSE

Defn:  $TQBF = \{ \langle \phi \rangle \mid \phi \text{ is a QBF that is TRUE} \}$

# Example: *TQBF*

**Defn:** A quantified Boolean formula (QBF) is a Boolean formula with leading exists ( $\exists x$ ) and for all ( $\forall x$ ) quantifiers. All variables must lie within the scope of a quantifier.

A QBF is TRUE or FALSE.

**Examples:**  $\phi_1 = \forall x \exists y [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$  TRUE

$\phi_2 = \exists y \forall x [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$  FALSE

Defn:  $TQBF = \{ \langle \phi \rangle \mid \phi \text{ is a QBF that is TRUE} \}$

Thus  $\phi_1 \in TQBF$  and  $\phi_2 \notin TQBF$ .



# Example: $TQBF$

**Defn:** A quantified Boolean formula (QBF) is a Boolean formula with leading exists ( $\exists x$ ) and for all ( $\forall x$ ) quantifiers. All variables must lie within the scope of a quantifier.

A QBF is TRUE or FALSE.

**Examples:**  $\phi_1 = \forall x \exists y [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$  TRUE  
 $\phi_2 = \exists y \forall x [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$  FALSE

Defn:  $TQBF = \{ \langle \phi \rangle \mid \phi \text{ is a QBF that is TRUE} \}$

Thus  $\phi_1 \in TQBF$  and  $\phi_2 \notin TQBF$ .

**Theorem:**  $TQBF \in PSPACE$

# Example: $TQBF$

**Defn:** A quantified Boolean formula (QBF) is a Boolean formula with leading exists ( $\exists x$ ) and for all ( $\forall x$ ) quantifiers. All variables must lie within the scope of a quantifier.

A QBF is TRUE or FALSE.

**Examples:**  $\phi_1 = \forall x \exists y [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$  TRUE  
 $\phi_2 = \exists y \forall x [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$  FALSE

Defn:  $TQBF = \{ \langle \phi \rangle \mid \phi \text{ is a QBF that is TRUE} \}$

Thus  $\phi_1 \in TQBF$  and  $\phi_2 \notin TQBF$ .

**Theorem:**  $TQBF \in \text{PSPACE}$

# Example: *TQBF*

**Defn:** A quantified Boolean formula (QBF) is a Boolean formula with leading exists ( $\exists x$ ) and for all ( $\forall x$ ) quantifiers. All variables must lie within the scope of a quantifier.

A QBF is TRUE or FALSE.

**Examples:**  $\phi_1 = \forall x \exists y [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$  TRUE  
 $\phi_2 = \exists y \forall x [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$  FALSE

Defn:  $TQBF = \{ \langle \phi \rangle \mid \phi \text{ is a QBF that is TRUE} \}$

Thus  $\phi_1 \in TQBF$  and  $\phi_2 \notin TQBF$ .

**Theorem:**  $TQBF \in PSPACE$

## Check-in 17.2

How is *SAT* a special case of *TQBF*?

- (a) Remove all quantifiers.
- (b) Add  $\exists$  and  $\forall$  quantifiers.
- (c) Add only  $\exists$  quantifiers.
- (d) Add only  $\forall$  quantifiers.

$$TQBF \in \text{PSPACE}$$

**Theorem:**  $TQBF \in \text{PSPACE}$

$$TQBF \in \text{PSPACE}$$

**Theorem:**  $TQBF \in \text{PSPACE}$

Proof: “On input  $\langle \phi \rangle$

$$TQBF \in \text{PSPACE}$$

**Theorem:**  $TQBF \in \text{PSPACE}$

Proof: “On input  $\langle \phi \rangle$

1. If  $\phi$  has no quantifiers, then  $\phi$  has no variables  
so either  $\phi = \text{True}$  or  $\phi = \text{False}$ . Output accordingly.

# $TQBF \in \text{PSPACE}$

**Theorem:**  $TQBF \in \text{PSPACE}$

Proof: “On input  $\langle \phi \rangle$

1. If  $\phi$  has no quantifiers, then  $\phi$  has no variables  
so either  $\phi = \text{True}$  or  $\phi = \text{False}$ . Output accordingly.
2. If  $\phi = \exists x \psi$  then evaluate  $\psi$  with  $x = \text{TRUE}$  and  $x = \text{FALSE}$  recursively.  
*Accept* if either accepts. *Reject* if not.

# $TQBF \in PSPACE$

**Theorem:**  $TQBF \in PSPACE$

Proof: “On input  $\langle \phi \rangle$

1. If  $\phi$  has no quantifiers, then  $\phi$  has no variables  
so either  $\phi = \text{True}$  or  $\phi = \text{False}$ . Output accordingly.
2. If  $\phi = \exists x \psi$  then evaluate  $\psi$  with  $x = \text{TRUE}$  and  $x = \text{FALSE}$  recursively.  
*Accept* if either accepts. *Reject* if not.
3. If  $\phi = \forall x \psi$  then evaluate  $\psi$  with  $x = \text{TRUE}$  and  $x = \text{FALSE}$  recursively.  
*Accept* if both accept. *Reject* if not.”



# $TQBF \in PSPACE$

**Theorem:**  $TQBF \in PSPACE$

Proof: “On input  $\langle \phi \rangle$

1. If  $\phi$  has no quantifiers, then  $\phi$  has no variables  
so either  $\phi = \text{True}$  or  $\phi = \text{False}$ . Output accordingly.
2. If  $\phi = \exists x \psi$  then evaluate  $\psi$  with  $x = \text{TRUE}$  and  $x = \text{FALSE}$  recursively.  
*Accept* if either accepts. *Reject* if not.
3. If  $\phi = \forall x \psi$  then evaluate  $\psi$  with  $x = \text{TRUE}$  and  $x = \text{FALSE}$  recursively.  
*Accept* if both accept. *Reject* if not.”

Space analysis:

# $TQBF \in PSPACE$

**Theorem:**  $TQBF \in PSPACE$

Proof: “On input  $\langle \phi \rangle$

1. If  $\phi$  has no quantifiers, then  $\phi$  has no variables  
so either  $\phi = \text{True}$  or  $\phi = \text{False}$ . Output accordingly.
2. If  $\phi = \exists x \psi$  then evaluate  $\psi$  with  $x = \text{TRUE}$  and  $x = \text{FALSE}$  recursively.  
*Accept* if either accepts. *Reject* if not.
3. If  $\phi = \forall x \psi$  then evaluate  $\psi$  with  $x = \text{TRUE}$  and  $x = \text{FALSE}$  recursively.  
*Accept* if both accept. *Reject* if not.”

Space analysis:

Each recursive level uses constant space (to record the  $x$  value).

# $TQBF \in PSPACE$

**Theorem:**  $TQBF \in PSPACE$

Proof: “On input  $\langle \phi \rangle$

1. If  $\phi$  has no quantifiers, then  $\phi$  has no variables  
so either  $\phi = \text{True}$  or  $\phi = \text{False}$ . Output accordingly.
2. If  $\phi = \exists x \psi$  then evaluate  $\psi$  with  $x = \text{TRUE}$  and  $x = \text{FALSE}$  recursively.  
*Accept* if either accepts. *Reject* if not.
3. If  $\phi = \forall x \psi$  then evaluate  $\psi$  with  $x = \text{TRUE}$  and  $x = \text{FALSE}$  recursively.  
*Accept* if both accept. *Reject* if not.”

Space analysis:

Each recursive level uses constant space (to record the  $x$  value).

The recursion depth is the number of quantifiers, at most  $n = |\langle \phi \rangle|$ .

# $TQBF \in PSPACE$

**Theorem:**  $TQBF \in PSPACE$

Proof: “On input  $\langle \phi \rangle$

1. If  $\phi$  has no quantifiers, then  $\phi$  has no variables  
so either  $\phi = \text{True}$  or  $\phi = \text{False}$ . Output accordingly.
2. If  $\phi = \exists x \psi$  then evaluate  $\psi$  with  $x = \text{TRUE}$  and  $x = \text{FALSE}$  recursively.  
*Accept* if either accepts. *Reject* if not.
3. If  $\phi = \forall x \psi$  then evaluate  $\psi$  with  $x = \text{TRUE}$  and  $x = \text{FALSE}$  recursively.  
*Accept* if both accept. *Reject* if not.”

Space analysis:

Each recursive level uses constant space (to record the  $x$  value).

The recursion depth is the number of quantifiers, at most  $n = |\langle \phi \rangle|$ .

So  $TQBF \in SPACE(n)$

# $TQBF \in PSPACE$

**Theorem:**  $TQBF \in PSPACE$

Proof: “On input  $\langle \phi \rangle$

1. If  $\phi$  has no quantifiers, then  $\phi$  has no variables  
so either  $\phi = \text{True}$  or  $\phi = \text{False}$ . Output accordingly.
2. If  $\phi = \exists x \psi$  then evaluate  $\psi$  with  $x = \text{TRUE}$  and  $x = \text{FALSE}$  recursively.  
*Accept* if either accepts. *Reject* if not.
3. If  $\phi = \forall x \psi$  then evaluate  $\psi$  with  $x = \text{TRUE}$  and  $x = \text{FALSE}$  recursively.  
*Accept* if both accept. *Reject* if not.”

Space analysis:

Each recursive level uses constant space (to record the  $x$  value).

The recursion depth is the number of quantifiers, at most  $n = |\langle \phi \rangle|$ .

So  $TQBF \in SPACE(n)$

# Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

# Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

A word ladder for English is a ladder of English words.

# Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

A word ladder for English is a ladder of English words.

Let  $A$  be a language. A ladder in  $A$  is a ladder of strings in  $A$ .



# Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

A word ladder for English is a ladder of English words.

Let  $A$  be a language. A ladder in  $A$  is a ladder of strings in  $A$ .

WORK
PORK
PORT

# Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

A word ladder for English is a ladder of English words.

Let  $A$  be a language. A ladder in  $A$  is a ladder of strings in  $A$ .

WORK
PORK
PORT
SORT

# Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

A word ladder for English is a ladder of English words.

Let  $A$  be a language. A ladder in  $A$  is a ladder of strings in  $A$ .

WORK
PORK
PORT
SORT
SOOT

# Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

A word ladder for English is a ladder of English words.

Let  $A$  be a language. A ladder in  $A$  is a ladder of strings in  $A$ .

WORK
PORK
PORT
SORT
SOOT
SLOT

# Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

A word ladder for English is a ladder of English words.

Let  $A$  be a language. A ladder in  $A$  is a ladder of strings in  $A$ .

WORK
PORK
PORT
SORT
SOOT
SLOT
PLOT

# Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

A word ladder for English is a ladder of English words.

Let  $A$  be a language. A ladder in  $A$  is a ladder of strings in  $A$ .

WORK
PORK
PORT
SORT
SOOT
SLOT
PLOT
PLOY

# Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

A word ladder for English is a ladder of English words.

Let  $A$  be a language. A ladder in  $A$  is a ladder of strings in  $A$ .

WORK
PORK
PORT
SORT
SOOT
SLOT
PLOT
PLOY
PLAY

# Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

A word ladder for English is a ladder of English words.

Let  $A$  be a language. A ladder in  $A$  is a ladder of strings in  $A$ .

WORK
PORK
PORT
SORT
SOOT
SLOT
PLOT
PLOY
PLAY



# Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

A word ladder for English is a ladder of English words.

Let  $A$  be a language. A ladder in  $A$  is a ladder of strings in  $A$ .

**Defn:**  $LADDERDFA = \{ \langle B, u, v \rangle \mid B \text{ is a DFA and } L(B) \text{ contains a ladder } y_1, y_2, \dots, y_k \text{ where } y_1 = u \text{ and } y_k = v \}.$

WORK
PORK
PORT
SORT
SOOT
SLOT
PLOT
PLOY
PLAY

# Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

A word ladder for English is a ladder of English words.

Let  $A$  be a language. A ladder in  $A$  is a ladder of strings in  $A$ .

**Defn:**  $LADDERDFA = \{ \langle B, u, v \rangle \mid B \text{ is a DFA and } L(B) \text{ contains a ladder } y_1, y_2, \dots, y_k \text{ where } y_1 = u \text{ and } y_k = v \}.$

**Theorem:**  $LADDERDFA \in \text{NPSPACE}$

WORK
PORK
PORT
SORT
SOOT
SLOT
PLOT
PLOY
PLAY

# Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

A word ladder for English is a ladder of English words.

Let  $A$  be a language. A ladder in  $A$  is a ladder of strings in  $A$ .

**Defn:**  $LADDERDFA = \{ \langle B, u, v \rangle \mid B \text{ is a DFA and } L(B) \text{ contains a ladder } y_1, y_2, \dots, y_k \text{ where } y_1 = u \text{ and } y_k = v \}$ .

**Theorem:**  $LADDERDFA \in \text{NPSPACE}$

WORK
PORK
PORT
SORT
SOOT
SLOT
PLOT
PLOY
PLAY

# *LADDER*DFA $\in$ NPSPACE

Theorem: *LADDER*DFA  $\in$  NPSPACE

# *LADDER*DFA $\in$ NPSPACE

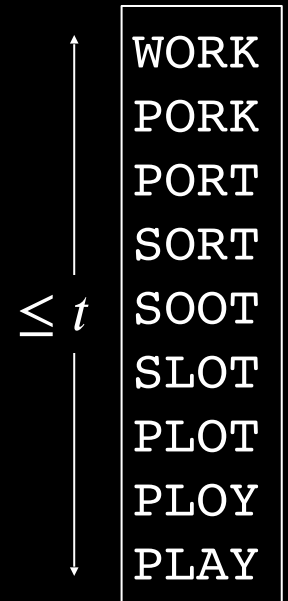
Theorem: *LADDER*DFA  $\in$  NPSPACE

Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

# *LADDER*DFA $\in$ NPSPACE

Theorem: *LADDER*DFA  $\in$  NPSPACE

Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

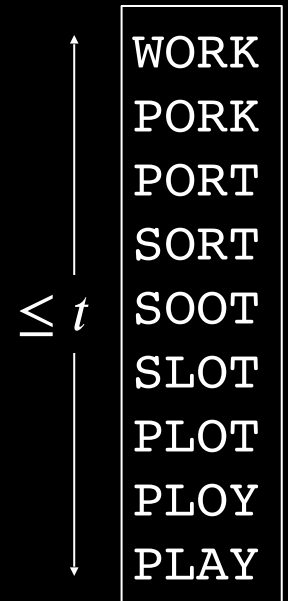


# *LADDER*DFA $\in$ NPSPACE

Theorem: *LADDER*DFA  $\in$  NPSPACE

Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

Careful- (a) cannot store sequence, (b) must terminate.



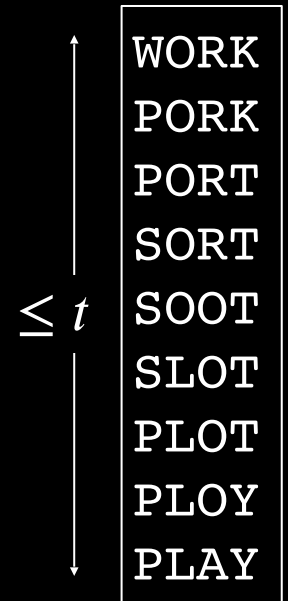
# *LADDER*DFA $\in$ NPSPACE

Theorem: *LADDER*DFA  $\in$  NPSPACE

Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

Careful- (a) cannot store sequence, (b) must terminate.

Proof: “On input  $\langle B, u, v \rangle$





# *LADDER*DFA $\in$ NPSPACE

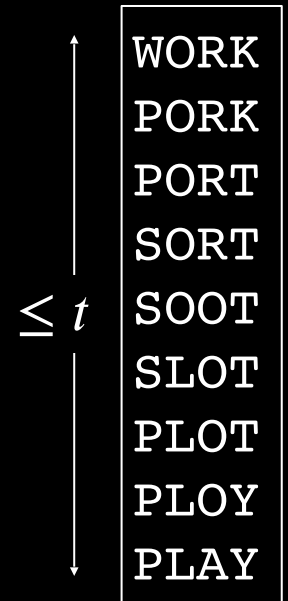
Theorem: *LADDER*DFA  $\in$  NPSPACE

Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

Careful- (a) cannot store sequence, (b) must terminate.

Proof: “On input  $\langle B, u, v \rangle$

1. Let  $y = u$  and let  $m = |u|$ .



# *LADDER*DFA $\in$ NPSPACE

Theorem: *LADDER*DFA  $\in$  NPSPACE

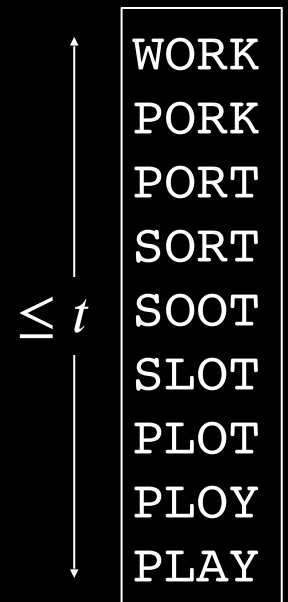
Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

Careful- (a) cannot store sequence, (b) must terminate.

Proof: “On input  $\langle B, u, v \rangle$

1. Let  $y = u$  and let  $m = |u|$ .

2. Repeat at most  $t$  times where  $t = |\Sigma|^m$ .



# *LADDER*DFA $\in$ NPSPACE

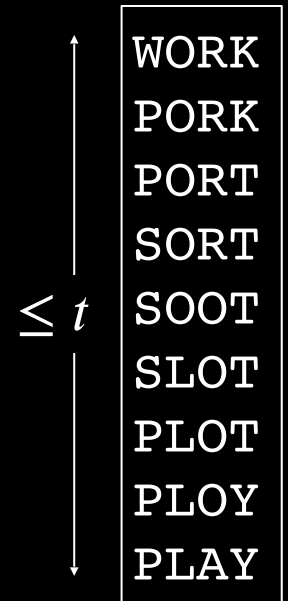
Theorem: *LADDER*DFA  $\in$  NPSPACE

Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

Careful- (a) cannot store sequence, (b) must terminate.

Proof: “On input  $\langle B, u, v \rangle$

1. Let  $y = u$  and let  $m = |u|$ .
2. Repeat at most  $t$  times where  $t = |\Sigma|^m$ .
3. Nondeterministically change one symbol in  $y$ .



# *LADDER*DFA $\in$ NPSPACE

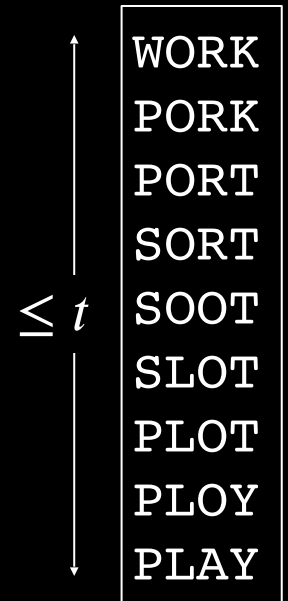
Theorem: *LADDER*DFA  $\in$  NPSPACE

Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

Careful- (a) cannot store sequence, (b) must terminate.

Proof: “On input  $\langle B, u, v \rangle$

1. Let  $y = u$  and let  $m = |u|$ .
2. Repeat at most  $t$  times where  $t = |\Sigma|^m$ .
3. Nondeterministically change one symbol in  $y$ .
4. *Reject* if  $y \notin L(B)$ .



# *LADDERDFA* $\in$ NPSPACE

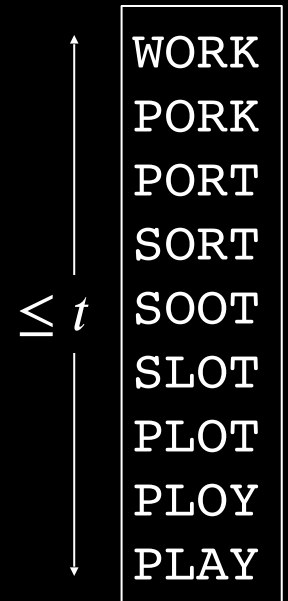
Theorem: *LADDERDFA*  $\in$  NPSPACE

Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

Careful- (a) cannot store sequence, (b) must terminate.

Proof: “On input  $\langle B, u, v \rangle$

1. Let  $y = u$  and let  $m = |u|$ .
2. Repeat at most  $t$  times where  $t = |\Sigma|^m$ .
3. Nondeterministically change one symbol in  $y$ .
4. *Reject* if  $y \notin L(B)$ .
5. *Accept* if  $y = v$ .



# *LADDER*DFA $\in$ NPSPACE

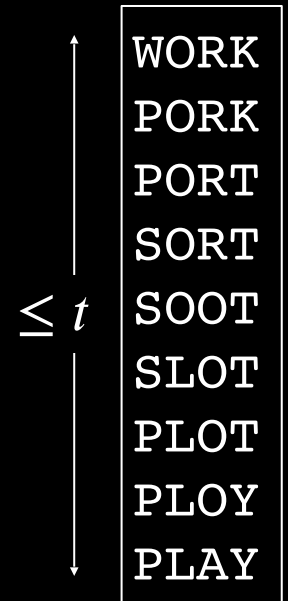
Theorem: *LADDER*DFA  $\in$  NPSPACE

Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

Careful- (a) cannot store sequence, (b) must terminate.

Proof: “On input  $\langle B, u, v \rangle$

1. Let  $y = u$  and let  $m = |u|$ .
2. Repeat at most  $t$  times where  $t = |\Sigma|^m$ .
3. Nondeterministically change one symbol in  $y$ .
4. *Reject* if  $y \notin L(B)$ .
5. *Accept* if  $y = v$ .
6. *Reject* [exceeded  $t$  steps].



# *LADDERDFA* $\in$ NPSPACE

Theorem: *LADDERDFA*  $\in$  NPSPACE

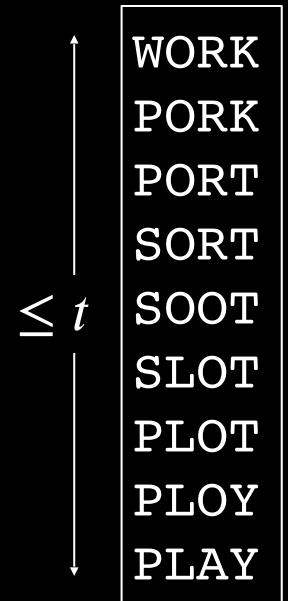
Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

Careful- (a) cannot store sequence, (b) must terminate.

Proof: “On input  $\langle B, u, v \rangle$

1. Let  $y = u$  and let  $m = |u|$ .
2. Repeat at most  $t$  times where  $t = |\Sigma|^m$ .
3. Nondeterministically change one symbol in  $y$ .
4. *Reject* if  $y \notin L(B)$ .
5. *Accept* if  $y = v$ .
6. *Reject* [exceeded  $t$  steps].

Space used is for storing  $y$  and  $t$ .



# *LADDERDFA* $\in$ NPSPACE

Theorem: *LADDERDFA*  $\in$  NPSPACE

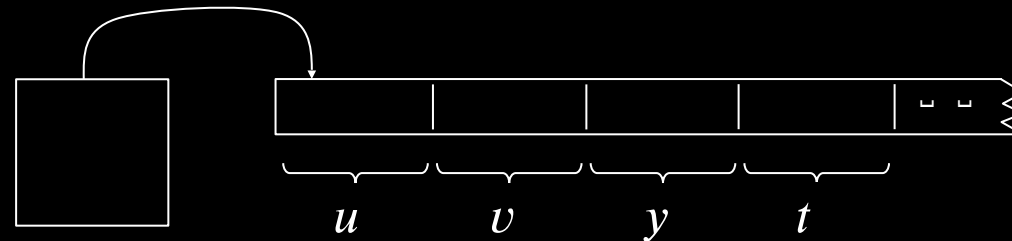
Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

Careful- (a) cannot store sequence, (b) must terminate.

Proof: "On input  $\langle B, u, v \rangle$

1. Let  $y = u$  and let  $m = |u|$ .
2. Repeat at most  $t$  times where  $t = |\Sigma|^m$ .
3. Nondeterministically change one symbol in  $y$ .
4. *Reject* if  $y \notin L(B)$ .
5. *Accept* if  $y = v$ .
6. *Reject* [exceeded  $t$  steps].

Space used is for storing  $y$  and  $t$ .



$\leq t$	WORK
	PORK
	PORT
	SORT
	SOOT
	SLOT
	PLOT
	PLAY



# *LADDERDFA* $\in$ NPSPACE

Theorem: *LADDERDFA*  $\in$  NPSPACE

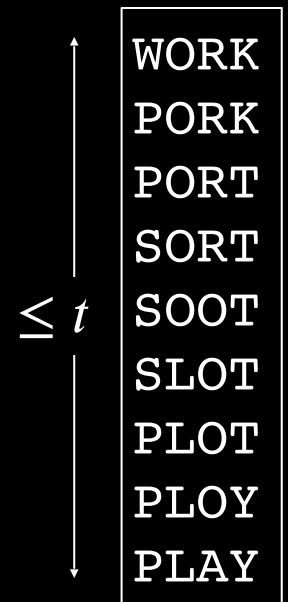
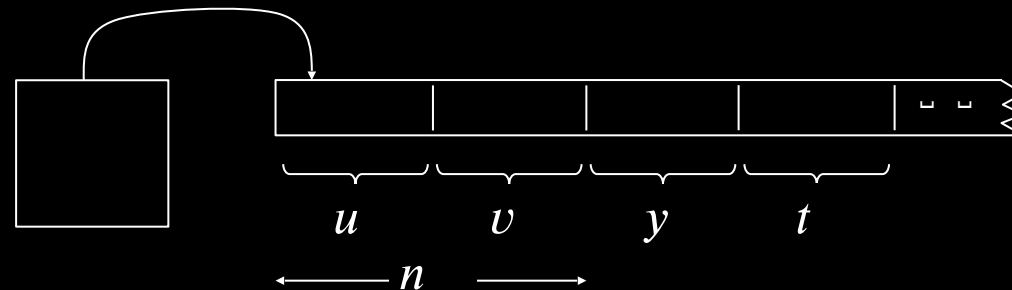
Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

Careful- (a) cannot store sequence, (b) must terminate.

Proof: “On input  $\langle B, u, v \rangle$

1. Let  $y = u$  and let  $m = |u|$ .
2. Repeat at most  $t$  times where  $t = |\Sigma|^m$ .
3. Nondeterministically change one symbol in  $y$ .
4. *Reject* if  $y \notin L(B)$ .
5. *Accept* if  $y = v$ .
6. *Reject* [exceeded  $t$  steps].

Space used is for storing  $y$  and  $t$ .



# *LADDERDFA* $\in$ NPSPACE

Theorem: *LADDERDFA*  $\in$  NPSPACE

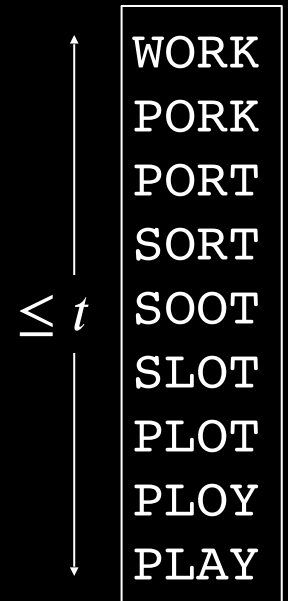
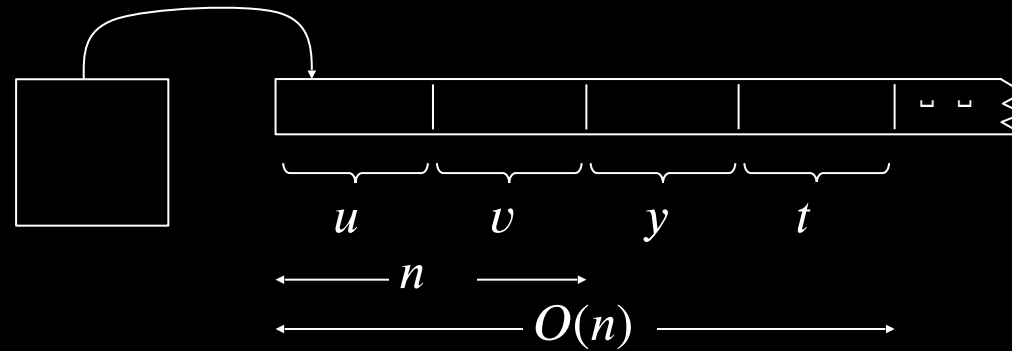
Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

Careful- (a) cannot store sequence, (b) must terminate.

Proof: “On input  $\langle B, u, v \rangle$

1. Let  $y = u$  and let  $m = |u|$ .
2. Repeat at most  $t$  times where  $t = |\Sigma|^m$ .
3. Nondeterministically change one symbol in  $y$ .
4. *Reject* if  $y \notin L(B)$ .
5. *Accept* if  $y = v$ .
6. *Reject* [exceeded  $t$  steps].

Space used is for storing  $y$  and  $t$ .



# *LADDERDFA* $\in$ NPSPACE

Theorem: *LADDERDFA*  $\in$  NPSPACE

Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

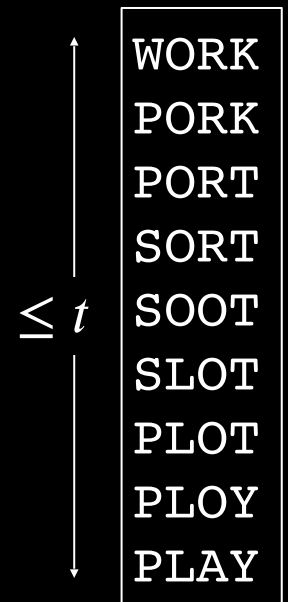
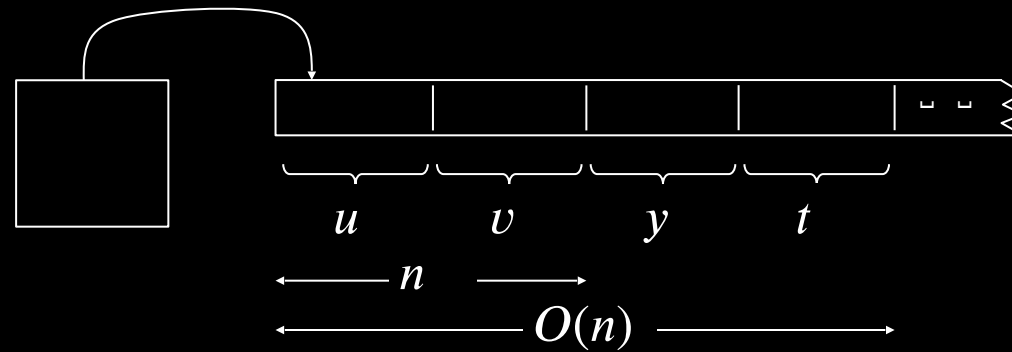
Careful- (a) cannot store sequence, (b) must terminate.

Proof: “On input  $\langle B, u, v \rangle$

1. Let  $y = u$  and let  $m = |u|$ .
2. Repeat at most  $t$  times where  $t = |\Sigma|^m$ .
3. Nondeterministically change one symbol in  $y$ .
4. *Reject* if  $y \notin L(B)$ .
5. *Accept* if  $y = v$ .
6. *Reject* [exceeded  $t$  steps].

Space used is for storing  $y$  and  $t$ .

*LADDERDFA*  $\in$  NPSPACE( $n$ ).



# *LADDERDFA* $\in$ NPSPACE

Theorem: *LADDERDFA*  $\in$  NPSPACE

Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

Careful- (a) cannot store sequence, (b) must terminate.

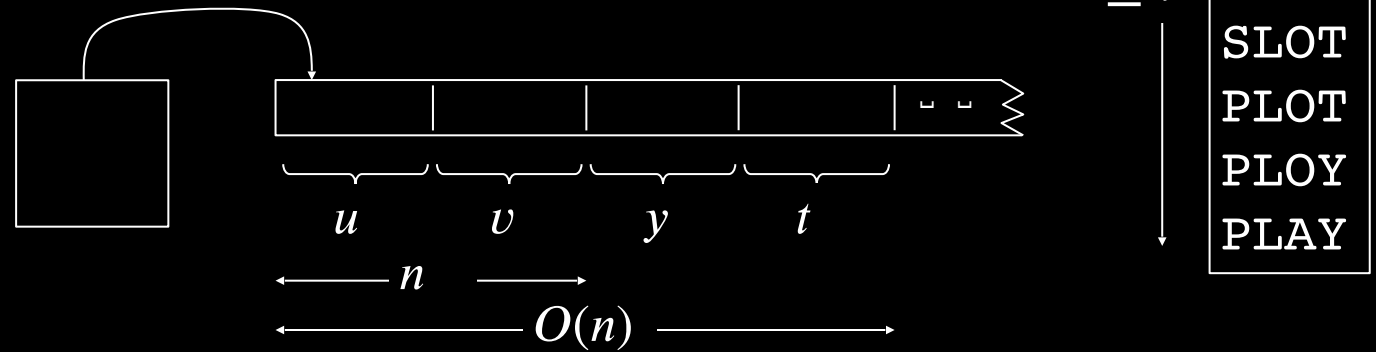
Proof: “On input  $\langle B, u, v \rangle$

1. Let  $y = u$  and let  $m = |u|$ .
2. Repeat at most  $t$  times where  $t = |\Sigma|^m$ .
3. Nondeterministically change one symbol in  $y$ .
4. *Reject* if  $y \notin L(B)$ .
5. *Accept* if  $y = v$ .
6. *Reject* [exceeded  $t$  steps].

Space used is for storing  $y$  and  $t$ .

*LADDERDFA*  $\in$  NSPACE( $n$ ).

Theorem: *LADDERDFA*  $\in$  PSPACE (!)



# *LADDERDFA* $\in$ NPSPACE

Theorem: *LADDERDFA*  $\in$  NPSPACE

Proof idea: Nondeterministically guess the sequence from  $u$  to  $v$ .

Careful- (a) cannot store sequence, (b) must terminate.

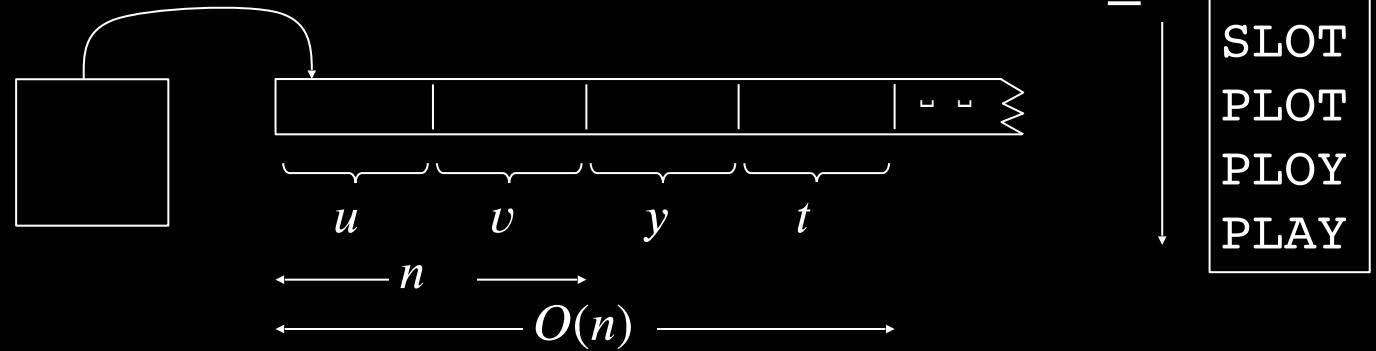
Proof: “On input  $\langle B, u, v \rangle$

1. Let  $y = u$  and let  $m = |u|$ .
2. Repeat at most  $t$  times where  $t = |\Sigma|^m$ .
3. Nondeterministically change one symbol in  $y$ .
4. *Reject* if  $y \notin L(B)$ .
5. *Accept* if  $y = v$ .
6. *Reject* [exceeded  $t$  steps].

Space used is for storing  $y$  and  $t$ .

*LADDERDFA*  $\in$  NSPACE( $n$ ).

Theorem: *LADDERDFA*  $\in$  PSPACE (!)



*LADDERDFA*  $\in$  PSPACE

# $LADDERDFA \in PSPACE$

Theorem:  $LADDERDFA \in SPACE(n^2)$

# $LADDERDFA \in PSPACE$

Theorem:  $LADDERDFA \in SPACE(n^2)$

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .



# *LADDERDFA* $\in$ PSPACE

Theorem: *LADDERDFA*  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

# *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

# *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

WORK

PLAY

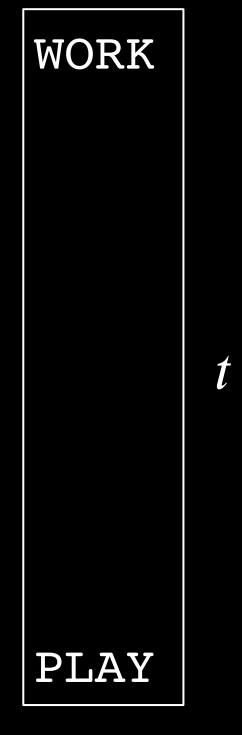
# *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$



# *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$



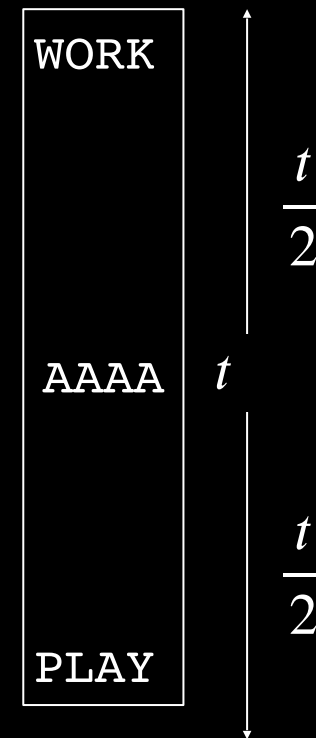
# *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$



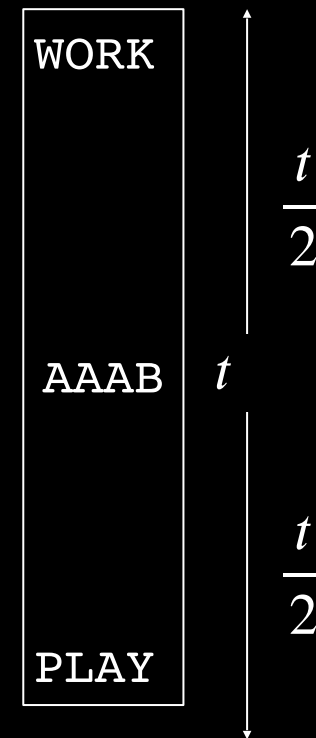
# *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$



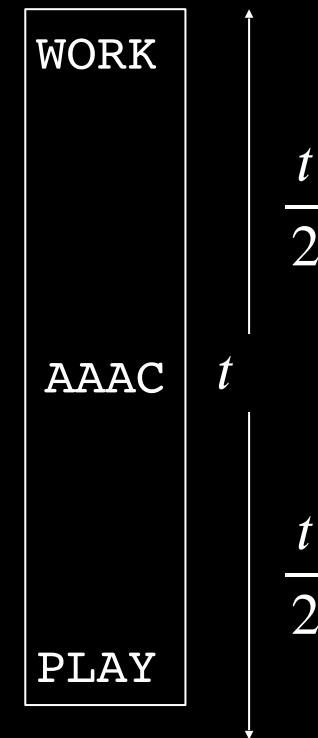
# *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$





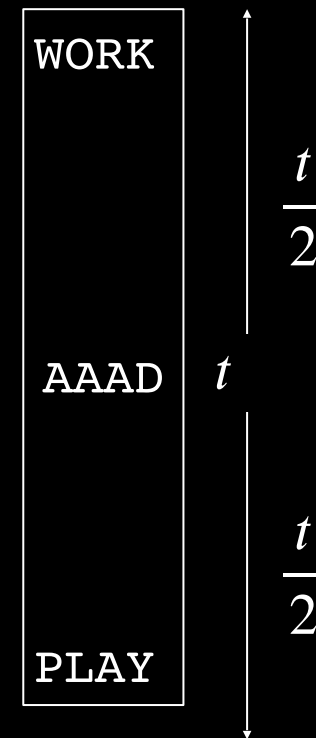
# *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$



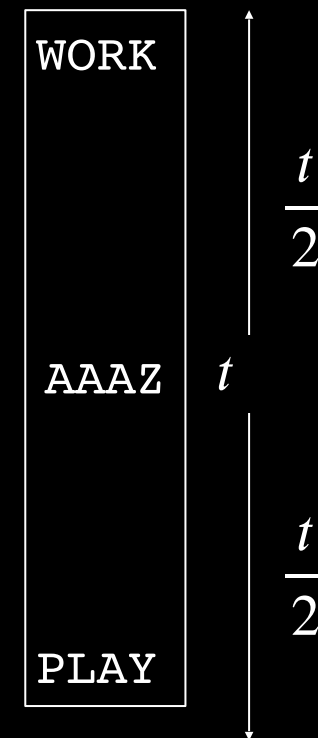
# *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$



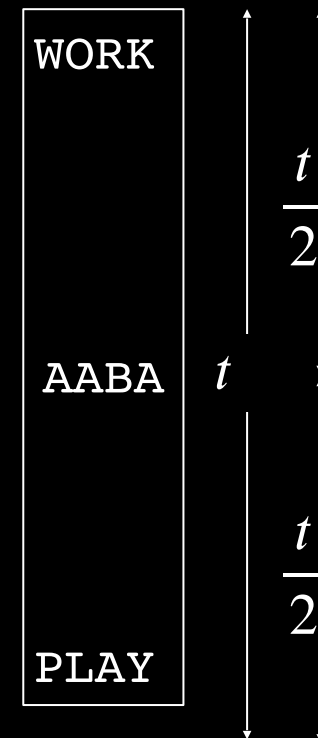
# *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$



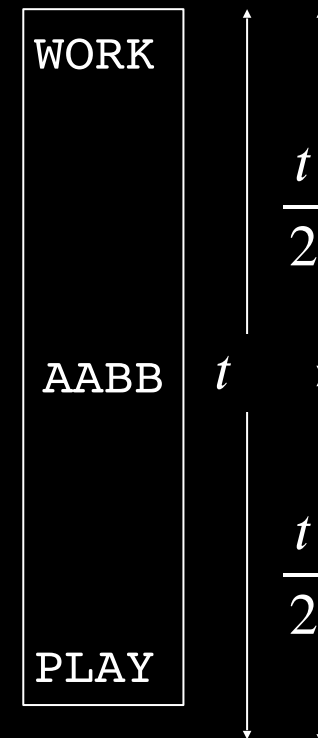
# $LADDERDFA \in PSPACE$

Theorem:  $LADDERDFA \in SPACE(n^2)$

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED-LADDERDFA = \{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$



# *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$



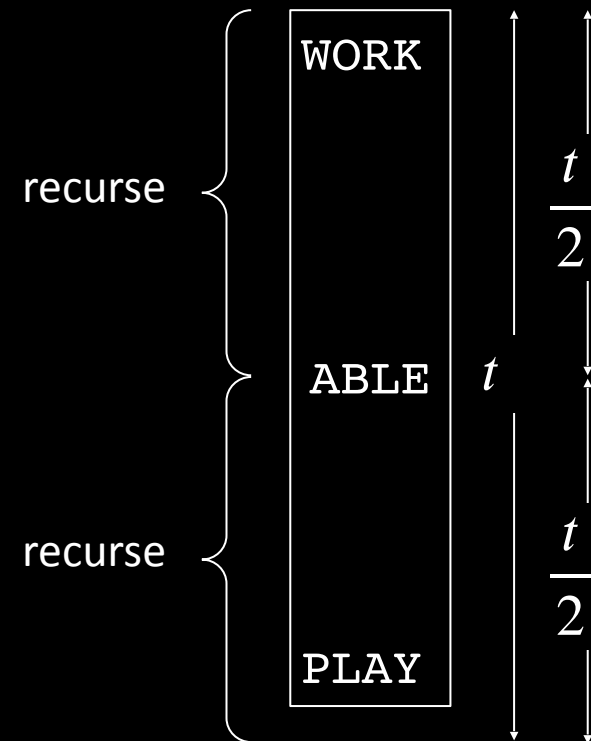
# *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$



# *LADDERDFA* $\in$ PSPACE

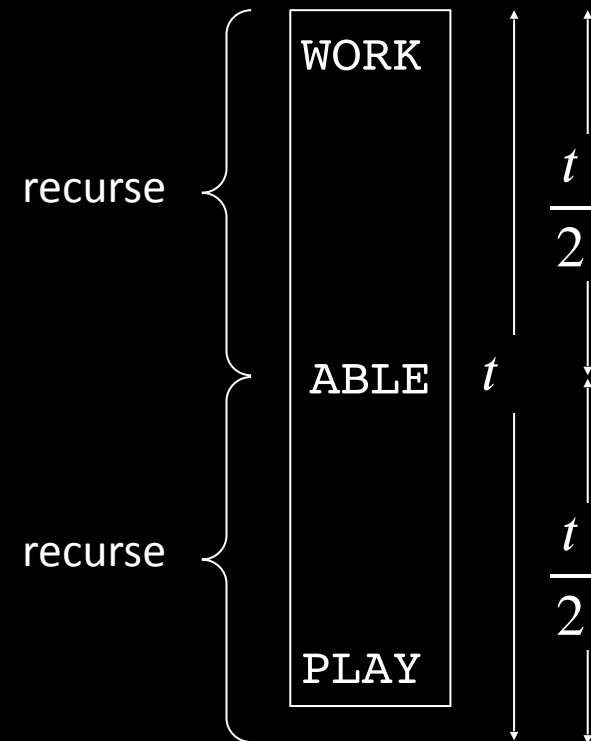
Theorem: *LADDERDFA*  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDERDFA* =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .



# *LADDERDFA* $\in$ PSPACE

Theorem: *LADDERDFA*  $\in$  SPACE( $n^2$ )

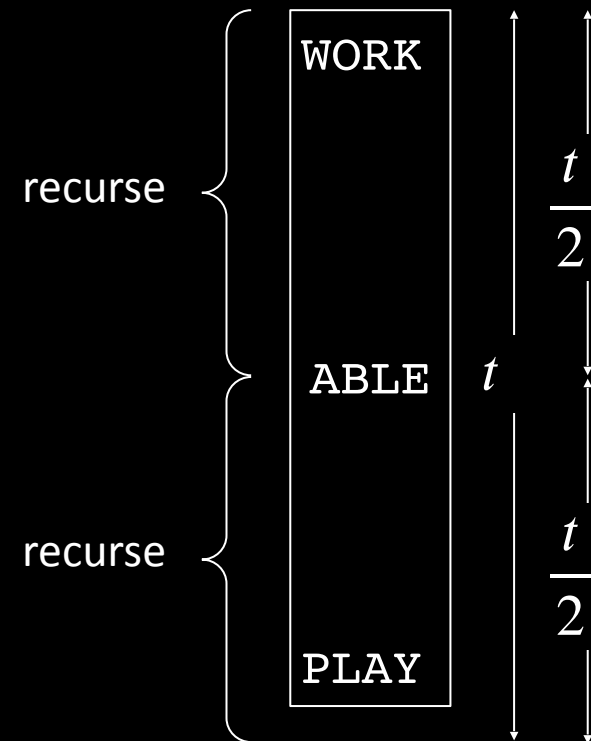
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDERDFA* =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.





# *LADDERDFA* $\in$ PSPACE

Theorem: *LADDERDFA*  $\in$  SPACE( $n^2$ )

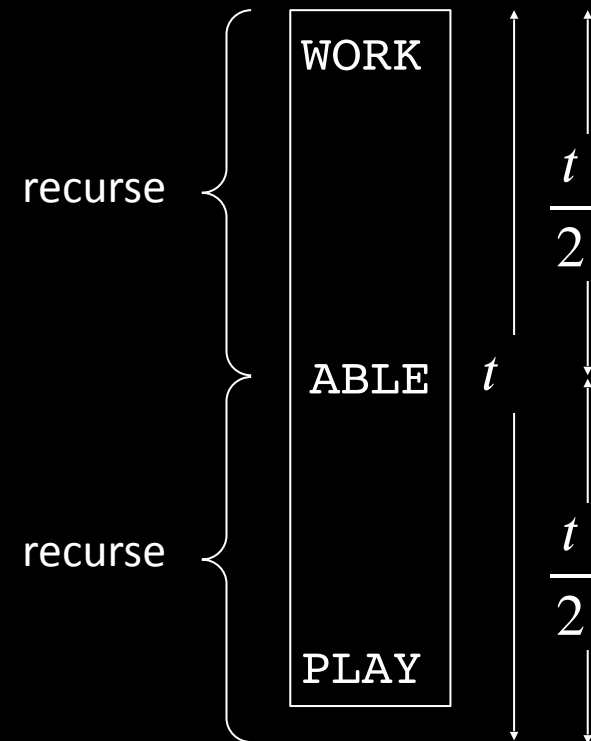
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDERDFA* =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w$  of length  $|u|$



# *LADDERDFA* $\in$ PSPACE

Theorem: *LADDERDFA*  $\in$  SPACE( $n^2$ )

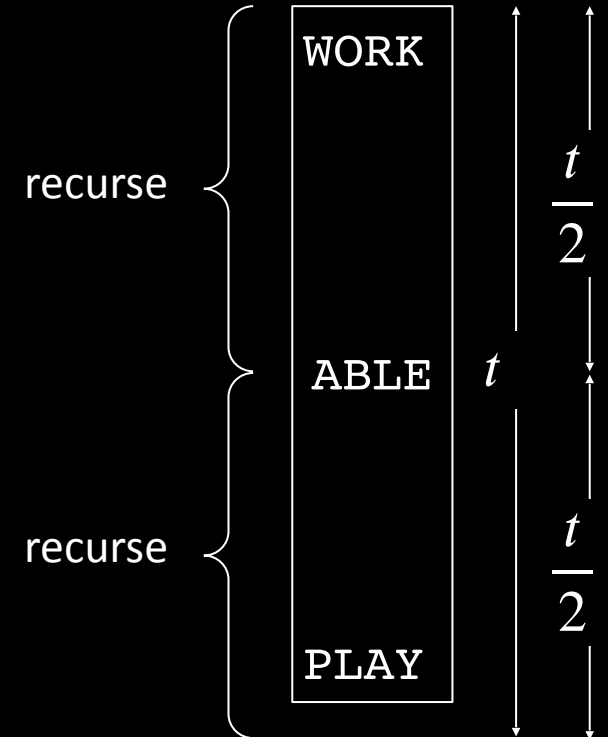
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDERDFA* =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]



# *LADDERDFA* $\in$ PSPACE

Theorem: *LADDERDFA*  $\in$  SPACE( $n^2$ )

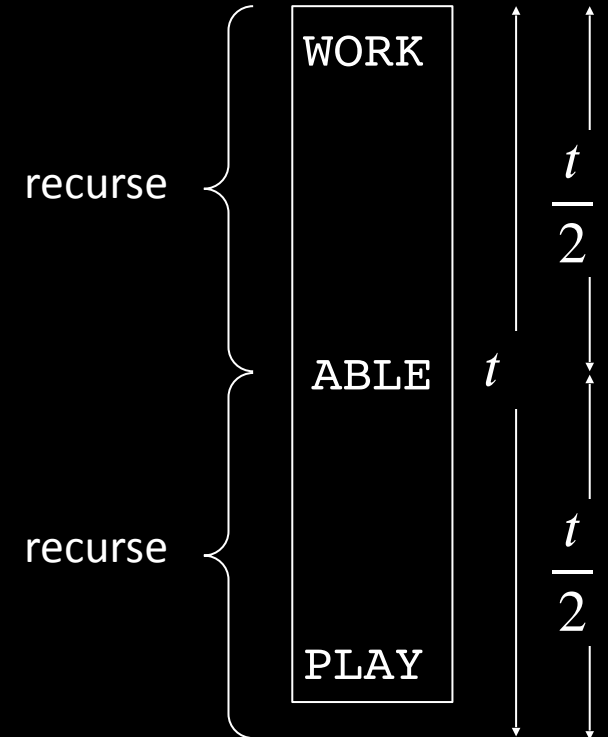
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDERDFA* =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.



# *LADDERDFA* $\in$ PSPACE

Theorem: *LADDERDFA*  $\in$  SPACE( $n^2$ )

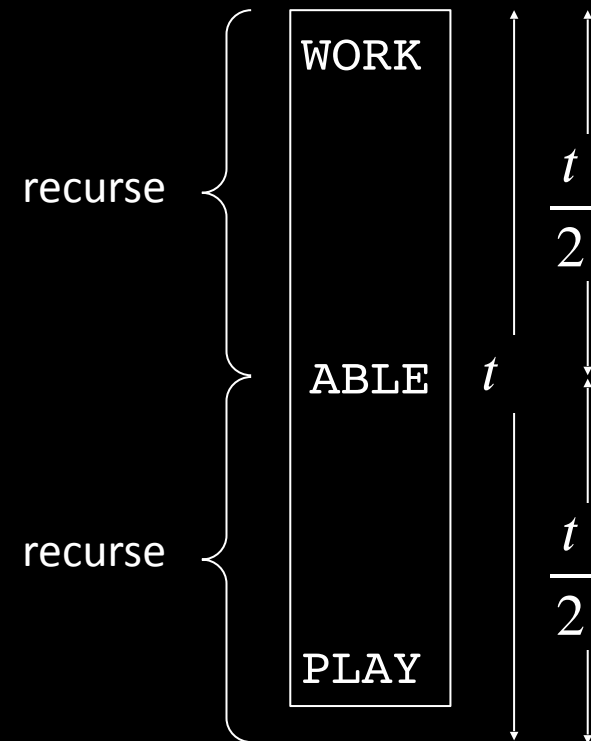
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDERDFA* =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



# *LADDERDFA* $\in$ PSPACE

Theorem: *LADDERDFA*  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

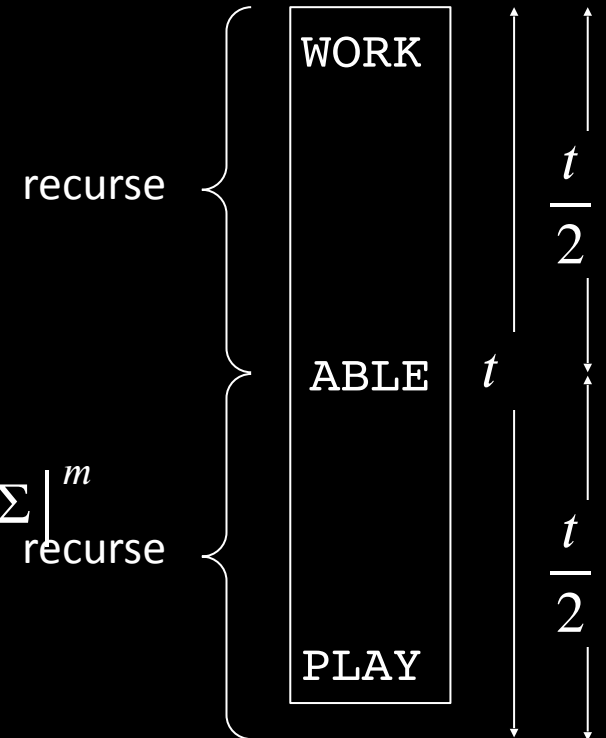
Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDERDFA* =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w$  of length  $|u|$ 
  3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
  4. *Accept* both accept.
  5. *Reject* [if all fail]."

Test  $\langle B, u, v \rangle \in \textit{LADDERDFA}$  with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = \left| \Sigma \right|_{\text{recurse}}^m$



# *LADDERDFA* $\in$ PSPACE

Theorem: *LADDERDFA*  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

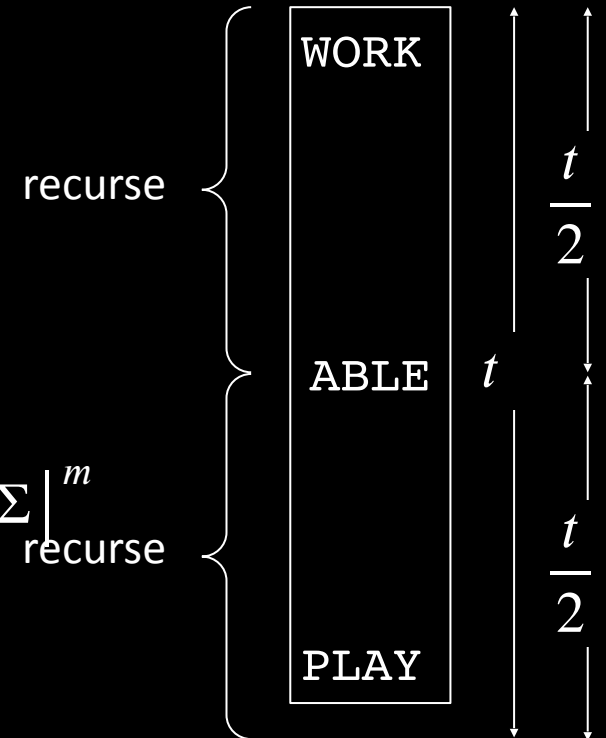
*BOUNDED-LADDERDFA* =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w$  of length  $|u|$ 
  3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
  4. *Accept* both accept.
  5. *Reject* [if all fail]."

Test  $\langle B, u, v \rangle \in \textit{LADDERDFA}$  with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = \left| \Sigma \right|_{\text{recurse}}^m$

Space analysis:



# LADDERDFA $\in$ PSPACE

Theorem:  $LADDERDFA \in \text{SPACE}(n^2)$

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED\text{-}LADDERDFA = \{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

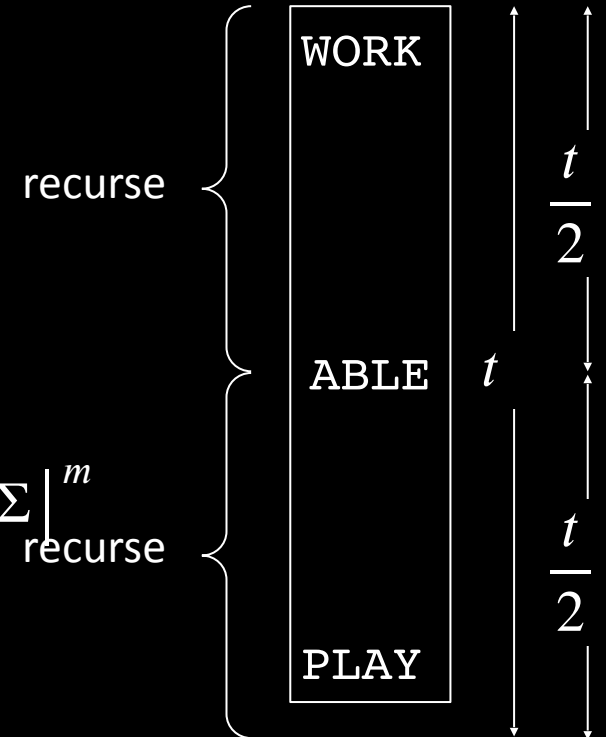
$B\text{-}L =$  "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w$  of length  $|u|$ 
  3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
  4. *Accept* both accept.
  5. *Reject* [if all fail]."

Test  $\langle B, u, v \rangle \in LADDERDFA$  with  $B\text{-}L$  procedure on input  $\langle B, u, v, t \rangle$  for  $t = \left| \Sigma \right|_{\text{recurse}}^m$

Space analysis:

Each recursive level uses space  $O(n)$  (to record  $w$ ).



# *LADDERDFA* $\in$ PSPACE

Theorem: *LADDERDFA*  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDERDFA* =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

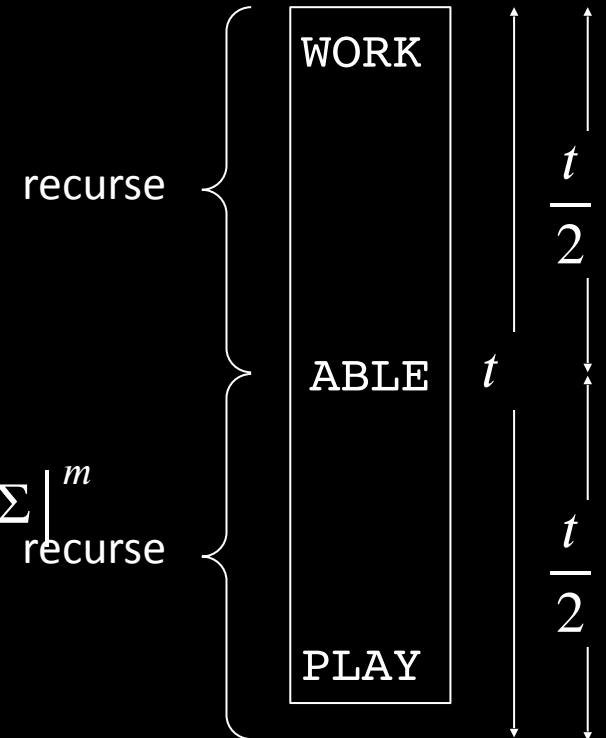
1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w$  of length  $|u|$ 
  3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
  4. *Accept* both accept.
  5. *Reject* [if all fail]."

Test  $\langle B, u, v \rangle \in \textit{LADDERDFA}$  with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = \left| \Sigma \right|_m^{\text{recurse}}$

Space analysis:

Each recursive level uses space  $O(n)$  (to record  $w$ ).

Recursion depth is  $\log t = O(m) = O(n)$ .





# $LADDERDFA \in PSPACE$

Theorem:  $LADDERDFA \in SPACE(n^2)$

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED-LADDERDFA = \{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

$B-L =$  "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w$  of length  $|u|$ 
  3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
  4. *Accept* both accept.
  5. *Reject* [if all fail]."

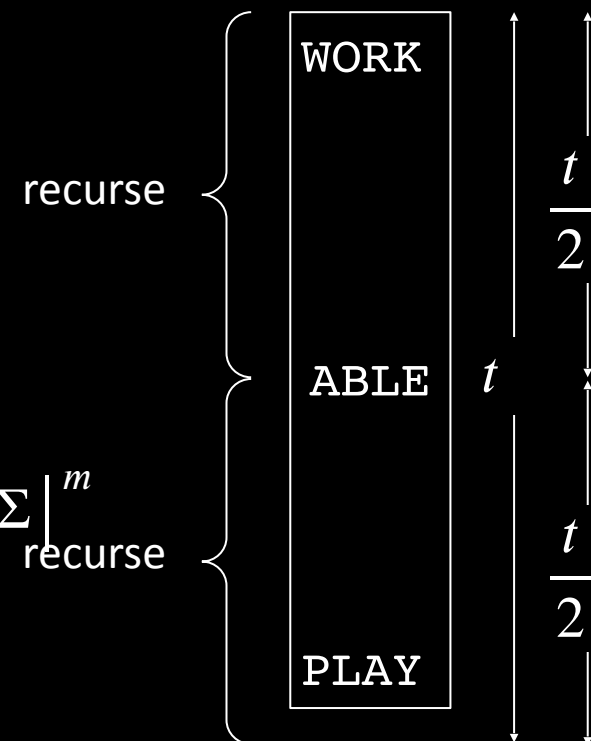
Test  $\langle B, u, v \rangle \in LADDERDFA$  with  $B-L$  procedure on input  $\langle B, u, v, t \rangle$  for  $t = \left| \Sigma \right|_m^{\text{recurse}}$

Space analysis:

Each recursive level uses space  $O(n)$  (to record  $w$ ).

Recursion depth is  $\log t = O(m) = O(n)$ .

Total space used is  $O(n^2)$ .



# LADDERDFA $\in$ PSPACE

Theorem:  $LADDERDFA \in \text{SPACE}(n^2)$

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED\text{-}LADDERDFA = \{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

$B\text{-}L =$  "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w$  of length  $|u|$ 
  3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
  4. *Accept* both accept.
  5. *Reject* [if all fail]."

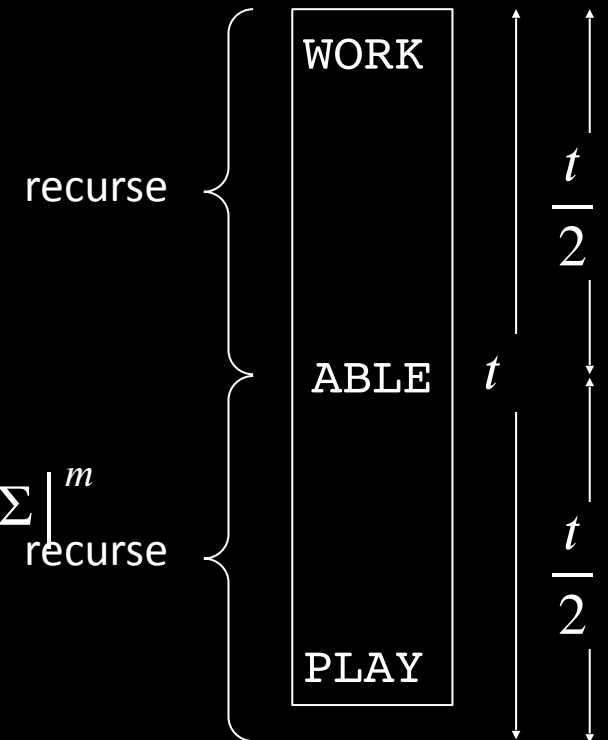
Test  $\langle B, u, v \rangle \in LADDERDFA$  with  $B\text{-}L$  procedure on input  $\langle B, u, v, t \rangle$  for  $t = \left| \sum_{\text{recurse}}^m \right|$

Space analysis:

Each recursive level uses space  $O(n)$  (to record  $w$ ).

Recursion depth is  $\log t = O(m) = O(n)$ .

Total space used is  $O(n^2)$ .



# *LADDERDFA* $\in$ PSPACE

Theorem: *LADDERDFA*  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDERDFA* =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5.     *Reject* [if all fail]."

Test  $\langle B, u, v \rangle \in \textit{LADDERDFA}$  with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t$

Space analysis:

Each recursive level uses space  $O(n)$  (to record  $w$ ).

Recursion depth is  $\log t = O(m) = O(n)$ .

Total space used is  $O(n^2)$ .

## Check-in 17.3

Find an English word ladder connecting MUST and VOTE.

(a) Already did it.

(b) I will.

# Review: SPACE Complexity

**Defn:** Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  where  $f(n) \geq n$ . Say TM  $M$  runs in space  $f(n)$  if  $M$  always halts and uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

# Review: SPACE Complexity

**Defn:** Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  where  $f(n) \geq n$ . Say TM  $M$  runs in space  $f(n)$  if  $M$  always halts and uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

An NTM  $M$  runs in space  $f(n)$  if all branches halt and each branch uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

# Review: SPACE Complexity

**Defn:** Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  where  $f(n) \geq n$ . Say TM  $M$  runs in space  $f(n)$  if  $M$  always halts and uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

An NTM  $M$  runs in space  $f(n)$  if all branches halt and each branch uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

$\text{SPACE}(f(n)) = \{ B \mid \text{some 1-tape TM decides } B \text{ in space } O(f(n)) \}$

$\text{NSPACE}(f(n)) = \{ B \mid \text{some 1-tape NTM decides } B \text{ in space } O(f(n)) \}$

# Review: SPACE Complexity

**Defn:** Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  where  $f(n) \geq n$ . Say TM  $M$  runs in space  $f(n)$  if  $M$  always halts and uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

An NTM  $M$  runs in space  $f(n)$  if all branches halt and each branch uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

$\text{SPACE}(f(n)) = \{B \mid \text{some 1-tape TM decides } B \text{ in space } O(f(n))\}$

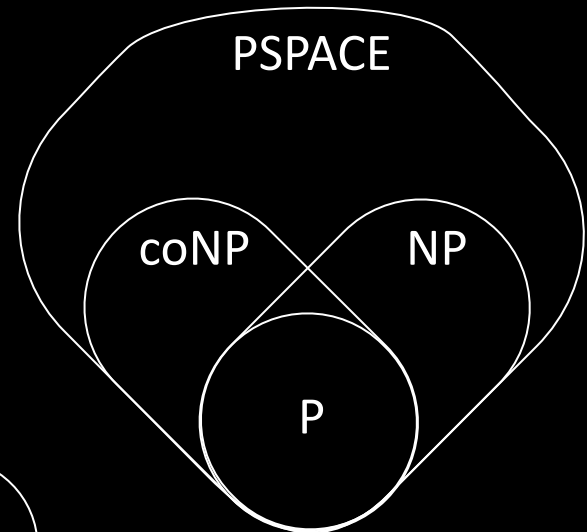
$\text{NSPACE}(f(n)) = \{B \mid \text{some 1-tape NTM decides } B \text{ in space } O(f(n))\}$

$\text{PSPACE} = \bigcup_k \text{SPACE}(n^k)$  “polynomial space”

$\text{NPSPACE} = \bigcup_k \text{NSPACE}(n^k)$  “nondeterministic polynomial space”

Or possibly:

$P = NP = \text{coNP} = \text{PSPACE}$



# Review: SPACE Complexity

**Defn:** Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  where  $f(n) \geq n$ . Say TM  $M$  runs in space  $f(n)$  if  $M$  always halts and uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

An NTM  $M$  runs in space  $f(n)$  if all branches halt and each branch uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

$\text{SPACE}(f(n)) = \{B \mid \text{some 1-tape TM decides } B \text{ in space } O(f(n))\}$

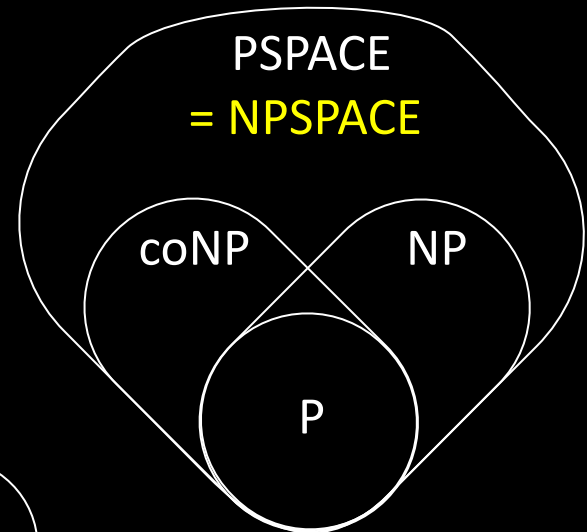
$\text{NSPACE}(f(n)) = \{B \mid \text{some 1-tape NTM decides } B \text{ in space } O(f(n))\}$

$\text{PSPACE} = \bigcup_k \text{SPACE}(n^k)$  “polynomial space”

$\text{NPSPACE} = \bigcup_k \text{NSPACE}(n^k)$  “nondeterministic polynomial space”

Or possibly:

$P = NP = \text{coNP} = \text{PSPACE}$



Today:  $\text{PSPACE} = \text{NPSPACE}$



# Review: SPACE Complexity

**Defn:** Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  where  $f(n) \geq n$ . Say TM  $M$  runs in space  $f(n)$  if  $M$  always halts and uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

An NTM  $M$  runs in space  $f(n)$  if all branches halt and each branch uses at most  $f(n)$  tape cells on all inputs of length  $n$ .

$\text{SPACE}(f(n)) = \{B \mid \text{some 1-tape TM decides } B \text{ in space } O(f(n))\}$

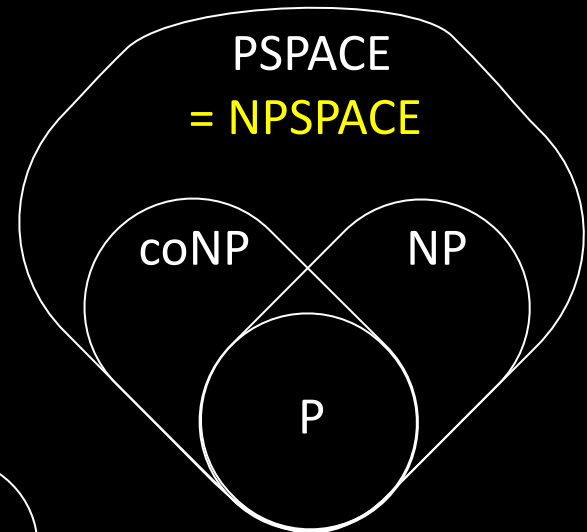
$\text{NSPACE}(f(n)) = \{B \mid \text{some 1-tape NTM decides } B \text{ in space } O(f(n))\}$

$\text{PSPACE} = \bigcup_k \text{SPACE}(n^k)$  “polynomial space”

$\text{NPSPACE} = \bigcup_k \text{NSPACE}(n^k)$  “nondeterministic polynomial space”

Or possibly:

$P = NP = \text{coNP} = \text{PSPACE}$



Today:  $\text{PSPACE} = \text{NPSPACE}$

Review: *LADDER*DFA  $\in$  PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

# Review: $LADDERDFA \in PSPACE$

Theorem:  $LADDERDFA \in SPACE(n^2)$

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED-LADDERDFA = \{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

# Review: *LADDER*DFA $\in$ PSPACE

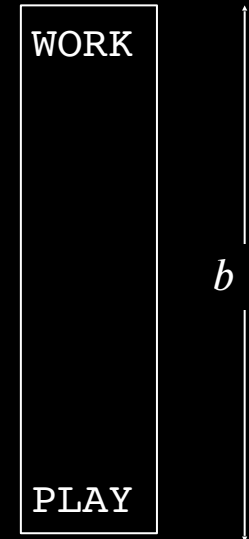
Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

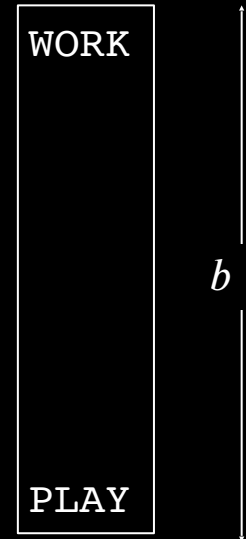
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.





# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

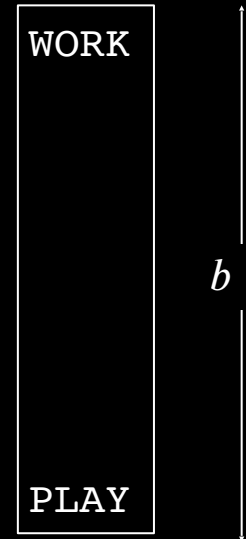
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

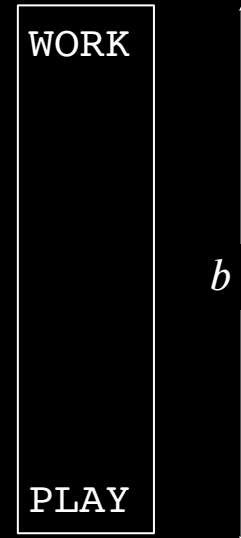
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

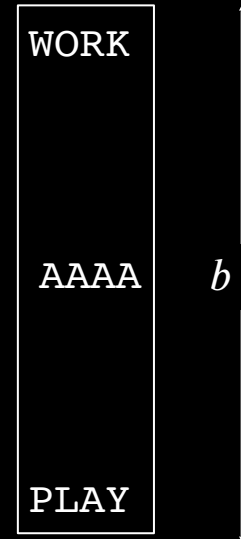
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

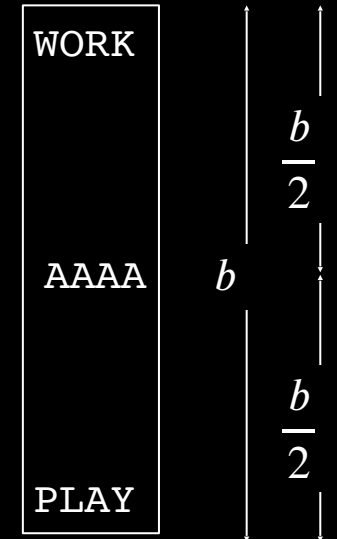
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

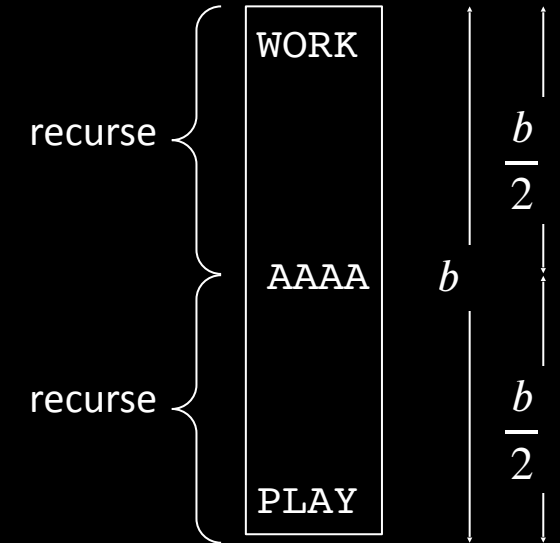
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

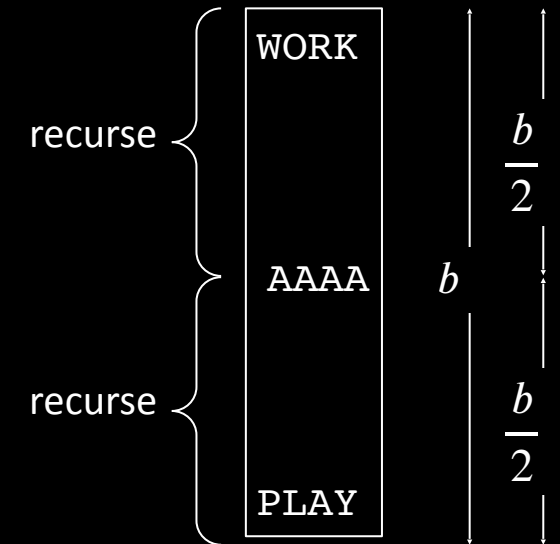
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

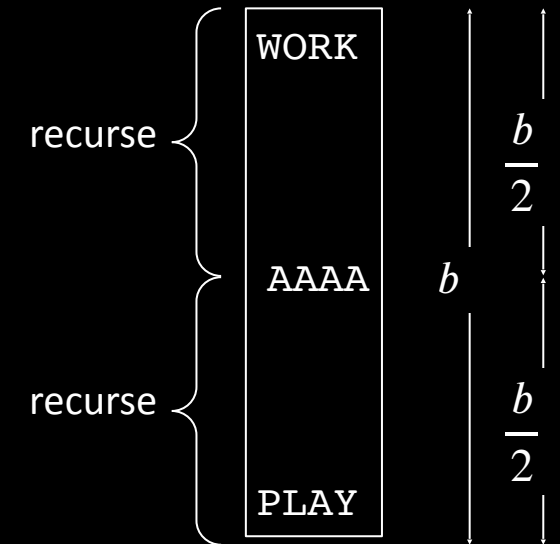
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

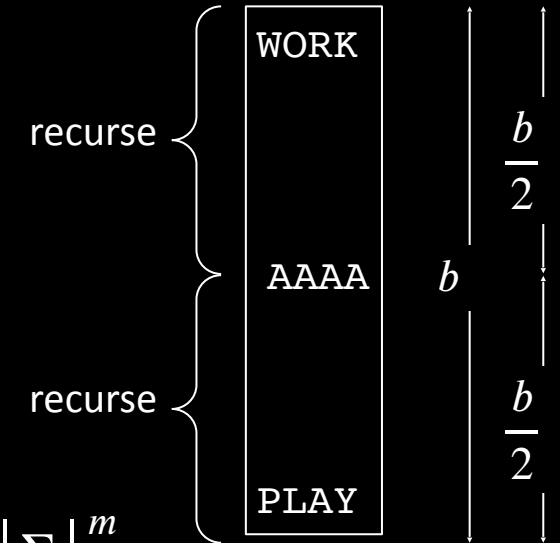
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

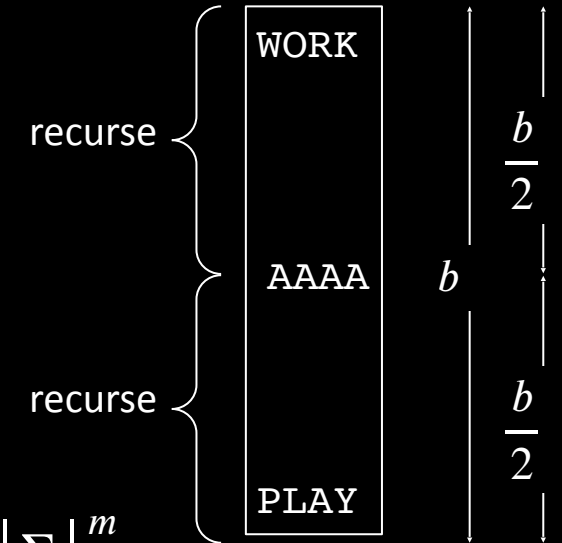
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:

# Review: $LADDERDFA \in PSPACE$

Theorem:  $LADDERDFA \in SPACE(n^2)$

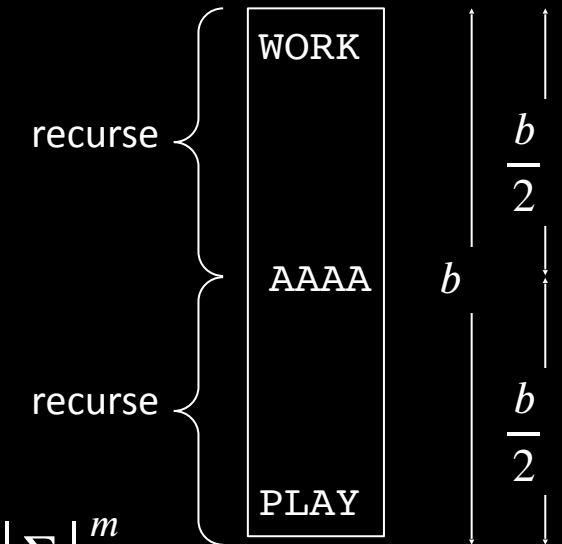
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED-LADDERDFA = \{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

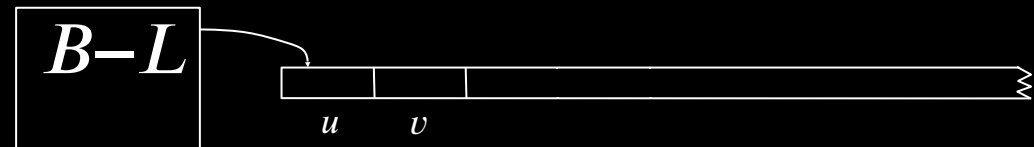
$B-L$  = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in LADDERDFA$  with  $B-L$  procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

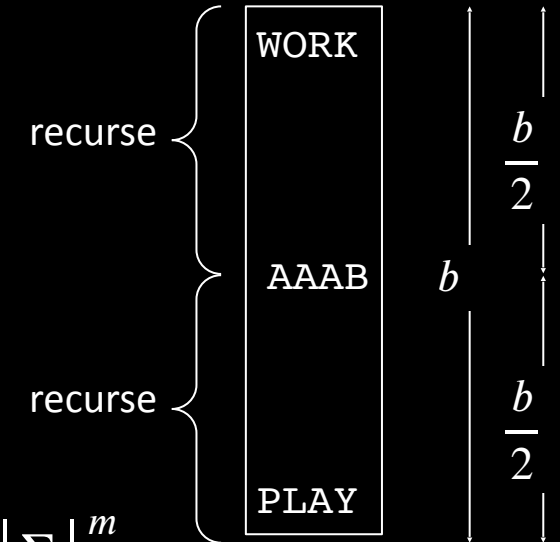
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

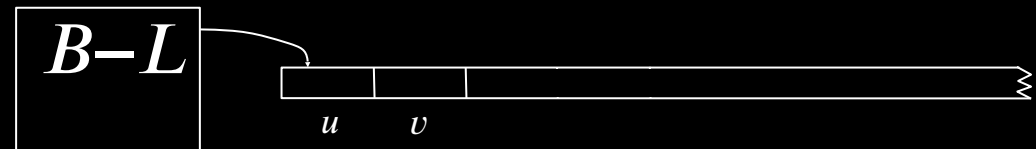
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

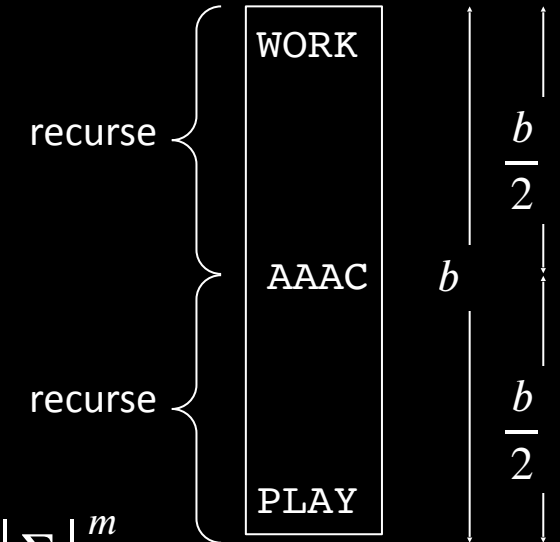
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

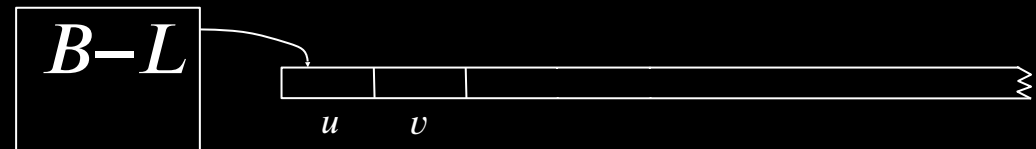
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

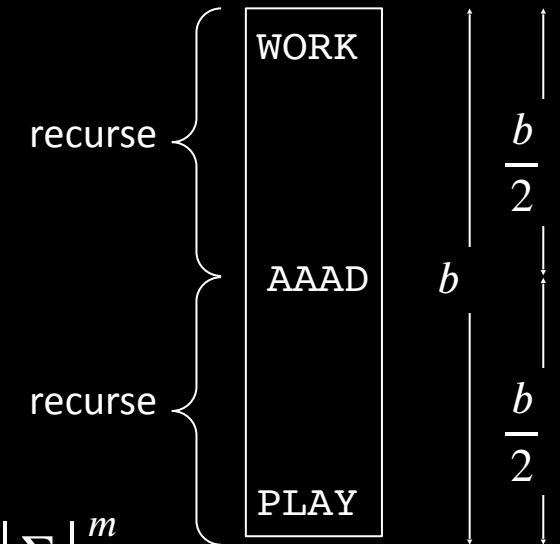
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

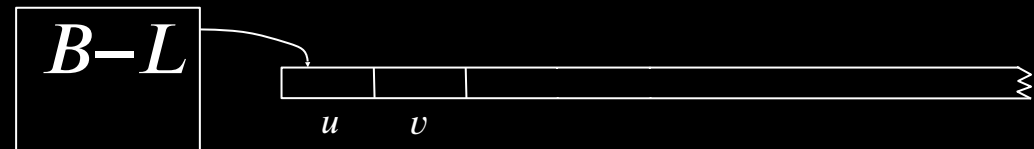
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: $LADDERDFA \in PSPACE$

Theorem:  $LADDERDFA \in SPACE(n^2)$

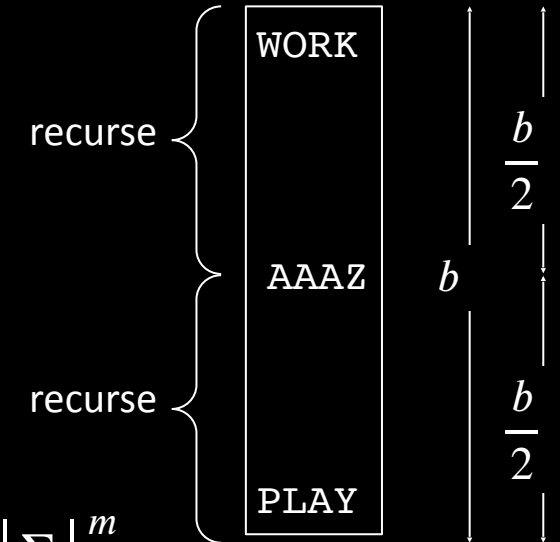
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED-LADDERDFA = \{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

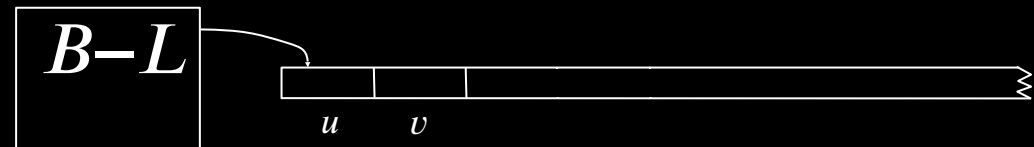
$B-L$  = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in LADDERDFA$  with  $B-L$  procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

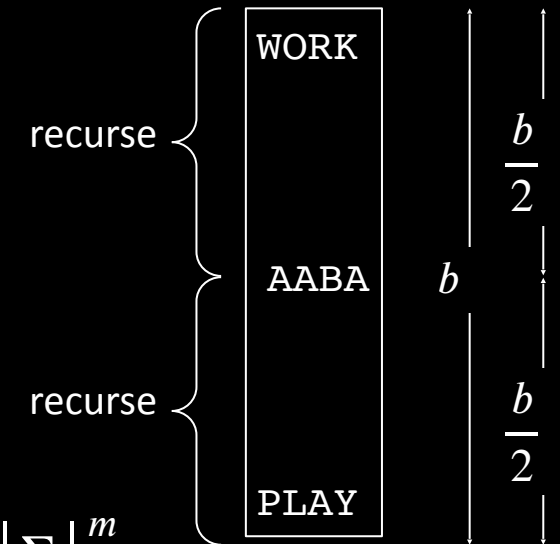
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

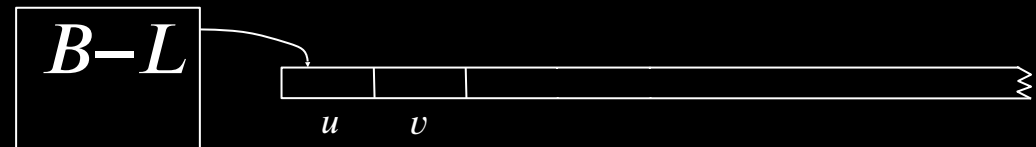
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

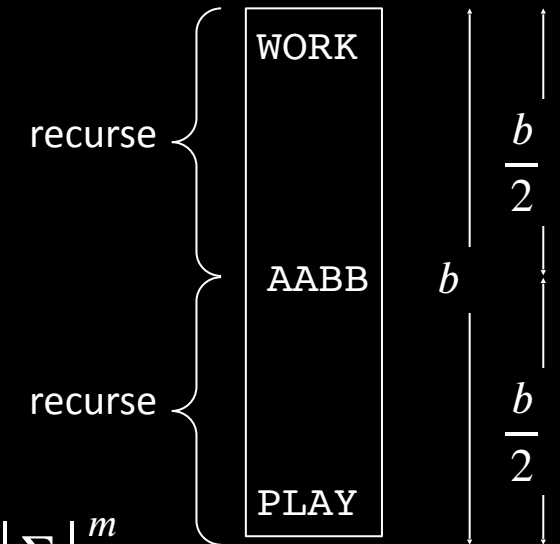
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

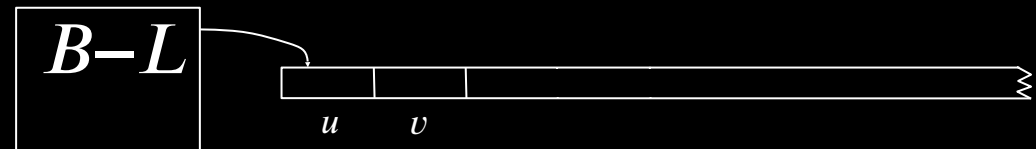
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:





# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

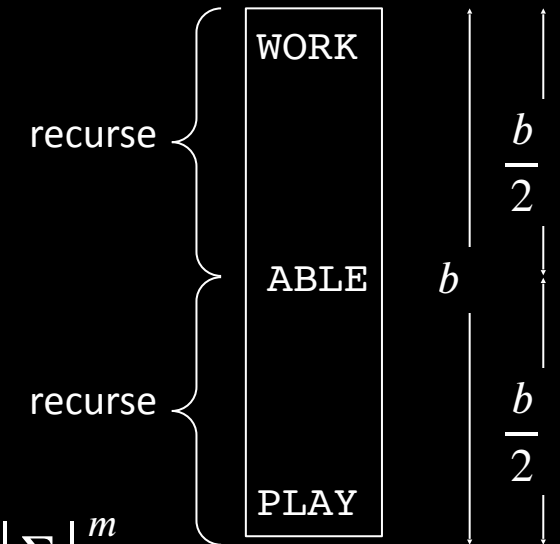
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

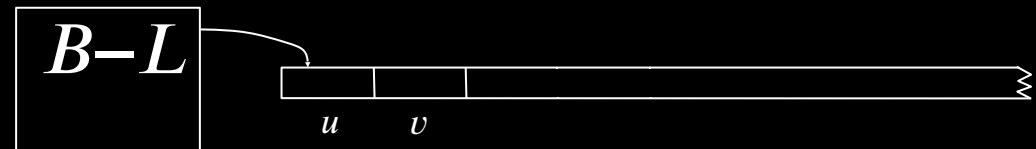
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

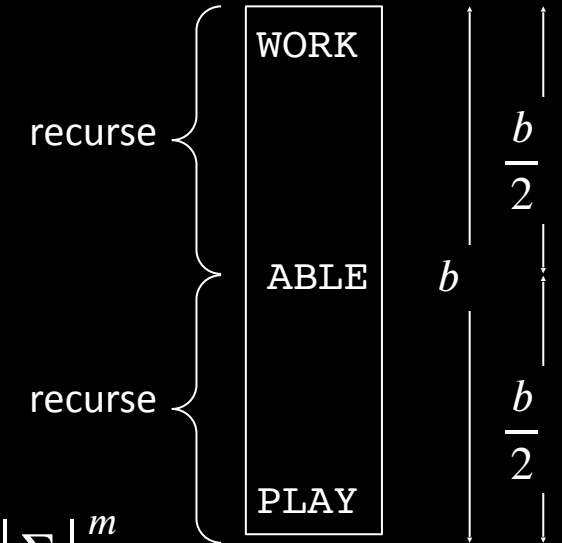
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

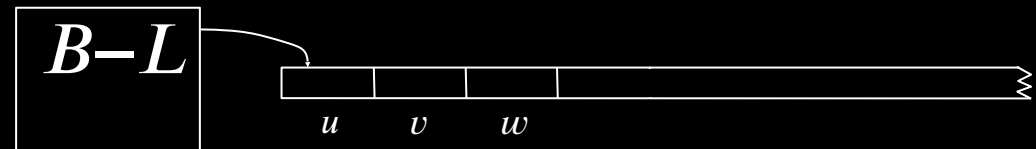
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

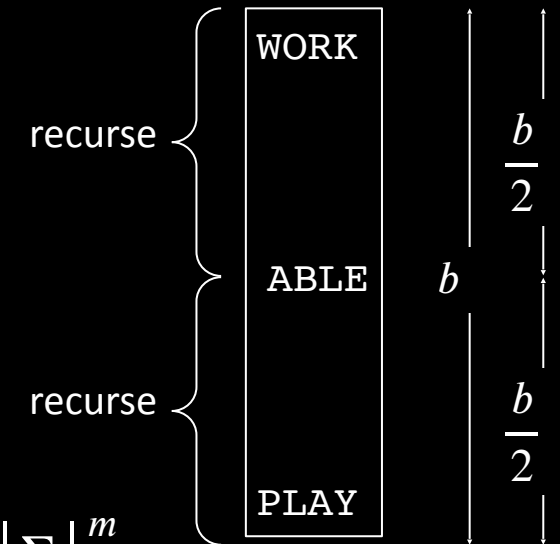
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

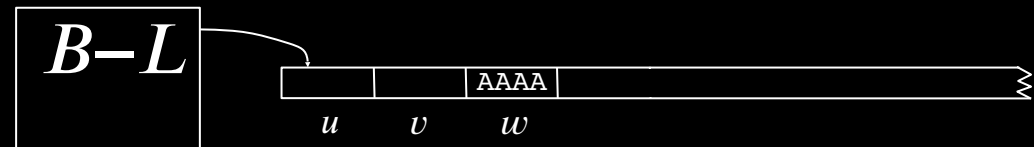
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

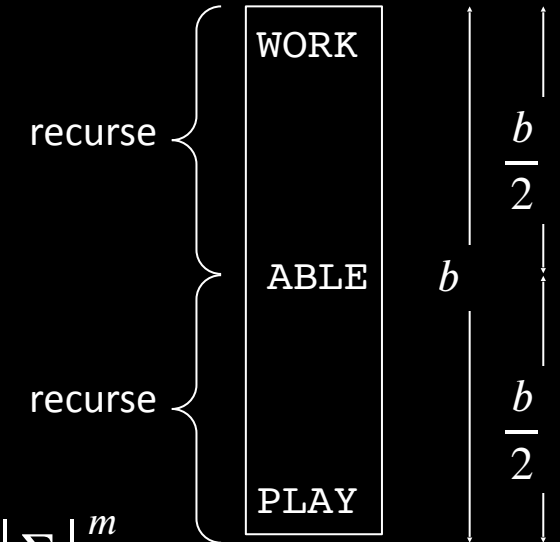
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

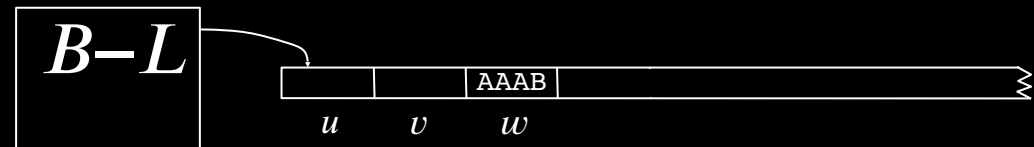
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

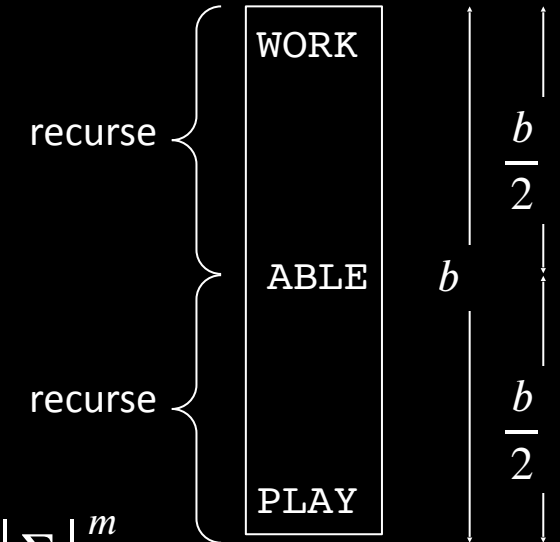
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

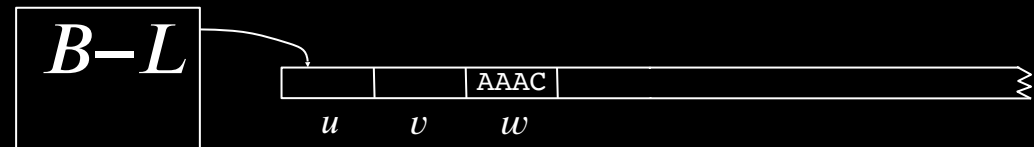
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

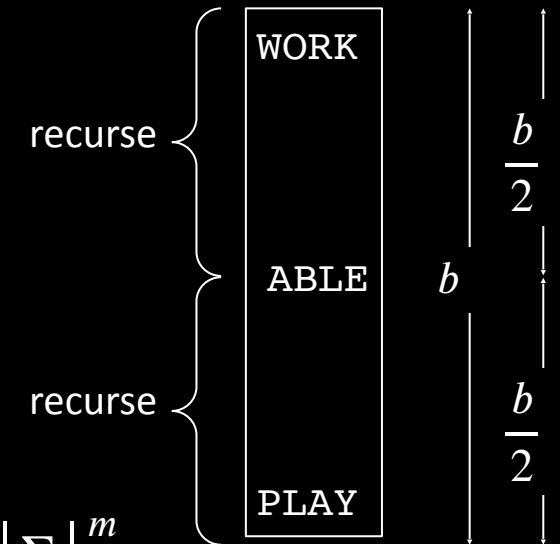
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

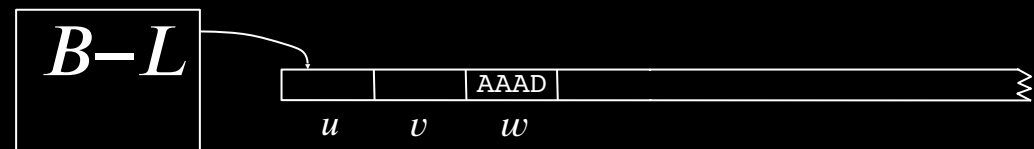
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

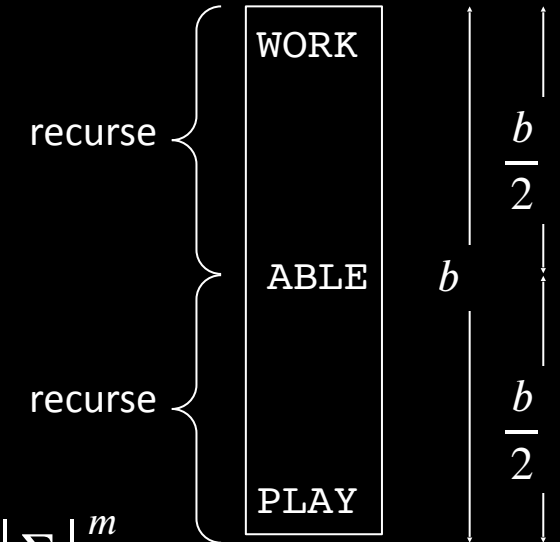
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

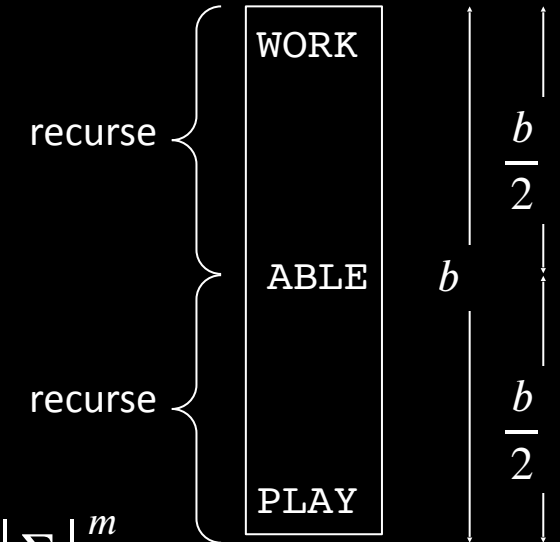
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

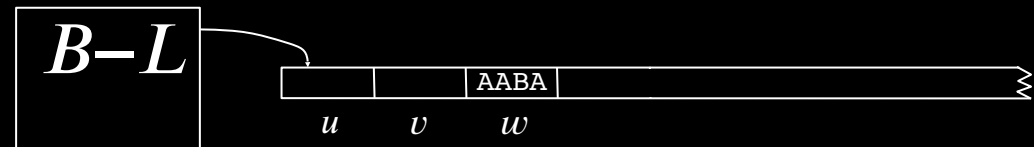
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:





# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

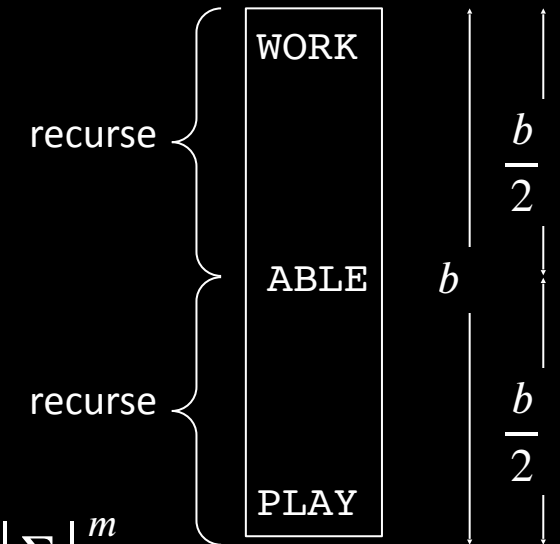
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

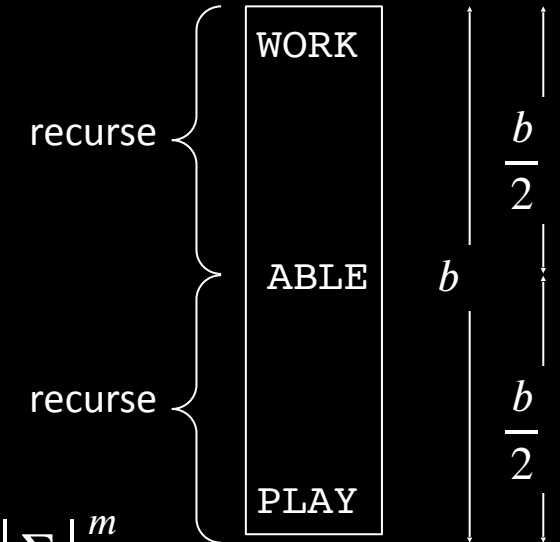
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

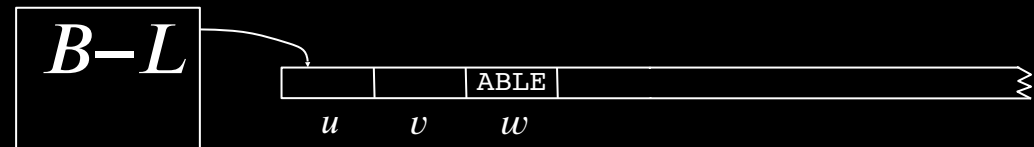
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: $LADDERDFA \in PSPACE$

Theorem:  $LADDERDFA \in SPACE(n^2)$

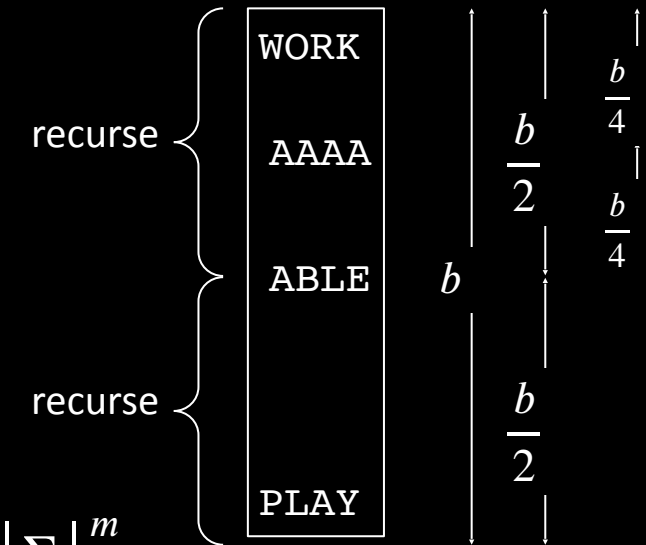
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED-LADDERDFA = \{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

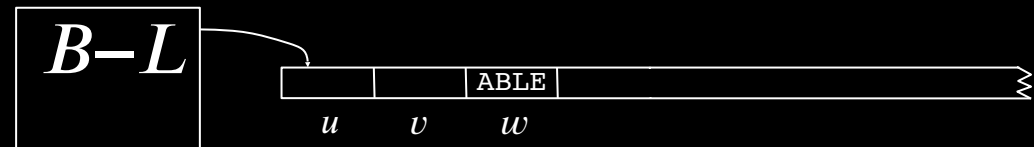
$B-L$  = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in LADDERDFA$  with  $B-L$  procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

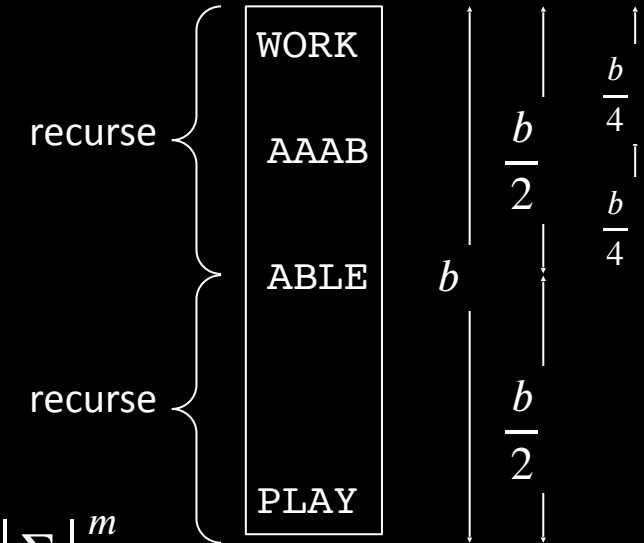
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

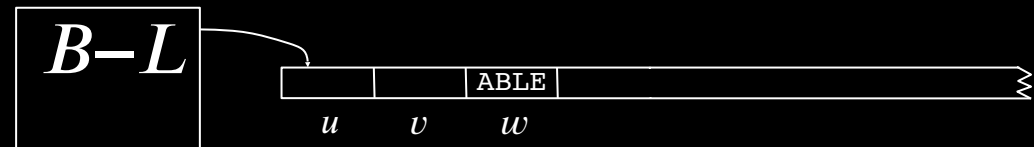
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

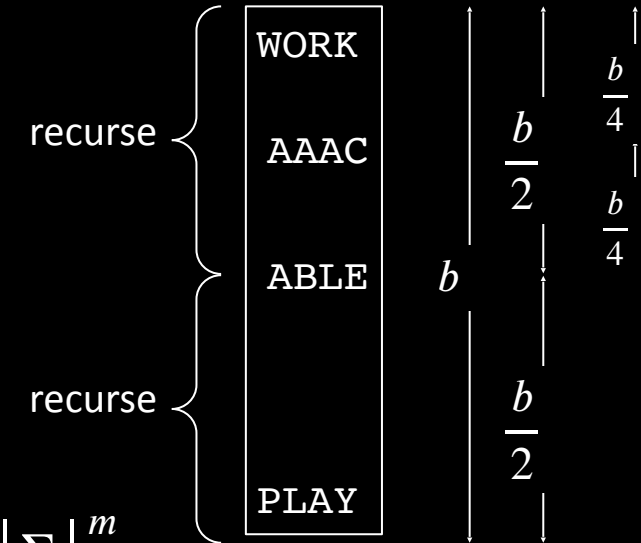
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

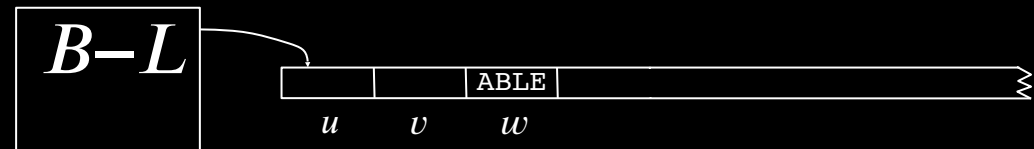
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

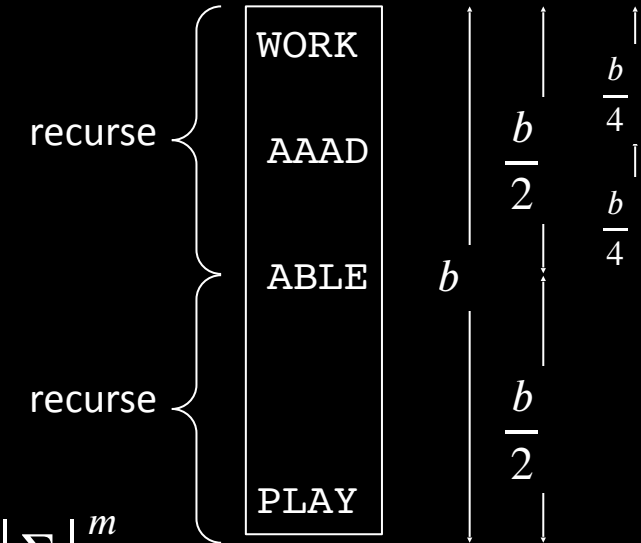
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

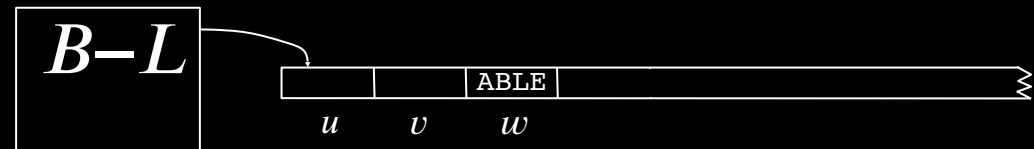
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: $LADDERDFA \in PSPACE$

Theorem:  $LADDERDFA \in SPACE(n^2)$

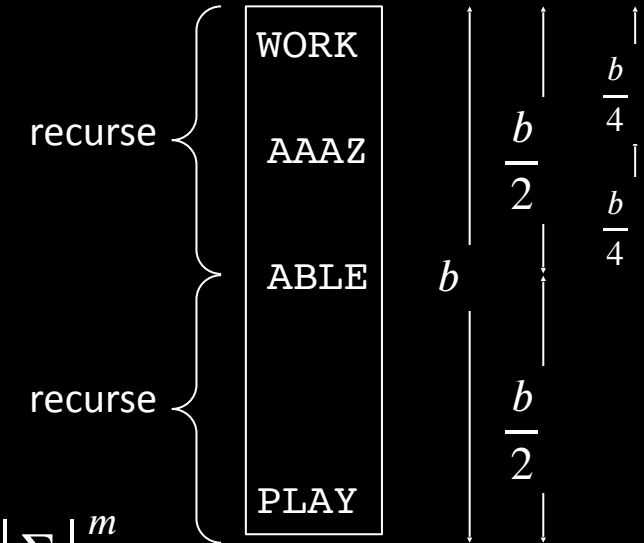
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED-LADDERDFA = \{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

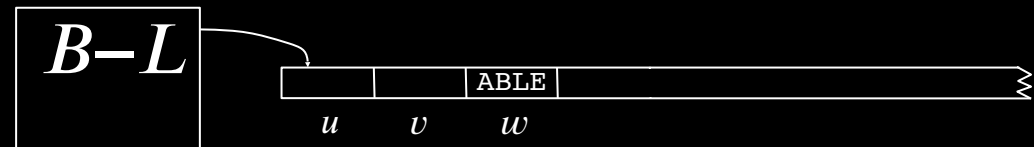
$B-L$  = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in LADDERDFA$  with  $B-L$  procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

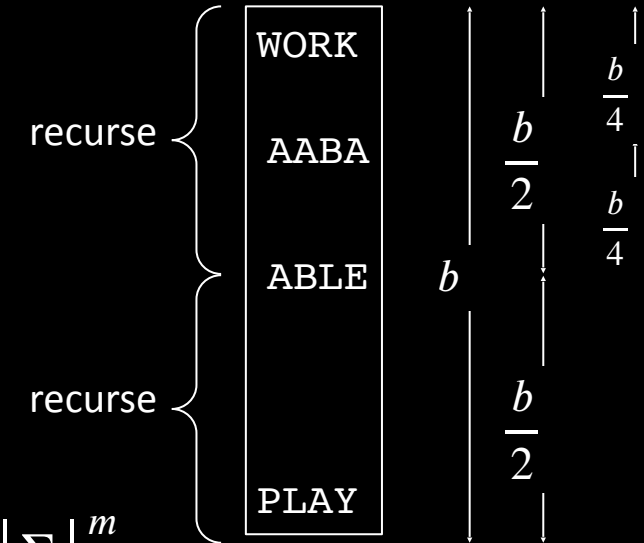
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

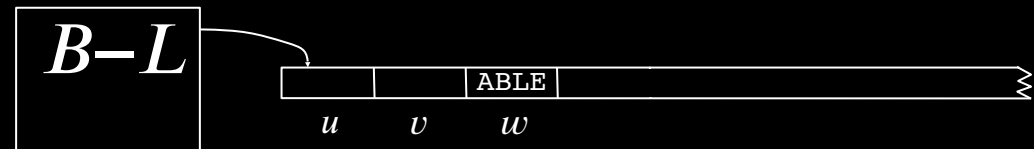
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:





# Review: $LADDERDFA \in PSPACE$

Theorem:  $LADDERDFA \in SPACE(n^2)$

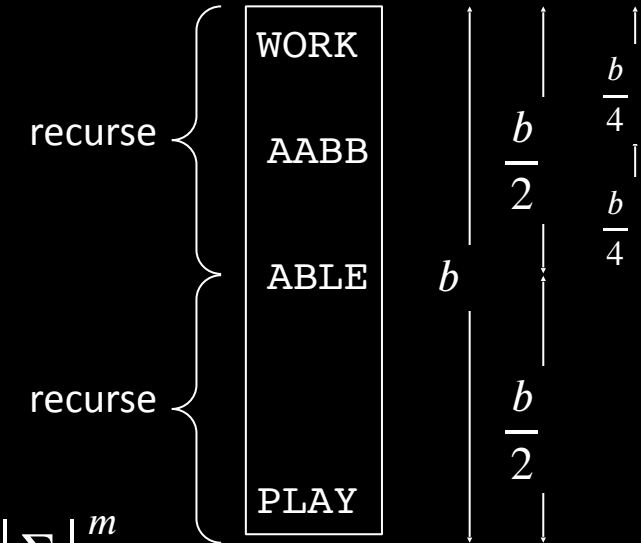
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED-LADDERDFA = \{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

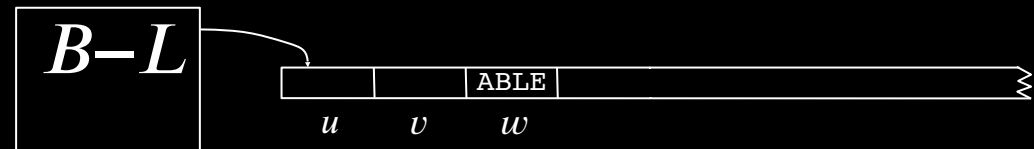
$B-L$  = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in LADDERDFA$  with  $B-L$  procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: $LADDERDFA \in PSPACE$

Theorem:  $LADDERDFA \in SPACE(n^2)$

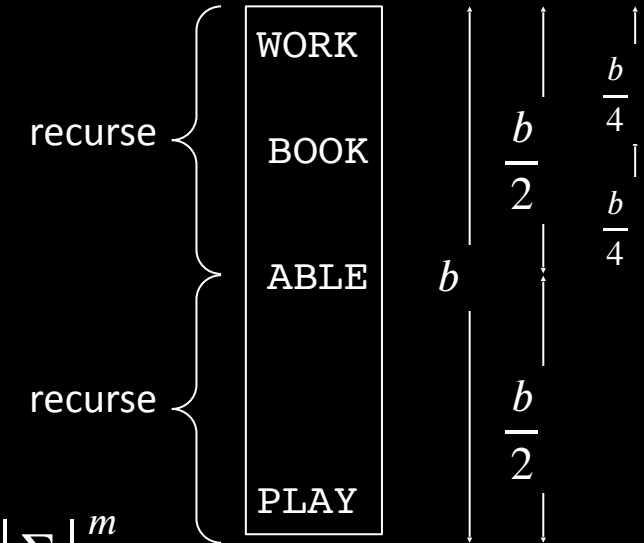
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED-LADDERDFA = \{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

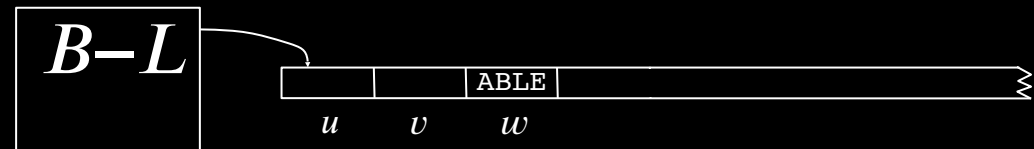
$B-L$  = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in LADDERDFA$  with  $B-L$  procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: $LADDERDFA \in PSPACE$

Theorem:  $LADDERDFA \in SPACE(n^2)$

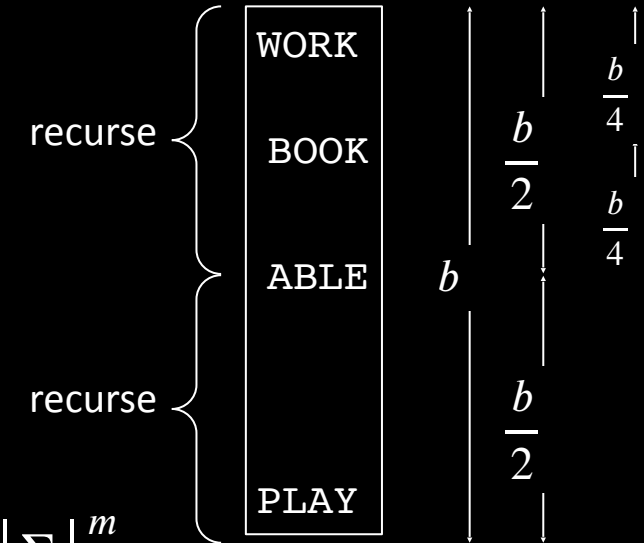
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED-LADDERDFA = \{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

$B-L$  = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in LADDERDFA$  with  $B-L$  procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

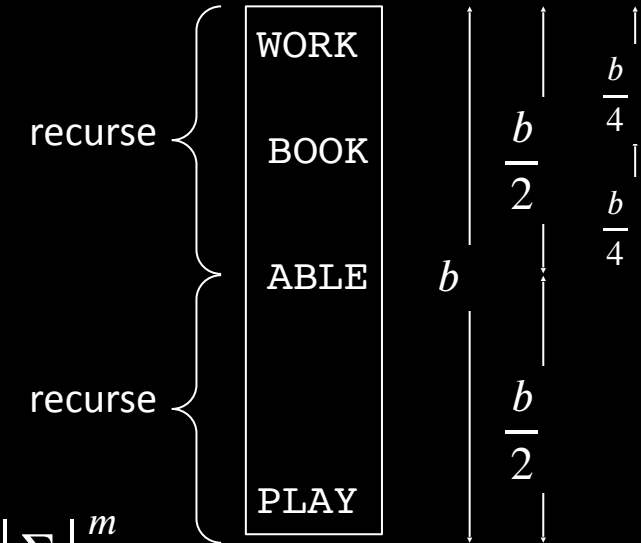
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

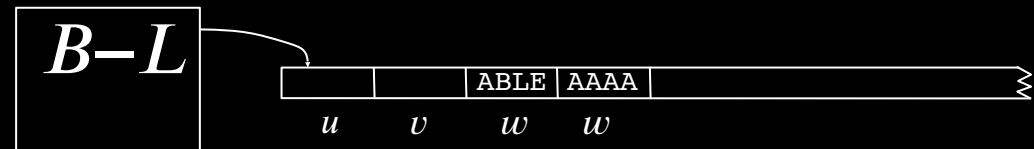
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

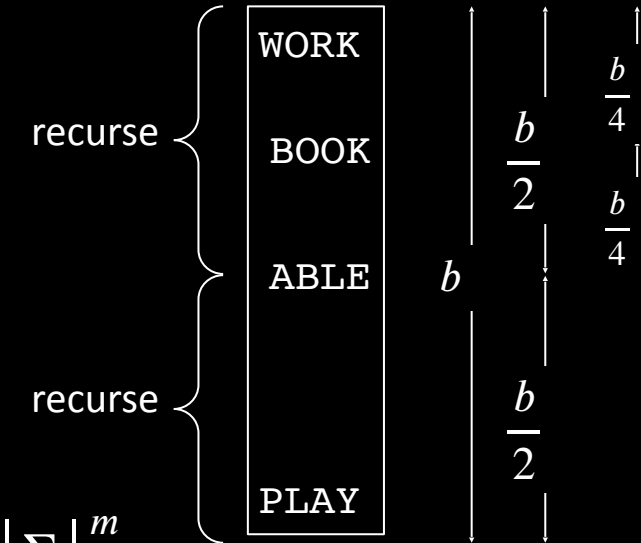
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

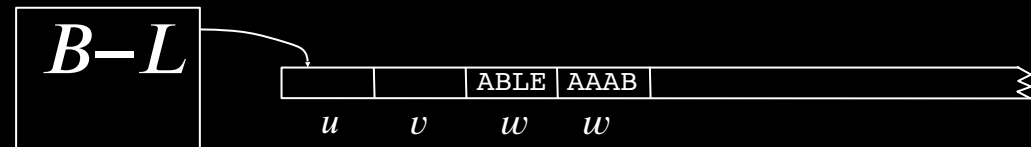
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: $LADDERDFA \in PSPACE$

Theorem:  $LADDERDFA \in SPACE(n^2)$

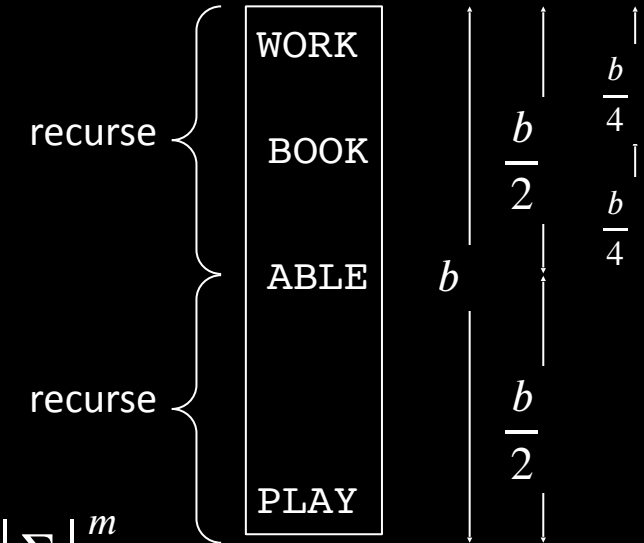
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED-LADDERDFA = \{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

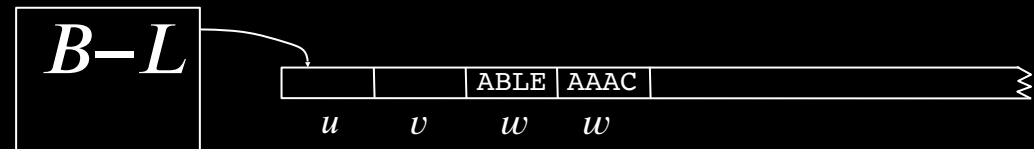
$B-L$  = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in LADDERDFA$  with  $B-L$  procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

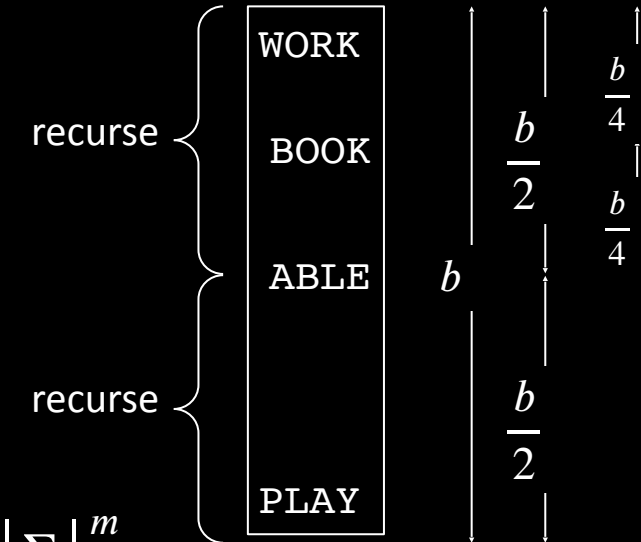
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

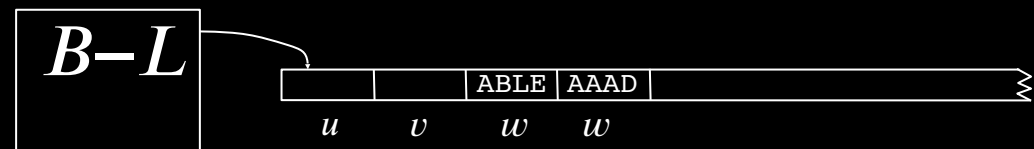
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: $LADDERDFA \in PSPACE$

Theorem:  $LADDERDFA \in SPACE(n^2)$

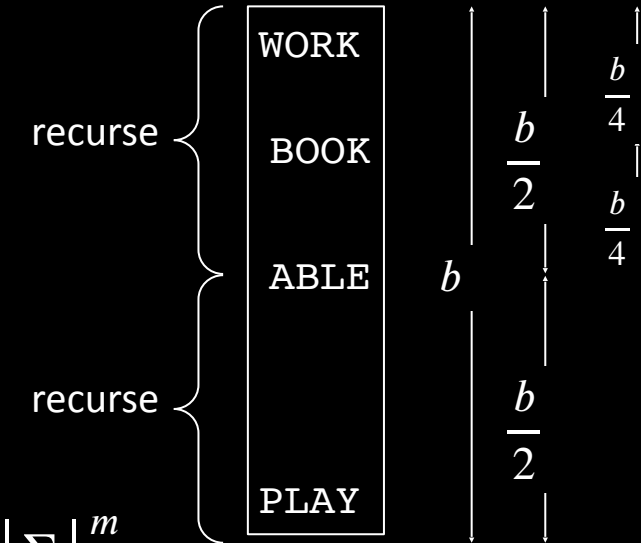
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED-LADDERDFA = \{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

$B-L$  = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in LADDERDFA$  with  $B-L$  procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:





# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

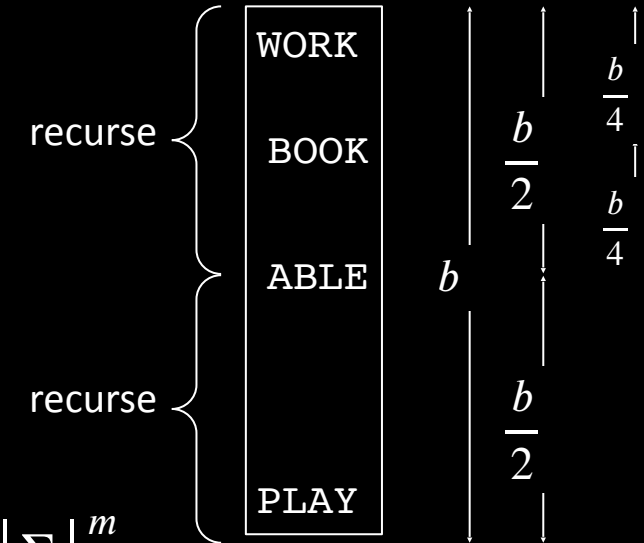
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

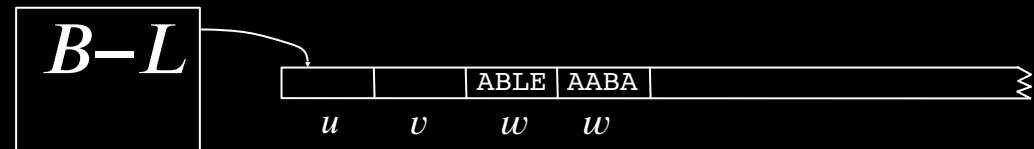
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

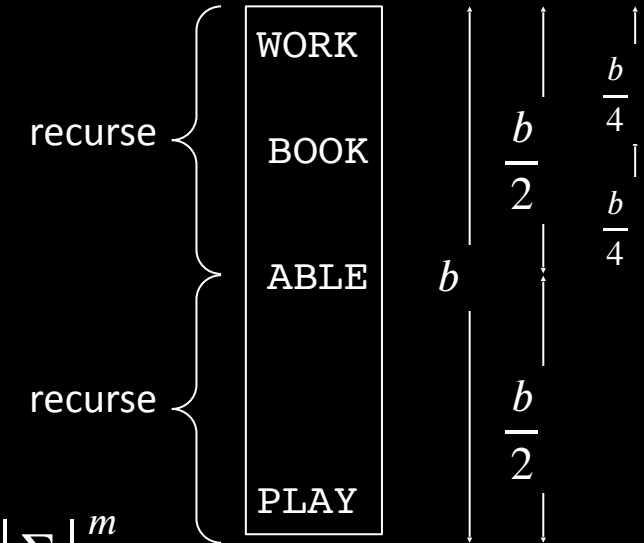
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

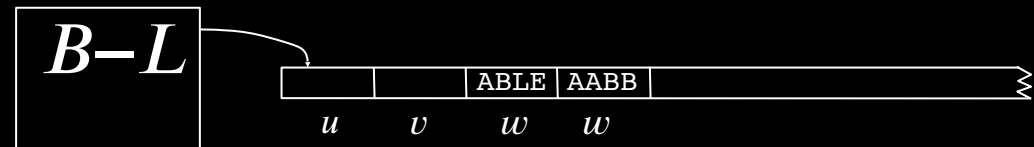
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

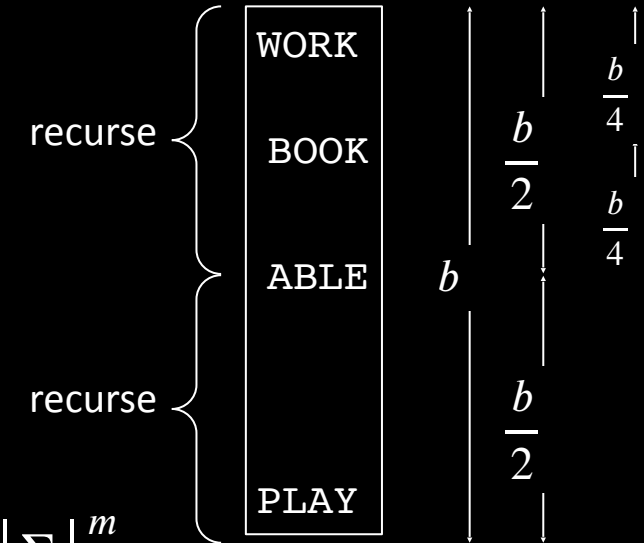
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

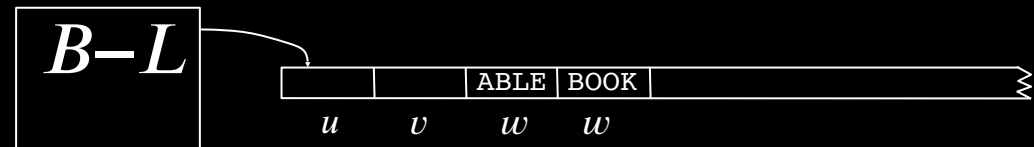
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

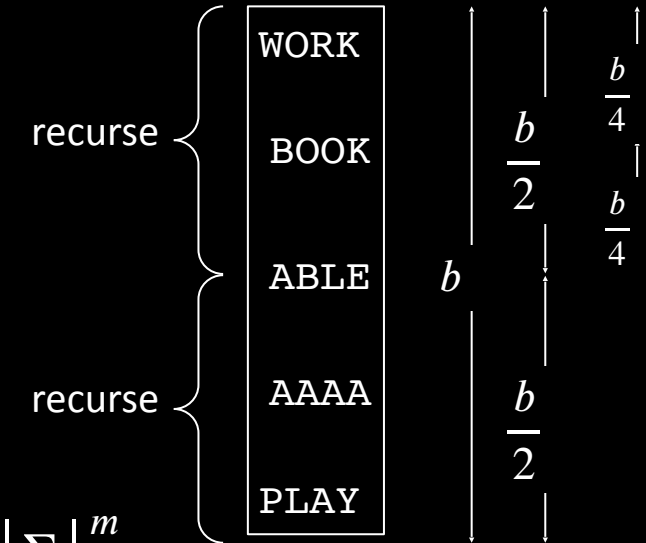
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

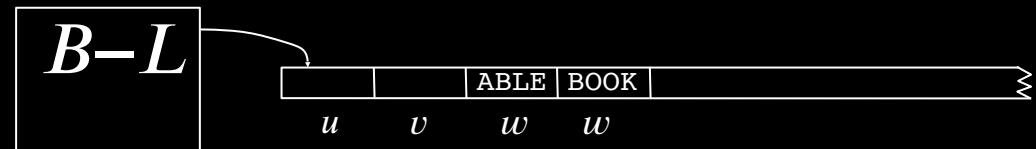
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

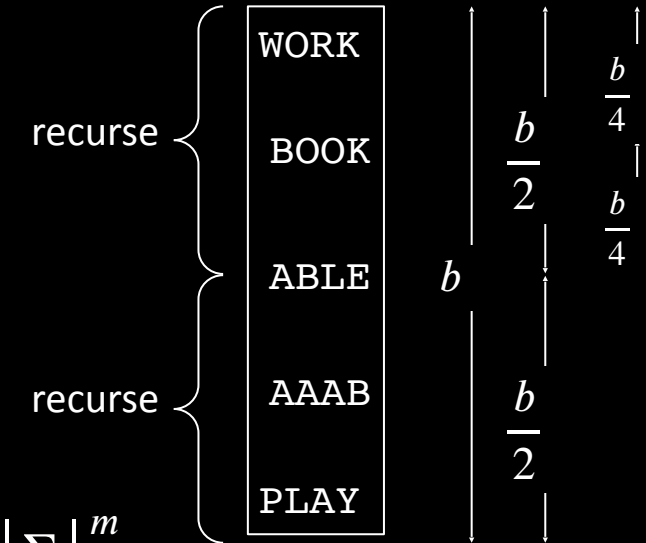
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

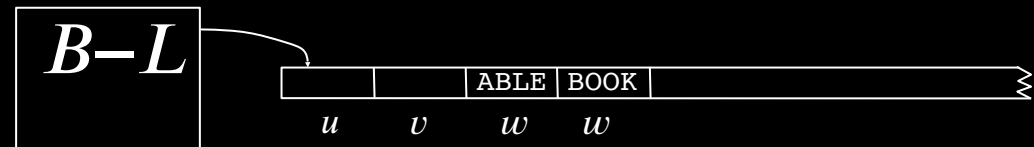
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

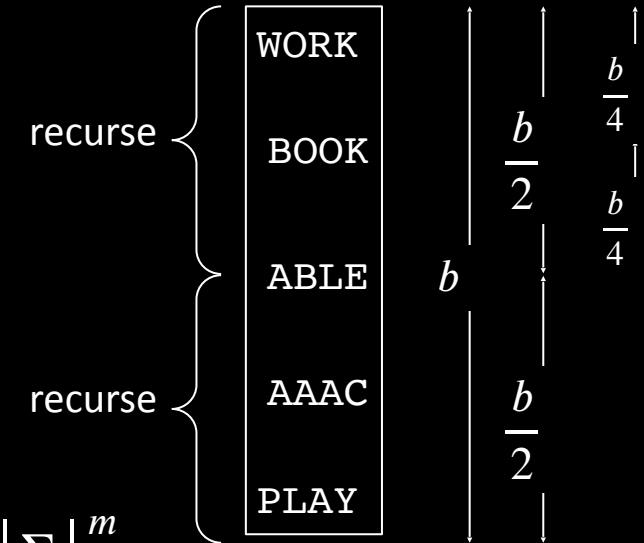
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

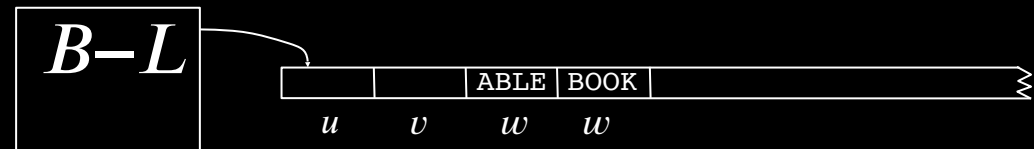
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

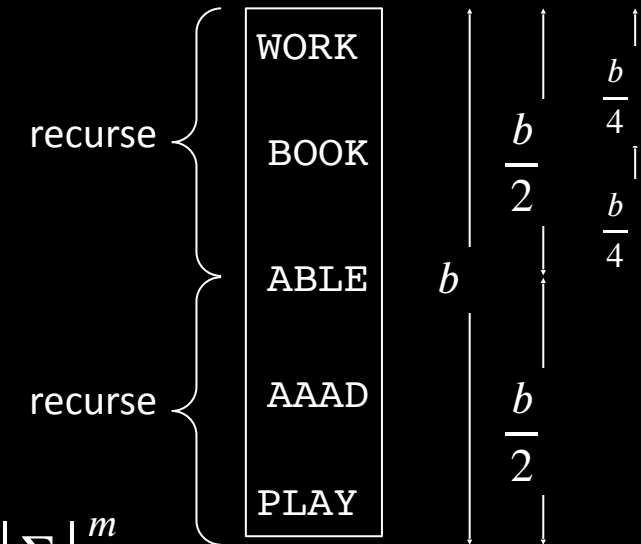
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

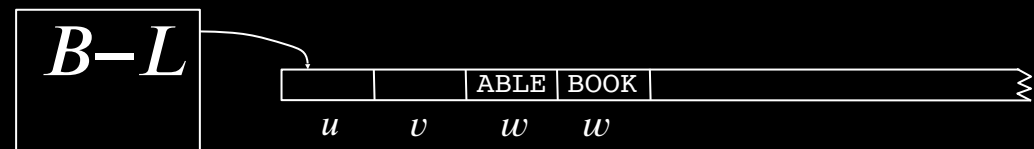
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

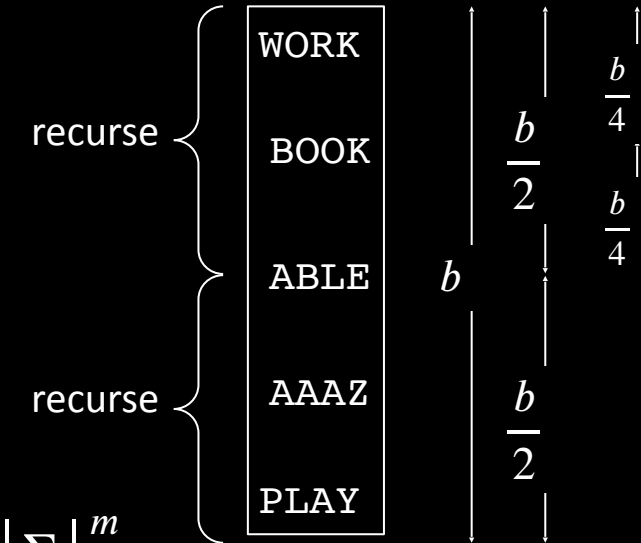
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

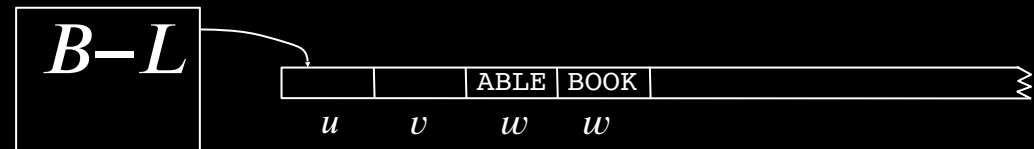
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:





# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

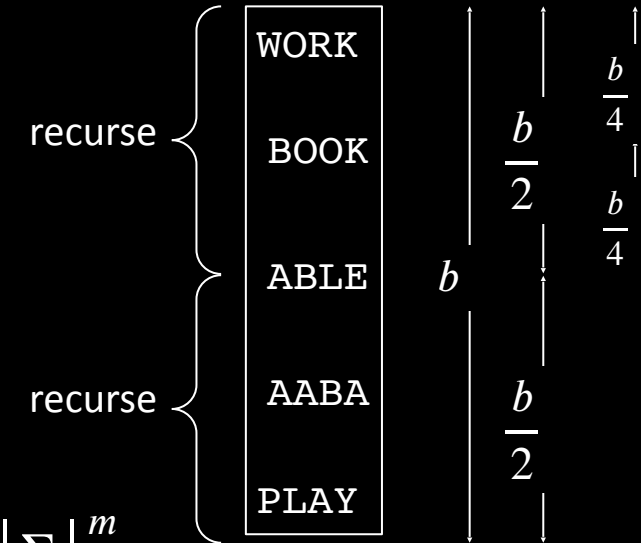
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

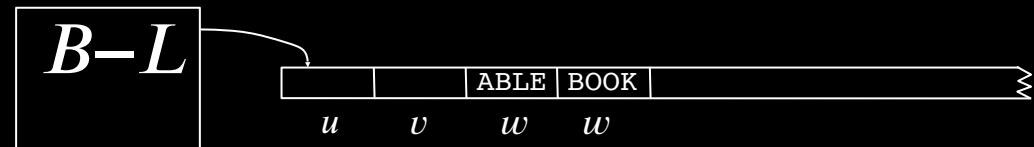
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

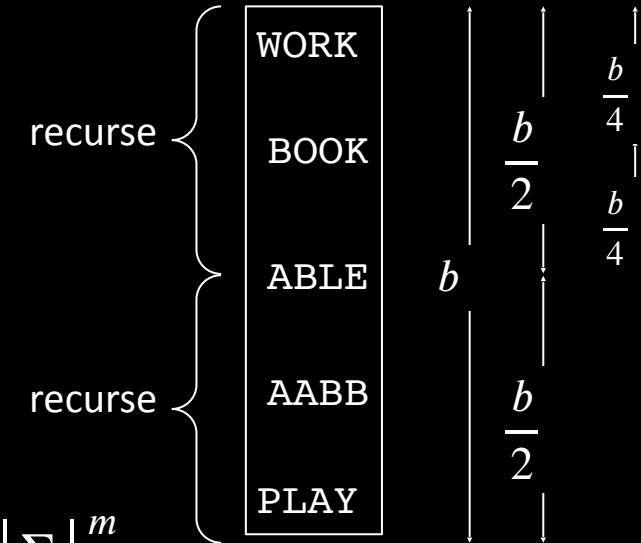
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

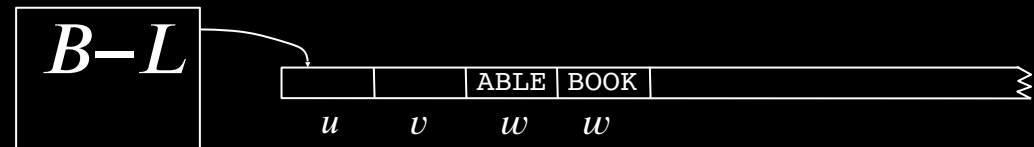
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

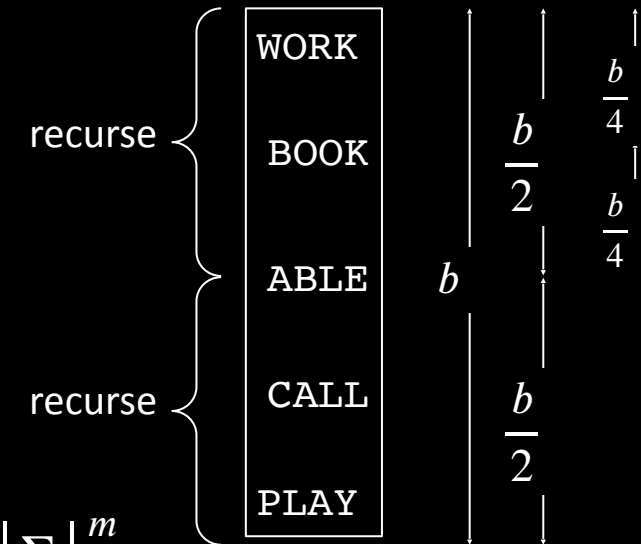
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

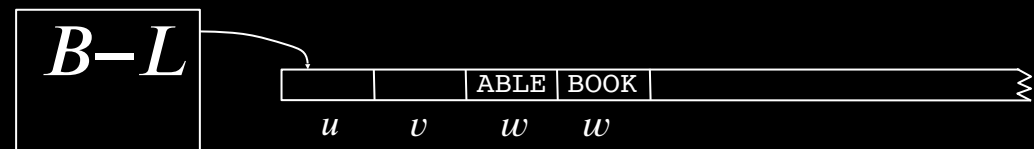
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

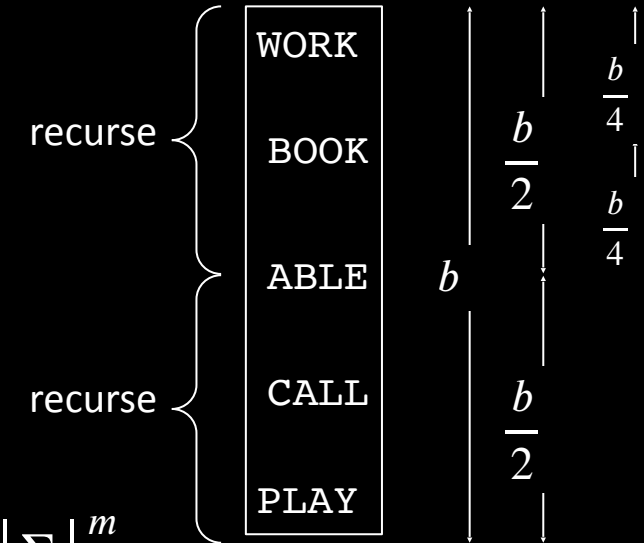
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

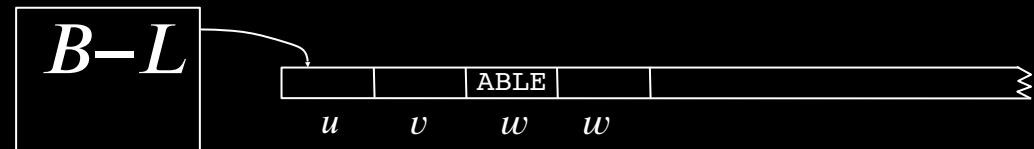
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

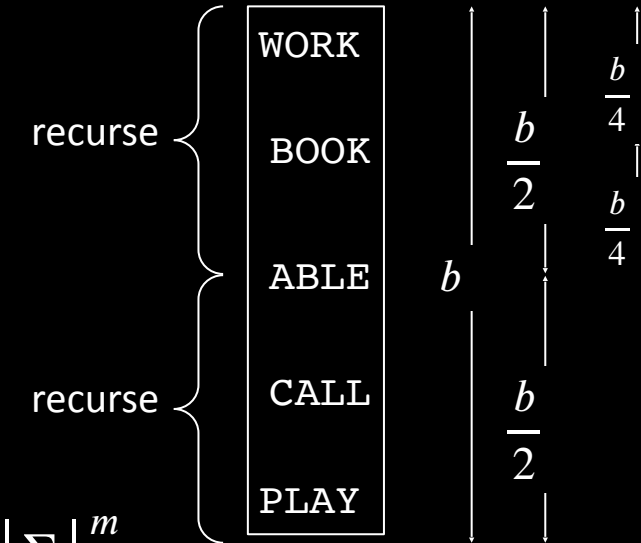
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

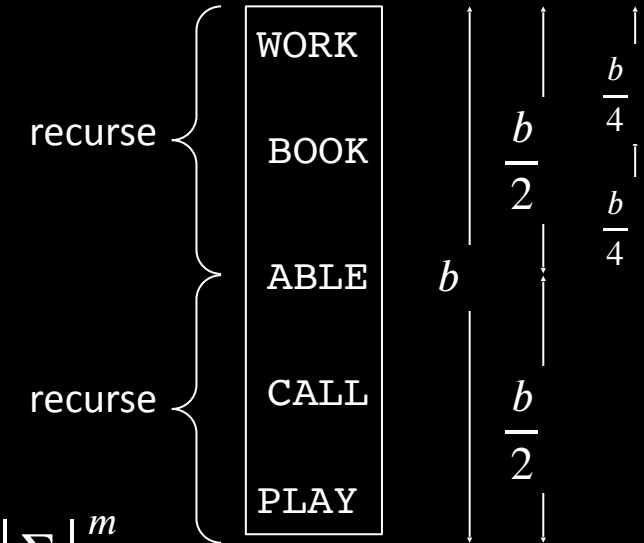
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

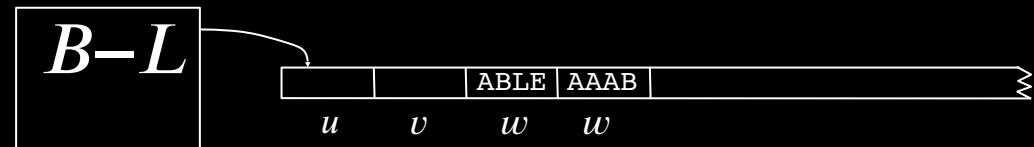
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

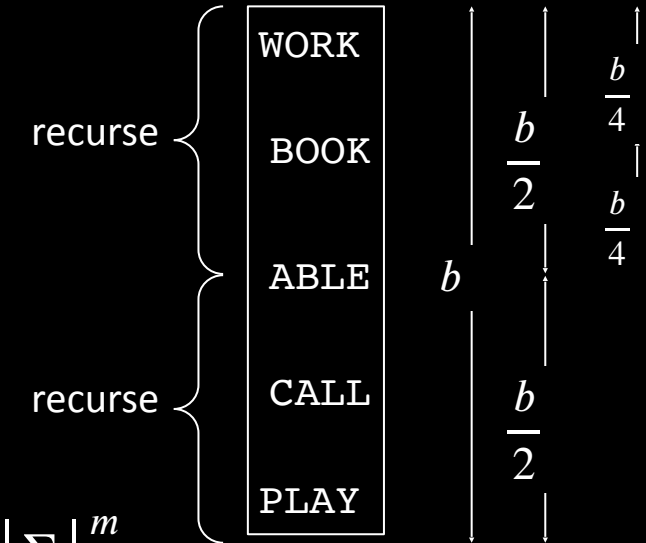
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

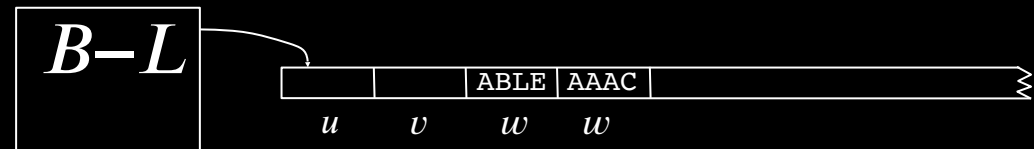
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

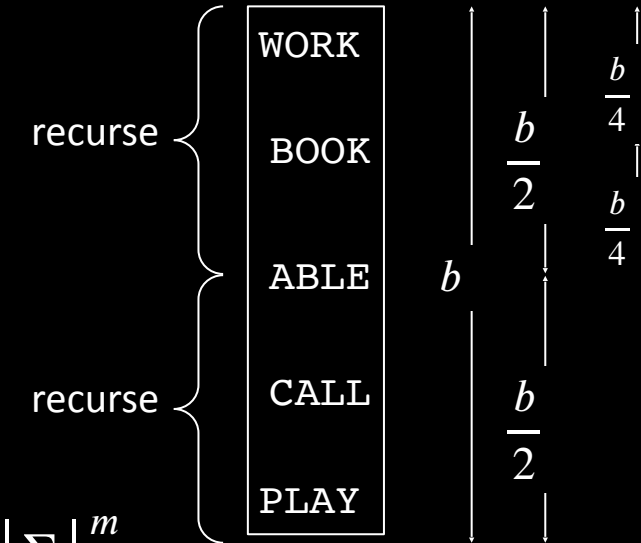
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:





# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

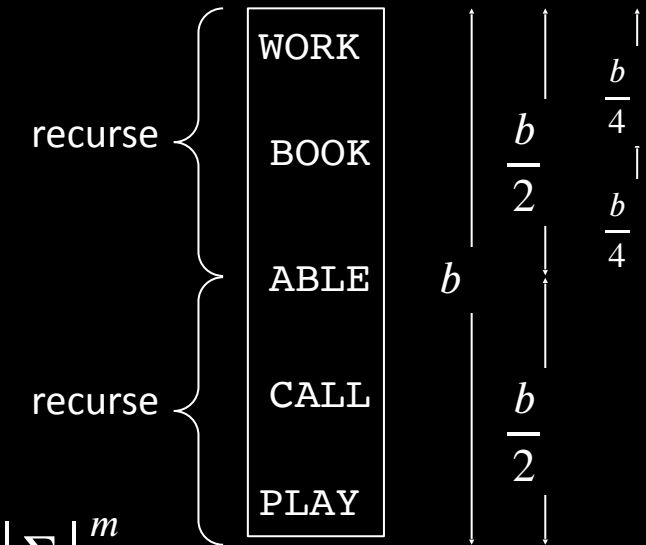
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

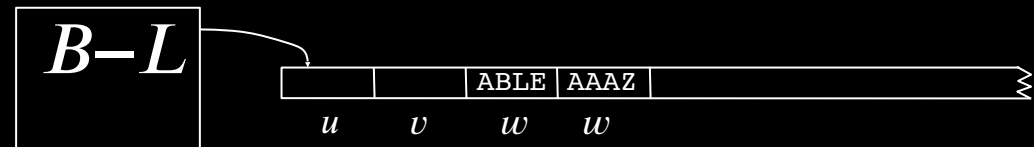
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

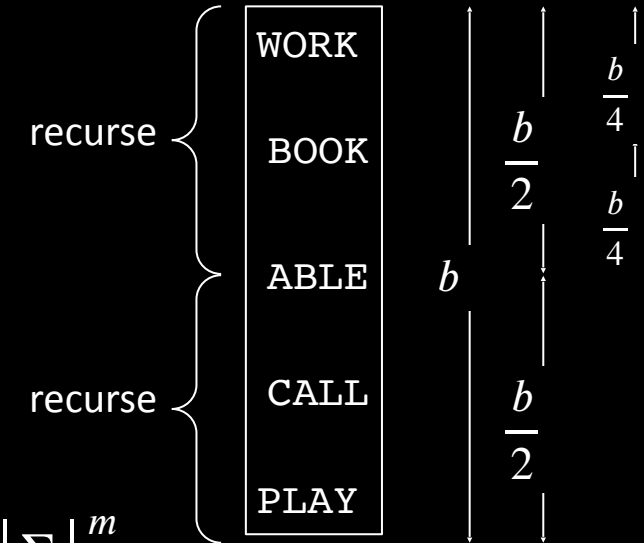
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

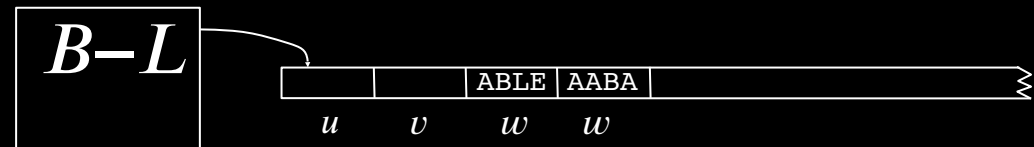
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

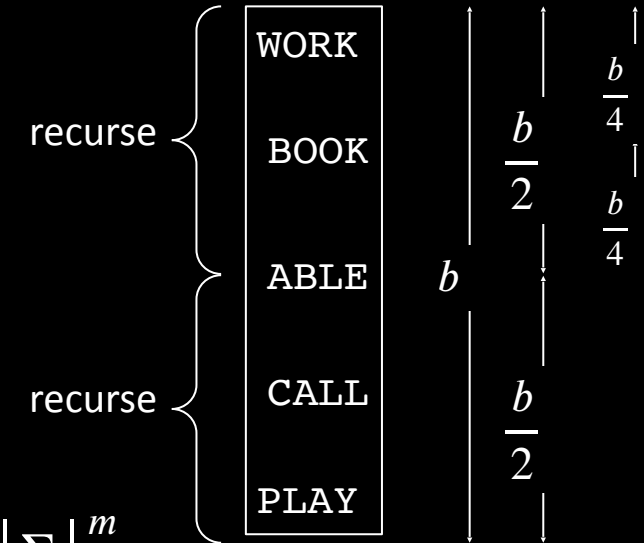
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

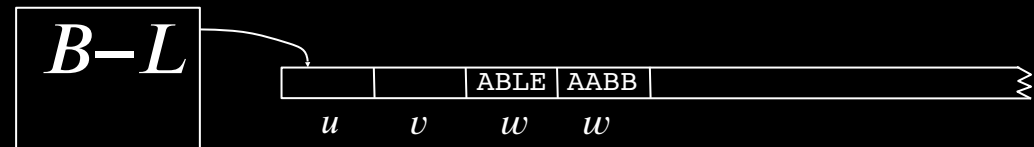
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

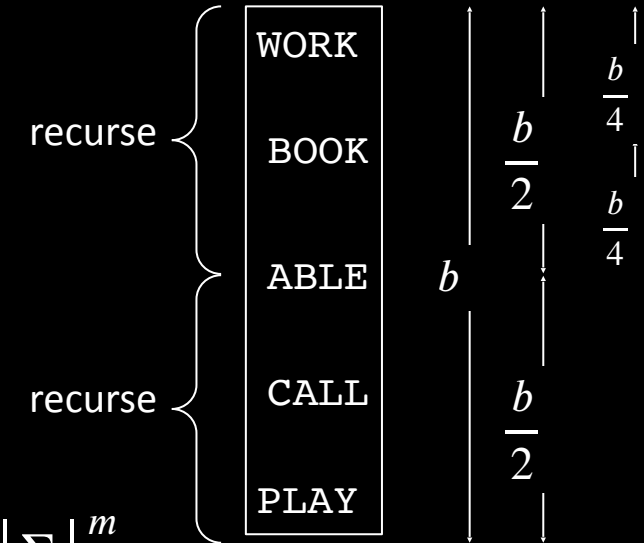
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

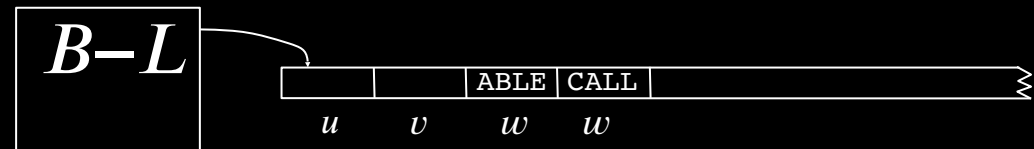
*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

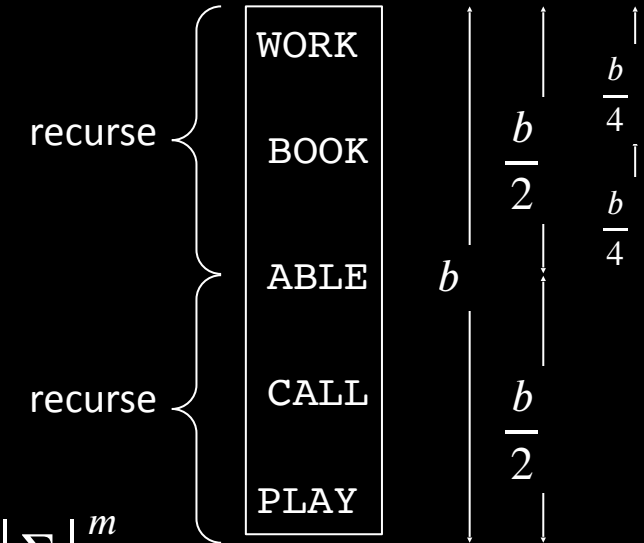
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3.     Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4.     *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:

Each recursive level uses space  $O(n)$  (to record  $w$ ).



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

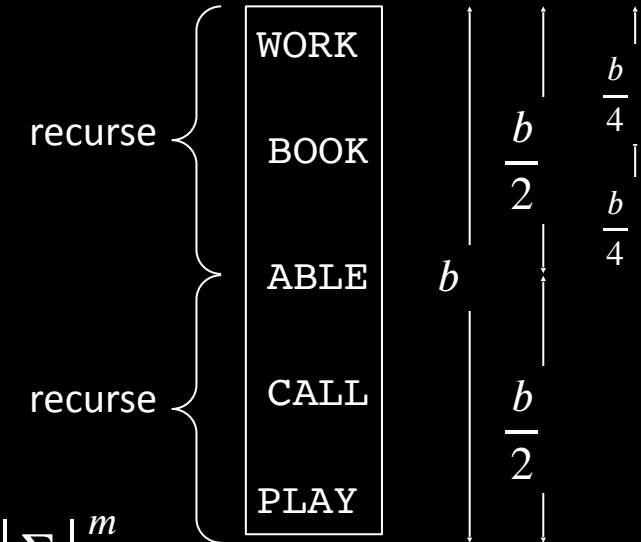
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:

Each recursive level uses space  $O(n)$  (to record  $w$ ).

Recursion depth is  $\log t = O(m) = O(n)$ .



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

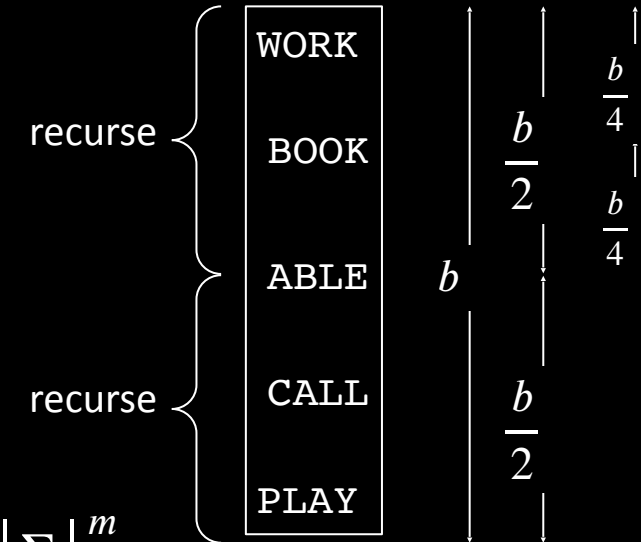
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:

Each recursive level uses space  $O(n)$  (to record  $w$ ).

Recursion depth is  $\log t = O(m) = O(n)$ .

Total space used is  $O(n^2)$ .



# Review: *LADDER*DFA $\in$ PSPACE

Theorem: *LADDER*DFA  $\in$  SPACE( $n^2$ )

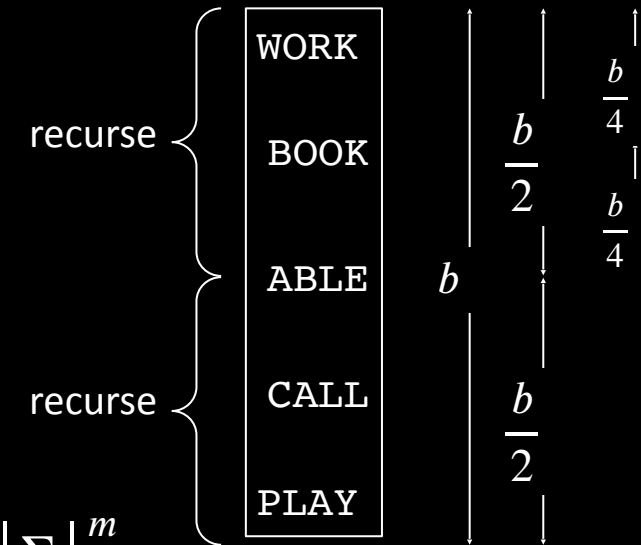
Proof: Write  $u \xrightarrow{b} v$  if there's a ladder from  $u$  to  $v$  of length  $\leq b$ .

Here's a recursive procedure to solve the bounded DFA ladder problem:

*BOUNDED-LADDER*DFA =  $\{ \langle B, u, v, b \rangle \mid B \text{ a DFA and } u \xrightarrow{b} v \text{ by a ladder in } L(B) \}$

*B-L* = "On input  $\langle B, u, v, b \rangle$  Let  $m = |u| = |v|$ .

1. For  $b = 1$ , *accept* if  $u, v \in L(B)$  and differ in  $\leq 1$  place, else *reject*.
2. For  $b > 1$ , repeat for each  $w \in L(B)$  of length  $|u|$
3. Recursively test  $u \xrightarrow{b/2} w$  and  $w \xrightarrow{b/2} v$  [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."



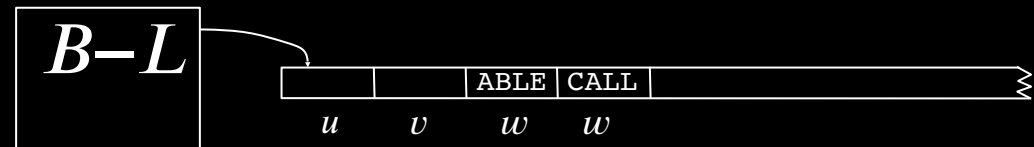
Test  $\langle B, u, v \rangle \in$  *LADDER*DFA with *B-L* procedure on input  $\langle B, u, v, t \rangle$  for  $t = |\Sigma|^m$

Space analysis:

Each recursive level uses space  $O(n)$  (to record  $w$ ).

Recursion depth is  $\log t = O(m) = O(n)$ .

Total space used is  $O(n^2)$ .





# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

Give recursive algorithm to test  $c_i \xrightarrow{b} c_j$ :

# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

Give recursive algorithm to test  $c_i \xrightarrow{b} c_j$ :

$M$  = "On input  $c_i, c_j, b$  [goal is to check  $c_i \xrightarrow{b} c_j$ ]

# PSPACE = NPSPACE

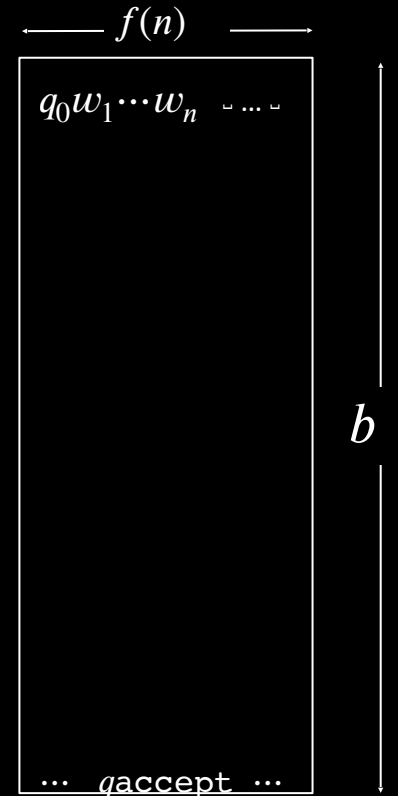
**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

Give recursive algorithm to test  $c_i \xrightarrow{b} c_j$ :

$M$  = "On input  $c_i, c_j, b$  [goal is to check  $c_i \xrightarrow{b} c_j$ ]



# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

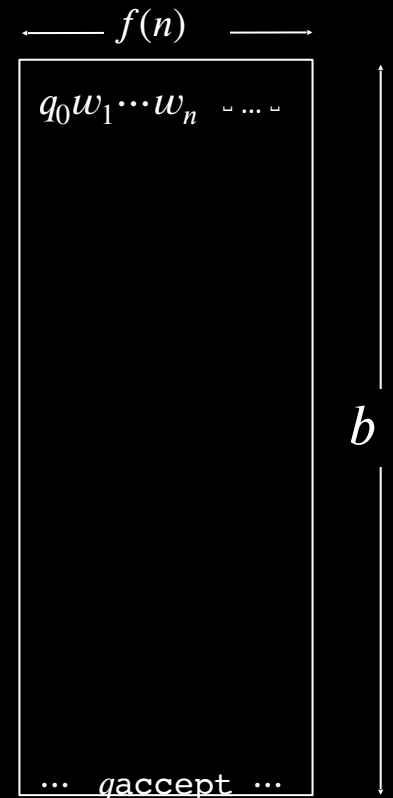
Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

Give recursive algorithm to test  $c_i \xrightarrow{b} c_j$ :

$M$  = "On input  $c_i, c_j, b$  [goal is to check  $c_i \xrightarrow{b} c_j$ ]

1. If  $b = 1$ , check directly by using  $N$ 's program and answer accordingly.



# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

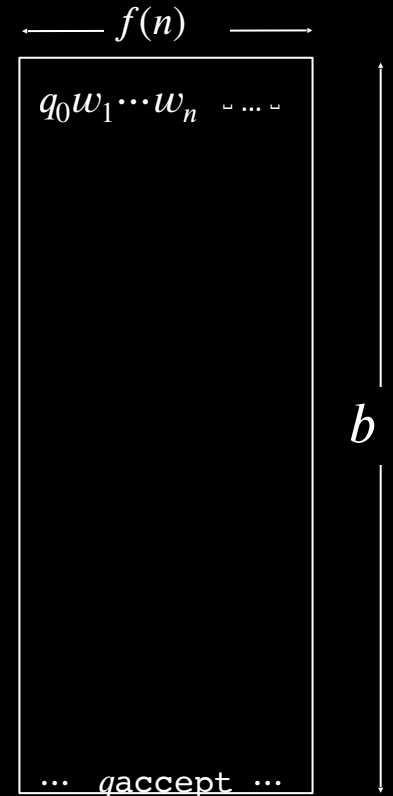
Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

Give recursive algorithm to test  $c_i \xrightarrow{b} c_j$ :

$M$  = "On input  $c_i, c_j, b$  [goal is to check  $c_i \xrightarrow{b} c_j$ ]

1. If  $b = 1$ , check directly by using  $N$ 's program and answer accordingly.
2. If  $b > 1$ , repeat for all configurations  $c_{\text{mid}}$  that use  $f(n)$  space.





# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

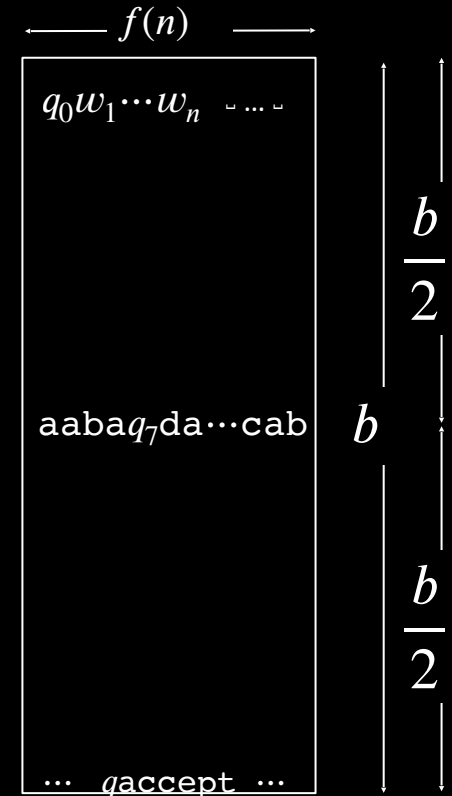
Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

Give recursive algorithm to test  $c_i \xrightarrow{b} c_j$ :

$M$  = "On input  $c_i, c_j, b$  [goal is to check  $c_i \xrightarrow{b} c_j$ ]

1. If  $b = 1$ , check directly by using  $N$ 's program and answer accordingly.
2. If  $b > 1$ , repeat for all configurations  $c_{\text{mid}}$  that use  $f(n)$  space.



# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

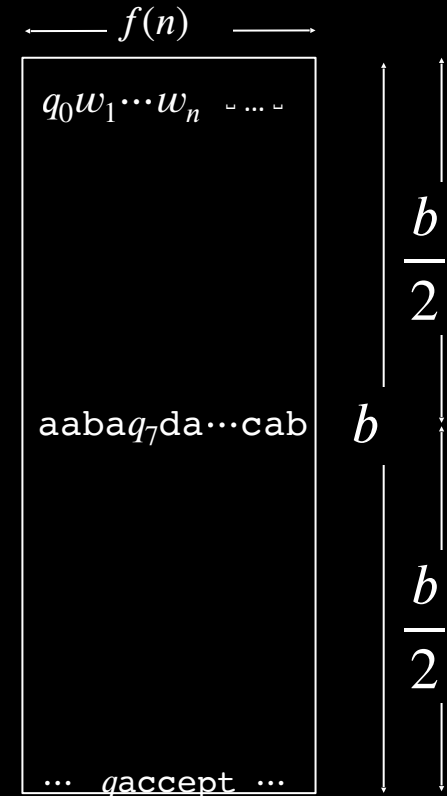
Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

Give recursive algorithm to test  $c_i \xrightarrow{b} c_j$ :

$M$  = "On input  $c_i, c_j, b$  [goal is to check  $c_i \xrightarrow{b} c_j$ ]

1. If  $b = 1$ , check directly by using  $N$ 's program and answer accordingly.
2. If  $b > 1$ , repeat for all configurations  $c_{\text{mid}}$  that use  $f(n)$  space.
3. Recursively test  $c_i \xrightarrow{b/2} c_{\text{mid}}$  and  $c_{\text{mid}} \xrightarrow{b/2} c_j$



# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

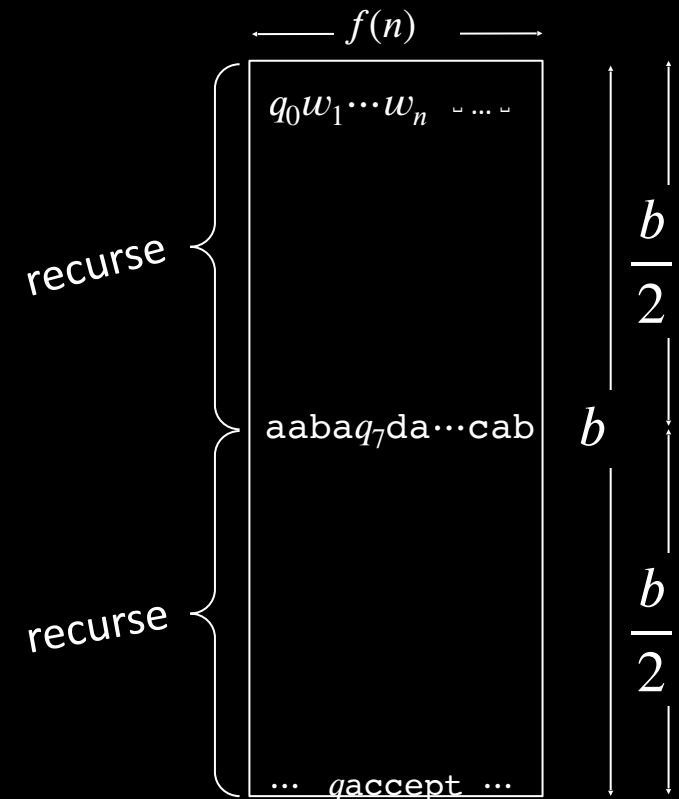
Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

Give recursive algorithm to test  $c_i \xrightarrow{b} c_j$ :

$M$  = "On input  $c_i, c_j, b$  [goal is to check  $c_i \xrightarrow{b} c_j$ ]

1. If  $b = 1$ , check directly by using  $N$ 's program and answer accordingly.
2. If  $b > 1$ , repeat for all configurations  $c_{\text{mid}}$  that use  $f(n)$  space.
3. Recursively test  $c_i \xrightarrow{b/2} c_{\text{mid}}$  and  $c_{\text{mid}} \xrightarrow{b/2} c_j$



# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

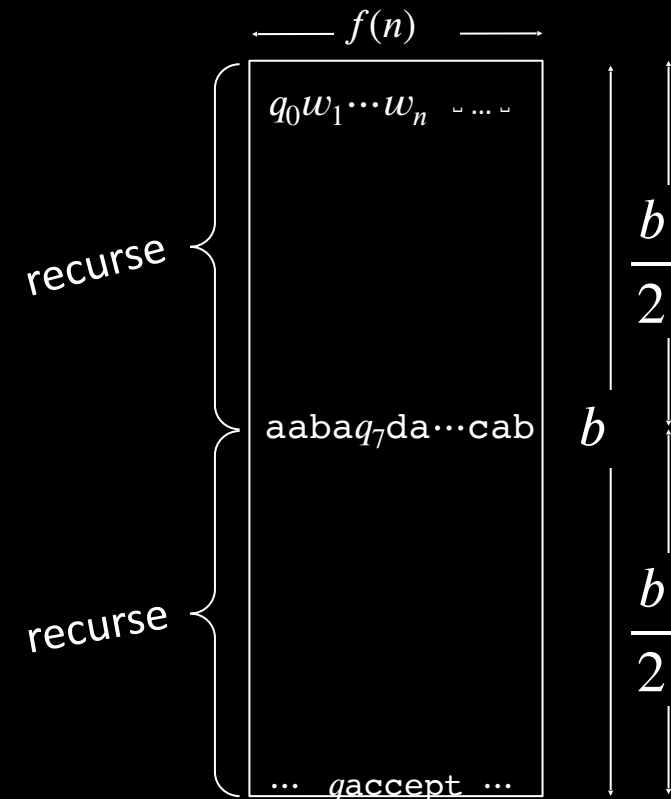
Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

Give recursive algorithm to test  $c_i \xrightarrow{b} c_j$ :

$M$  = "On input  $c_i, c_j, b$  [goal is to check  $c_i \xrightarrow{b} c_j$ ]

1. If  $b = 1$ , check directly by using  $N$ 's program and answer accordingly.
2. If  $b > 1$ , repeat for all configurations  $c_{\text{mid}}$  that use  $f(n)$  space.
3. Recursively test  $c_i \xrightarrow{b/2} c_{\text{mid}}$  and  $c_{\text{mid}} \xrightarrow{b/2} c_j$
4. If both are true, *accept*. If not, continue.



# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

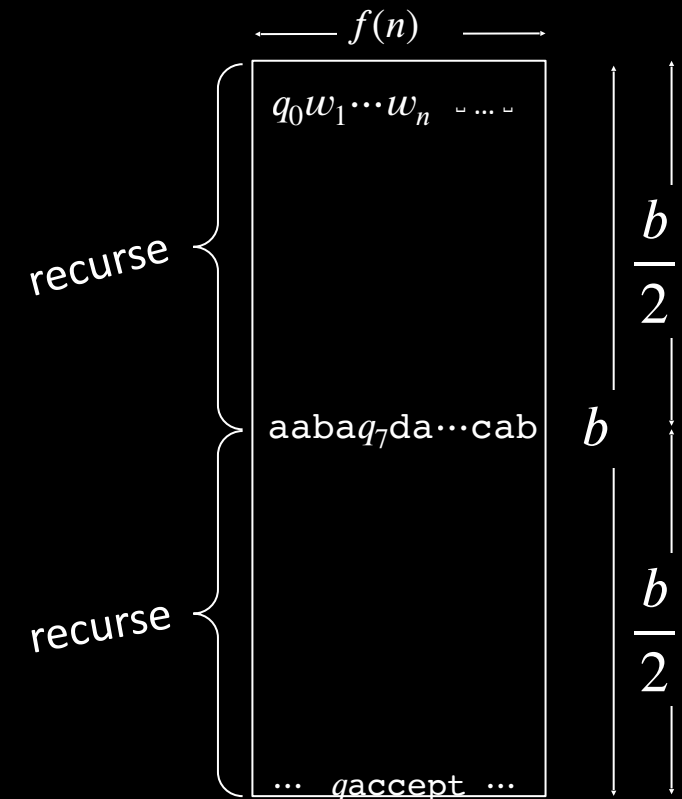
Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

Give recursive algorithm to test  $c_i \xrightarrow{b} c_j$ :

$M$  = "On input  $c_i, c_j, b$  [goal is to check  $c_i \xrightarrow{b} c_j$ ]

1. If  $b = 1$ , check directly by using  $N$ 's program and answer accordingly.
2. If  $b > 1$ , repeat for all configurations  $c_{\text{mid}}$  that use  $f(n)$  space.
3. Recursively test  $c_i \xrightarrow{b/2} c_{\text{mid}}$  and  $c_{\text{mid}} \xrightarrow{b/2} c_j$
4. If both are true, *accept*. If not, continue.
5. *Reject* if haven't yet accepted."



# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

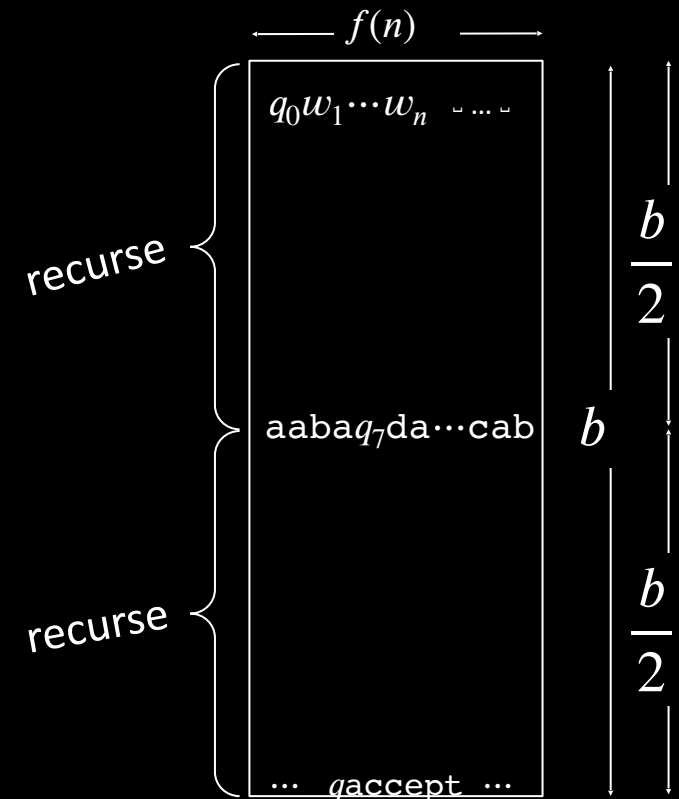
For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

Give recursive algorithm to test  $c_i \xrightarrow{b} c_j$ :

$M$  = "On input  $c_i, c_j, b$  [goal is to check  $c_i \xrightarrow{b} c_j$ ]

1. If  $b = 1$ , check directly by using  $N$ 's program and answer accordingly.
2. If  $b > 1$ , repeat for all configurations  $c_{\text{mid}}$  that use  $f(n)$  space.
3. Recursively test  $c_i \xrightarrow{b/2} c_{\text{mid}}$  and  $c_{\text{mid}} \xrightarrow{b/2} c_j$
4. If both are true, *accept*. If not, continue.
5. *Reject* if haven't yet accepted."

Test if  $N$  accepts  $w$  by testing  $c_{\text{start}} \xrightarrow{t} c_{\text{accept}}$  where  $t$  = number of configurations



# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

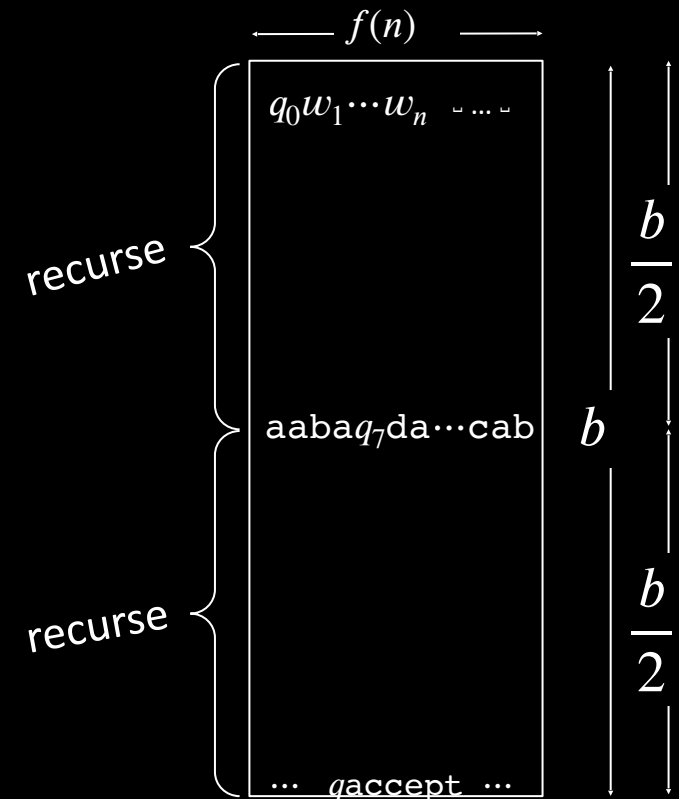
For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

Give recursive algorithm to test  $c_i \xrightarrow{b} c_j$ :

$M$  = "On input  $c_i, c_j, b$  [goal is to check  $c_i \xrightarrow{b} c_j$ ]

1. If  $b = 1$ , check directly by using  $N$ 's program and answer accordingly.
2. If  $b > 1$ , repeat for all configurations  $c_{\text{mid}}$  that use  $f(n)$  space.
3. Recursively test  $c_i \xrightarrow{b/2} c_{\text{mid}}$  and  $c_{\text{mid}} \xrightarrow{b/2} c_j$
4. If both are true, *accept*. If not, continue.
5. *Reject* if haven't yet accepted."

Test if  $N$  accepts  $w$  by testing  $c_{\text{start}} \xrightarrow{t} c_{\text{accept}}$  where  $t$  = number of configurations  
 $= |Q| \times f(n) \times d^{f(n)}$



# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

Give recursive algorithm to test  $c_i \xrightarrow{b} c_j$ :

$M$  = "On input  $c_i, c_j, b$  [goal is to check  $c_i \xrightarrow{b} c_j$ ]

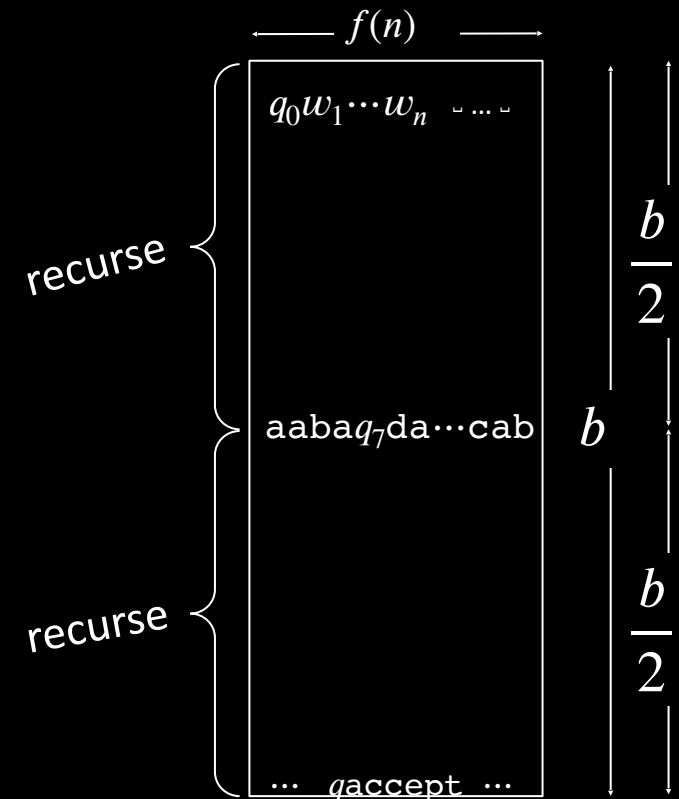
1. If  $b = 1$ , check directly by using  $N$ 's program and answer accordingly.
2. If  $b > 1$ , repeat for all configurations  $c_{\text{mid}}$  that use  $f(n)$  space.
3. Recursively test  $c_i \xrightarrow{b/2} c_{\text{mid}}$  and  $c_{\text{mid}} \xrightarrow{b/2} c_j$
4. If both are true, *accept*. If not, continue.
5. *Reject* if haven't yet accepted."

Test if  $N$  accepts  $w$  by testing  $c_{\text{start}} \xrightarrow{t} c_{\text{accept}}$  where  $t$  = number of configurations

$$= |Q| \times f(n) \times d^{f(n)}$$

Each recursion level stores 1 config =  $O(f(n))$  space.

Number of levels =  $\log t = O(f(n))$ . Total  $O(f^2(n))$  space.





# PSPACE = NPSPACE

**Savitch's Theorem:** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$

Proof: Convert NTM  $N$  to equivalent TM  $M$ , only squaring the space used.

For configurations  $c_i$  and  $c_j$  of  $N$ , write  $c_i \xrightarrow{b} c_j$  if can get from  $c_i$  to  $c_j$  in  $\leq b$  steps.

Give recursive algorithm to test  $c_i \xrightarrow{b} c_j$ :

$M$  = "On input  $c_i, c_j, b$  [goal is to check  $c_i \xrightarrow{b} c_j$ ]

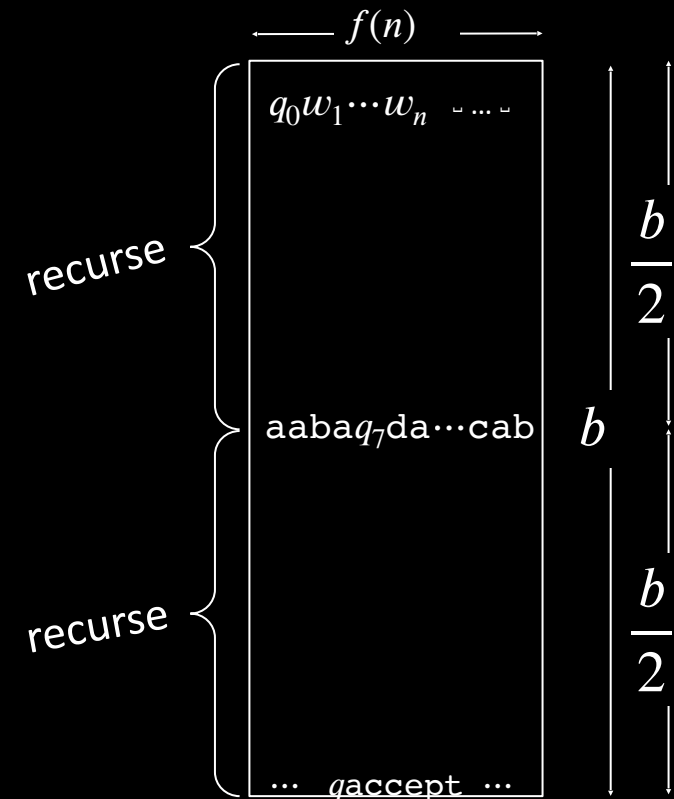
1. If  $b = 1$ , check directly by using  $N$ 's program and answer accordingly.
2. If  $b > 1$ , repeat for all configurations  $c_{\text{mid}}$  that use  $f(n)$  space.
3. Recursively test  $c_i \xrightarrow{b/2} c_{\text{mid}}$  and  $c_{\text{mid}} \xrightarrow{b/2} c_j$
4. If both are true, *accept*. If not, continue.
5. *Reject* if haven't yet accepted."

Test if  $N$  accepts  $w$  by testing  $c_{\text{start}} \xrightarrow{t} c_{\text{accept}}$  where  $t$  = number of configurations

$$= |Q| \times f(n) \times d^{f(n)}$$

Each recursion level stores 1 config =  $O(f(n))$  space.

Number of levels =  $\log t = O(f(n))$ . Total  $O(f^2(n))$  space.



# PSPACE-completeness

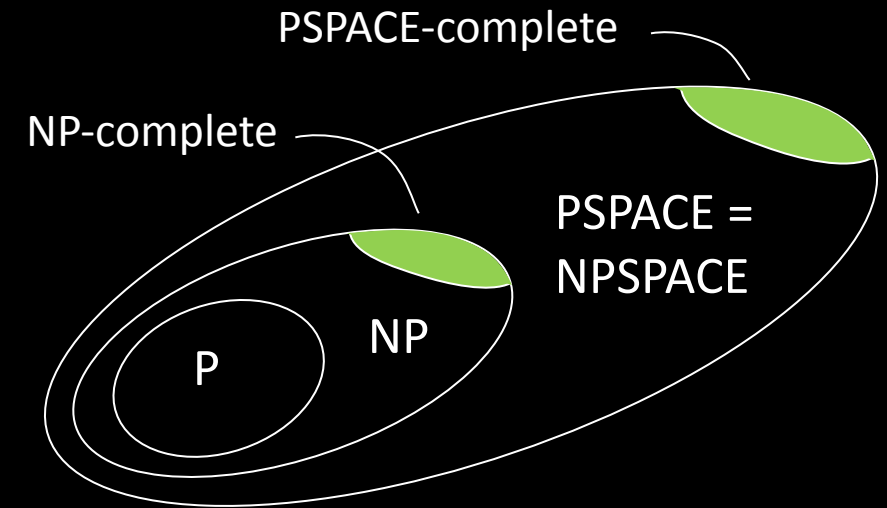
**Defn:**  $B$  is PSPACE-complete if

- 1)  $B \in \text{PSPACE}$
- 2) For all  $A \in \text{PSPACE}$ ,  $A \leq_p B$

# PSPACE-completeness

**Defn:**  $B$  is PSPACE-complete if

- 1)  $B \in \text{PSPACE}$
- 2) For all  $A \in \text{PSPACE}$ ,  $A \leq_p B$



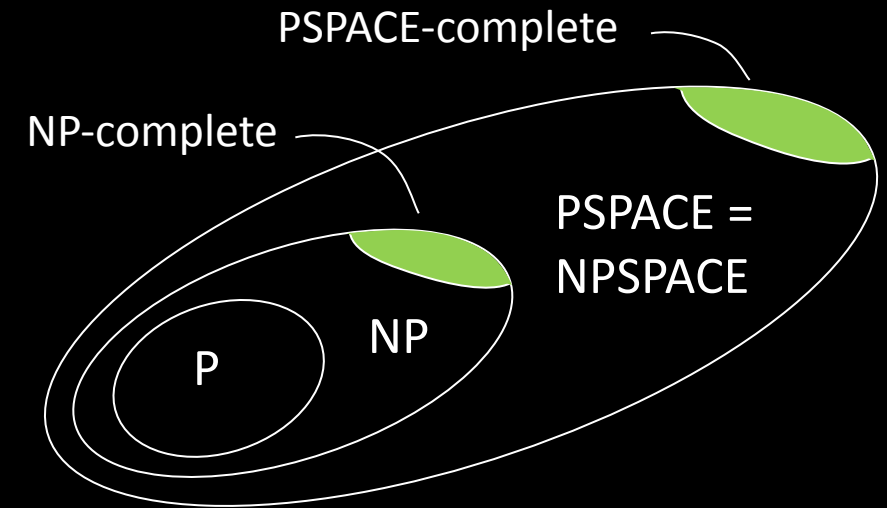
Think of complete problems as the “hardest” in their associated class.

# PSPACE-completeness

**Defn:**  $B$  is PSPACE-complete if

- 1)  $B \in \text{PSPACE}$
- 2) For all  $A \in \text{PSPACE}$ ,  $A \leq_p B$

If  $B$  is PSPACE-complete and  $B \in P$  then  $P = \text{PSPACE}$ .



Think of complete problems as the “hardest” in their associated class.

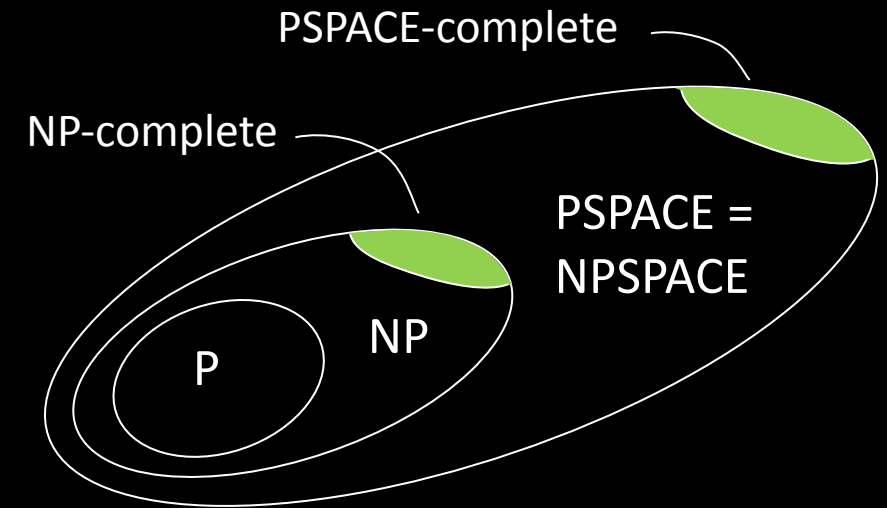
# PSPACE-completeness

**Defn:**  $B$  is PSPACE-complete if

- 1)  $B \in \text{PSPACE}$
- 2) For all  $A \in \text{PSPACE}$ ,  $A \leq_p B$

If  $B$  is PSPACE-complete and  $B \in P$  then  $P = \text{PSPACE}$ .

Why  $\leq_p$  and not  $\leq_{\text{PSPACE}}$  when defining PSPACE-complete?



Think of complete problems as the “hardest” in their associated class.

# PSPACE-completeness

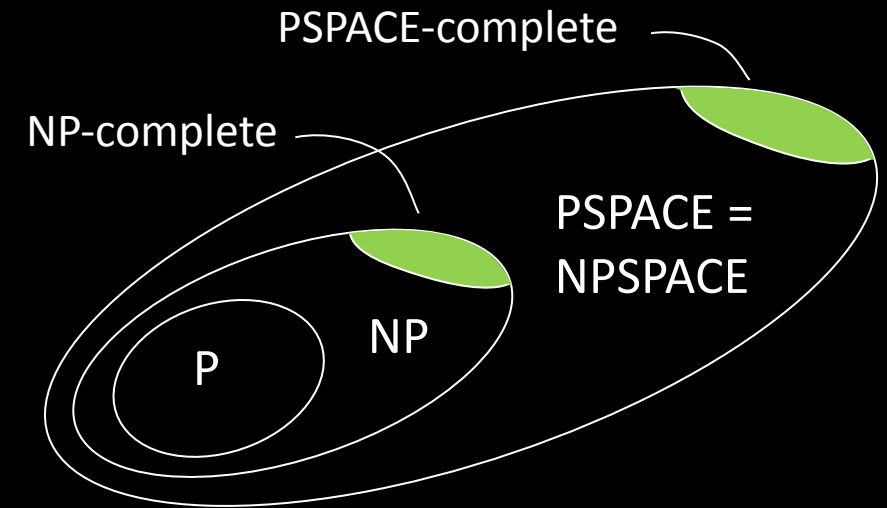
**Defn:**  $B$  is PSPACE-complete if

- 1)  $B \in \text{PSPACE}$
- 2) For all  $A \in \text{PSPACE}$ ,  $A \leq_p B$

If  $B$  is PSPACE-complete and  $B \in P$  then  $P = \text{PSPACE}$ .

Why  $\leq_p$  and not  $\leq_{\text{PSPACE}}$  when defining PSPACE-complete?

- Reductions should be “weaker” than the class. Otherwise all problems in the class would be reducible to each other, and then all problems in the class would be complete.



Think of complete problems as the “hardest” in their associated class.

# PSPACE-completeness

**Defn:**  $B$  is PSPACE-complete if

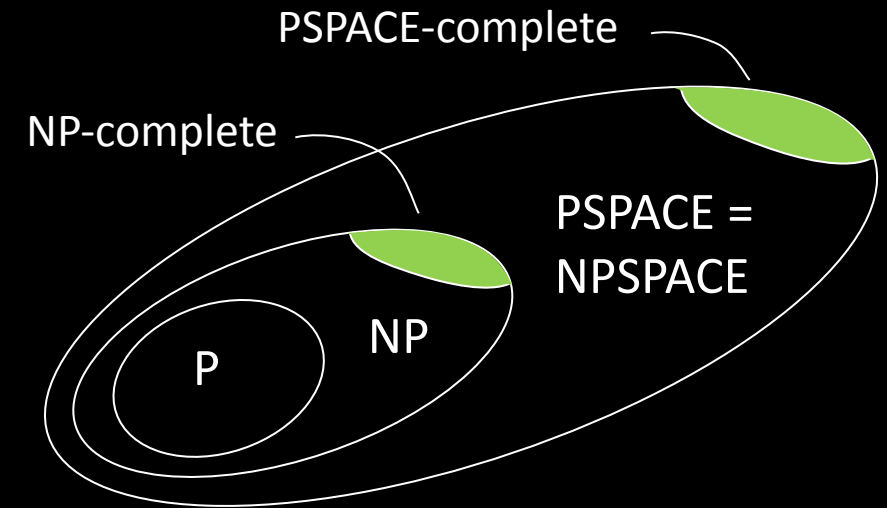
- 1)  $B \in \text{PSPACE}$
- 2) For all  $A \in \text{PSPACE}$ ,  $A \leq_p B$

If  $B$  is PSPACE-complete and  $B \in P$  then  $P = \text{PSPACE}$ .

Why  $\leq_p$  and not  $\leq_{\text{PSPACE}}$  when defining PSPACE-complete?

- Reductions should be “weaker” than the class. Otherwise all problems in the class would be reducible to each other, and then all problems in the class would be complete.

**Theorem:** *TQBF* is PSPACE-complete



Think of complete problems as the “hardest” in their associated class.

# PSPACE-completeness

**Defn:**  $B$  is PSPACE-complete if

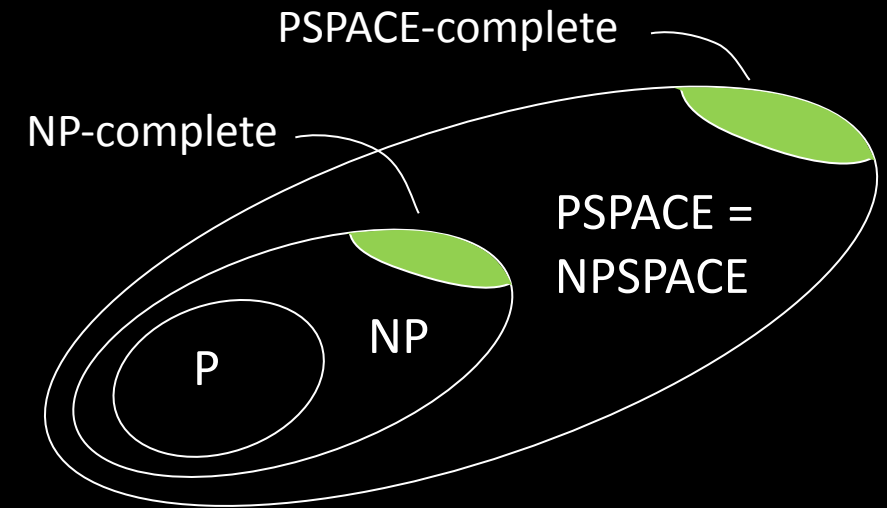
- 1)  $B \in \text{PSPACE}$
- 2) For all  $A \in \text{PSPACE}$ ,  $A \leq_p B$

If  $B$  is PSPACE-complete and  $B \in P$  then  $P = \text{PSPACE}$ .

Why  $\leq_p$  and not  $\leq_{\text{PSPACE}}$  when defining PSPACE-complete?

- Reductions should be “weaker” than the class. Otherwise all problems in the class would be reducible to each other, and then all problems in the class would be complete.

**Theorem:** *TQBF* is PSPACE-complete



Think of complete problems as the “hardest” in their associated class.



# PSPACE-completeness

**Defn:**  $B$  is PSPACE-complete if

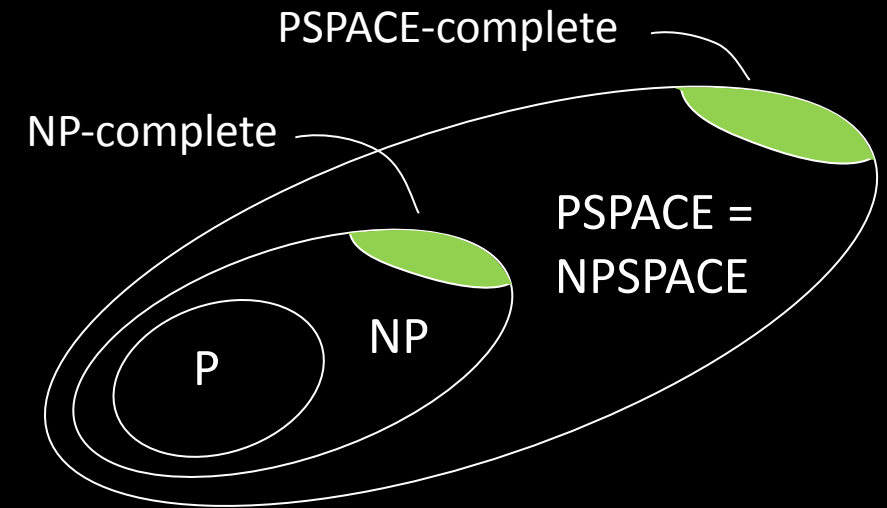
- 1)  $B \in \text{PSPACE}$
- 2) For all  $A \in \text{PSPACE}$ ,  $A \leq_p B$

If  $B$  is PSPACE-complete and  $B \in P$  then  $P = \text{PSPACE}$ .

## Check-in 18.1

Knowing that  $TQBF$  is PSPACE-complete, what can we conclude if  $TQBF \in \text{NP}$ ? Check all that apply.

- (a)  $P = \text{PSPACE}$
- (b)  $\text{NP} = \text{PSPACE}$
- (c)  $P = \text{NP}$
- (d)  $\text{NP} = \text{coNP}$



Think of complete problems as the “hardest” in their associated class.

# $TQBF$ is PSPACE-complete

Recall:  $TQBF = \{ \langle \phi \rangle \mid \phi \text{ is a QBF that is TRUE} \}$

**Examples:**  $\phi_1 = \forall x \exists y \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \in TQBF$  [TRUE]

$\phi_2 = \exists y \forall x \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \notin TQBF$  [FALSE]

# $TQBF$ is PSPACE-complete

Recall:  $TQBF = \{ \langle \phi \rangle \mid \phi \text{ is a QBF that is TRUE} \}$

**Examples:**  $\phi_1 = \forall x \exists y \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \in TQBF$  [TRUE]

$\phi_2 = \exists y \forall x \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \notin TQBF$  [FALSE]

**Theorem:**  $TQBF$  is PSPACE-complete

# $TQBF$ is PSPACE-complete

Recall:  $TQBF = \{ \langle \phi \rangle \mid \phi \text{ is a QBF that is TRUE} \}$

**Examples:**  $\phi_1 = \forall x \exists y \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \in TQBF$  [TRUE]

$\phi_2 = \exists y \forall x \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \notin TQBF$  [FALSE]

**Theorem:**  $TQBF$  is PSPACE-complete

Proof: 1)  $TQBF \in \text{PSPACE}$  ✓

# $TQBF$ is PSPACE-complete

Recall:  $TQBF = \{ \langle \phi \rangle \mid \phi \text{ is a QBF that is TRUE} \}$

**Examples:**  $\phi_1 = \forall x \exists y \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \in TQBF$  [TRUE]

$\phi_2 = \exists y \forall x \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \notin TQBF$  [FALSE]

**Theorem:**  $TQBF$  is PSPACE-complete

Proof: 1)  $TQBF \in \text{PSPACE}$  ✓

2) For all  $A \in \text{PSPACE}$ ,  $A \leq_p TQBF$

# $TQBF$ is PSPACE-complete

Recall:  $TQBF = \{ \langle \phi \rangle \mid \phi \text{ is a QBF that is TRUE} \}$

**Examples:**  $\phi_1 = \forall x \exists y \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \in TQBF$  [TRUE]

$\phi_2 = \exists y \forall x \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \notin TQBF$  [FALSE]

**Theorem:**  $TQBF$  is PSPACE-complete

Proof: 1)  $TQBF \in \text{PSPACE}$  ✓

2) For all  $A \in \text{PSPACE}$ ,  $A \leq_p TQBF$

Let  $A \in \text{PSPACE}$  be decided by TM  $M$  in space  $n^k$ .

Give a polynomial-time reduction  $f$  mapping  $A$  to  $TQBF$ .

# $TQBF$ is PSPACE-complete

Recall:  $TQBF = \{ \langle \phi \rangle \mid \phi \text{ is a QBF that is TRUE} \}$

**Examples:**  $\phi_1 = \forall x \exists y \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \in TQBF$  [TRUE]

$\phi_2 = \exists y \forall x \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \notin TQBF$  [FALSE]

**Theorem:**  $TQBF$  is PSPACE-complete

Proof: 1)  $TQBF \in \text{PSPACE}$  ✓

2) For all  $A \in \text{PSPACE}$ ,  $A \leq_p TQBF$

Let  $A \in \text{PSPACE}$  be decided by TM  $M$  in space  $n^k$ .

Give a polynomial-time reduction  $f$  mapping  $A$  to  $TQBF$ .

$f: \Sigma^* \rightarrow \text{QBFs}$

$f(w) = \langle \phi_{M,w} \rangle$

$w \in A$  iff  $\phi_{M,w}$  is TRUE

# $TQBF$ is PSPACE-complete

Recall:  $TQBF = \{ \langle \phi \rangle \mid \phi \text{ is a QBF that is TRUE} \}$

**Examples:**  $\phi_1 = \forall x \exists y \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \in TQBF$  [TRUE]

$\phi_2 = \exists y \forall x \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \notin TQBF$  [FALSE]

**Theorem:**  $TQBF$  is PSPACE-complete

Proof: 1)  $TQBF \in \text{PSPACE}$  ✓

2) For all  $A \in \text{PSPACE}$ ,  $A \leq_p TQBF$

Let  $A \in \text{PSPACE}$  be decided by TM  $M$  in space  $n^k$ .

Give a polynomial-time reduction  $f$  mapping  $A$  to  $TQBF$ .

$$f: \Sigma^* \rightarrow \text{QBFs}$$

$$f(w) = \langle \phi_{M,w} \rangle$$

$$w \in A \text{ iff } \phi_{M,w} \text{ is TRUE}$$

Plan: Design  $\phi_{M,w}$  to “say”  $M$  accepts  $w$ .  $\phi_{M,w}$  simulates  $M$  on  $w$ .



# $TQBF$ is PSPACE-complete

Recall:  $TQBF = \{ \langle \phi \rangle \mid \phi \text{ is a QBF that is TRUE} \}$

**Examples:**  $\phi_1 = \forall x \exists y \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \in TQBF$  [TRUE]

$\phi_2 = \exists y \forall x \left[ (x \vee y) \wedge (\bar{x} \vee \bar{y}) \right] \notin TQBF$  [FALSE]

**Theorem:**  $TQBF$  is PSPACE-complete

Proof: 1)  $TQBF \in \text{PSPACE}$  ✓

2) For all  $A \in \text{PSPACE}$ ,  $A \leq_p TQBF$

Let  $A \in \text{PSPACE}$  be decided by TM  $M$  in space  $n^k$ .

Give a polynomial-time reduction  $f$  mapping  $A$  to  $TQBF$ .

$$f: \Sigma^* \rightarrow \text{QBFs}$$

$$f(w) = \langle \phi_{M,w} \rangle$$

$$w \in A \text{ iff } \phi_{M,w} \text{ is TRUE}$$

Plan: Design  $\phi_{M,w}$  to “say”  $M$  accepts  $w$ .  $\phi_{M,w}$  simulates  $M$  on  $w$ .

# Constructing $\phi_{M,w}$ : 1<sup>st</sup> try

## Tableau for $M$ on $w$

			<div> <div></div> <div>...</div> <div></div> </div>
a			

Recall: A tableau for  $M$  on  $w$  represents a computation history for  $M$  on  $w$  when  $M$  accepts  $w$ . Rows of that tableau are configurations.

# Constructing $\phi_{M,w}$ : 1<sup>st</sup> try

## Tableau for $M$ on $w$

[illegible]

Recall: A tableau for  $M$  on  $w$  represents  
a computation history for  $M$  on  $w$   
when  $M$  accepts  $w$ .

Rows of that tableau are configurations.

$M$  runs in space  $n^k$ , its tableau has:

# Constructing $\phi_{M,w}$ : 1<sup>st</sup> try

## Tableau for $M$ on $w$

[illegible]

Recall: A tableau for  $M$  on  $w$  represents  
a computation history for  $M$  on  $w$   
when  $M$  accepts  $w$ .

Rows of that tableau are configurations.

$M$  runs in space  $n^k$ , its tableau has:

- $n^k$  columns (max size of a configuration)

# Constructing $\phi_{M,w}$ : 1<sup>st</sup> try

## Tableau for $M$ on $w$

$$\leftarrow n^k \rightarrow$$

			<div> <div></div> <div>...</div> <div></div> </div>
a			

Recall: A tableau for  $M$  on  $w$  represents  
a computation history for  $M$  on  $w$   
when  $M$  accepts  $w$ .

Rows of that tableau are configurations.

$M$  runs in space  $n^k$ , its tableau has:

- $n^k$  columns (max size of a configuration)

# Constructing $\phi_{M,w}$ : 1<sup>st</sup> try

## Tableau for $M$ on $w$

[illegible]

Recall: A tableau for  $M$  on  $w$  represents  
a computation history for  $M$  on  $w$   
when  $M$  accepts  $w$ .

Rows of that tableau are configurations.

$M$  runs in space  $n^k$ , its tableau has:

- $n^k$  columns (max size of a configuration)
- $d^{(n^k)}$  rows (max number of steps)

# Constructing $\phi_{M,w}$ : 1<sup>st</sup> try

## Tableau for $M$ on $w$

The diagram illustrates a tensor network for a matrix product state (MPS). It consists of a vertical chain of tensors labeled  $a$ . Each tensor  $a$  has two vertical indices,  $i$  and  $j$ , and two horizontal indices. The horizontal indices are grouped into a block of size  $n^k$ , indicated by a horizontal line with a brace labeled  $n^k$ . The vertical indices are grouped into a block of size  $d^{(n^k)}$ , indicated by a vertical line with a brace labeled  $d^{(n^k)}$ . The tensors are connected horizontally and vertically, forming a grid-like structure.

Recall: A tableau for  $M$  on  $w$  represents a computation history for  $M$  on  $w$  when  $M$  accepts  $w$ .

Rows of that tableau are configurations.

$M$  runs in space  $n^k$ , its tableau has:

- $n^k$  columns (max size of a configuration)
- $d^{(n^k)}$  rows (max number of steps)

# Constructing $\phi_{M,w}$ : 1<sup>st</sup> try

# Tableau for $M$ on $w$

Diagram illustrating a tensor network structure. A vertical line on the left is labeled  $d^{(n^k)}$ . A horizontal line at the top is labeled  $n^k$ . The main part is a grid of boxes. The first row has three boxes, with the last one containing 'a'. The first column has three boxes, with the last one containing 'a'. The rest of the grid is empty.

Recall: A tableau for  $M$  on  $w$  represents a computation history for  $M$  on  $w$  when  $M$  accepts  $w$ .

Rows of that tableau are configurations.

$M$  runs in space  $n^k$ , its tableau has:

- $n^k$  columns (max size of a configuration)
- $d^{(n^k)}$  rows (max number of steps)

Constructing  $\phi_{M,w}$ . Try Cook-Levin method.



# Constructing $\phi_{M,w}$ : 1<sup>st</sup> try

## Tableau for $M$ on $w$

Diagram illustrating a tensor network structure for a matrix multiplication. The diagram shows a large vertical rectangle divided into three columns. The top row is labeled 'a' and the bottom row is labeled 'b'. The top row is connected to the bottom row by a horizontal line labeled  $n^k$ . The left side of the rectangle is labeled  $d^{(n^k)}$ .

Recall: A tableau for  $M$  on  $w$  represents a computation history for  $M$  on  $w$  when  $M$  accepts  $w$ .

Rows of that tableau are configurations.

$M$  runs in space  $n^k$ , its tableau has:

- $n^k$  columns (max size of a configuration)
- $d^{(n^k)}$  rows (max number of steps)

Constructing  $\phi_{M,w}$ . Try Cook-Levin method.

Then  $\phi_{M,w}$  will be as big as tableau.

# Constructing $\phi_{M,w}$ : 1<sup>st</sup> try

## Tableau for $M$ on $w$

Diagram illustrating a tensor network structure for a matrix multiplication. The diagram shows a large vertical rectangle divided into three columns. The top row is labeled 'a' and the bottom row is labeled 'b'. The top row is connected to the bottom row by a horizontal line labeled  $n^k$ . The left side of the rectangle is labeled  $d^{(n^k)}$ .

Recall: A tableau for  $M$  on  $w$  represents a computation history for  $M$  on  $w$  when  $M$  accepts  $w$ .

Rows of that tableau are configurations.

$M$  runs in space  $n^k$ , its tableau has:

- $n^k$  columns (max size of a configuration)
- $d^{(n^k)}$  rows (max number of steps)

Constructing  $\phi_{M,w}$ . Try Cook-Levin method.

Then  $\phi_{M,w}$  will be as big as tableau.

But that is exponential:  $n^k \times d^{(n^k)}$ .

# Constructing $\phi_{M,w}$ : 1<sup>st</sup> try

## Tableau for $M$ on $w$

Diagram illustrating a 3D tensor  $d^{(n^k)}$  with dimensions  $n^k$  along all three axes. The tensor is represented as a 3D box with three axes labeled  $n^k$ . The top face is a grid with 3 columns and 3 rows. The front face is a grid with 3 columns and 3 rows. The right face is a grid with 3 columns and 3 rows. The tensor is labeled  $d^{(n^k)}$  on the left.

Recall: A tableau for  $M$  on  $w$  represents a computation history for  $M$  on  $w$  when  $M$  accepts  $w$ .

Rows of that tableau are configurations.

$M$  runs in space  $n^k$ , its tableau has:

- $n^k$  columns (max size of a configuration)
- $d^{(n^k)}$  rows (max number of steps)

Constructing  $\phi_{M,w}$ . Try Cook-Levin method.

Then  $\phi_{M,w}$  will be as big as tableau.

But that is exponential:  $n^k \times d^{(n^k)}$ .

Too big! ☹️

# Constructing $\phi_{M,w}$ : 1<sup>st</sup> try

## Tableau for $M$ on $w$

Diagram illustrating a tensor network structure for a matrix multiplication. The structure consists of a large vertical rectangle divided into three columns. The top row is labeled 'a' and the bottom row is labeled 'b'. The top row is connected to the bottom row by a horizontal line labeled  $n^k$ . The vertical lines are labeled  $d^{(n^k)}$ .

Recall: A tableau for  $M$  on  $w$  represents a computation history for  $M$  on  $w$  when  $M$  accepts  $w$ .

Rows of that tableau are configurations.

$M$  runs in space  $n^k$ , its tableau has:

- $n^k$  columns (max size of a configuration)
- $d^{(n^k)}$  rows (max number of steps)

Constructing  $\phi_{M,w}$ . Try Cook-Levin method.

Then  $\phi_{M,w}$  will be as big as tableau.

But that is exponential:  $n^k \times d^{(n^k)}$ .

Too big! ☹️

# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

## Tableau for $M$ on $w$

The diagram illustrates a 3D tensor  $d^{(n^k)}$  with three dimensions, each of size  $n^k$ . The tensor is represented as a cube. The top face is labeled with indices  $1 \dots n^k$ . The front face is labeled with indices  $1 \dots n^k$ . The right face is labeled with indices  $1 \dots n^k$ . The tensor is labeled  $d^{(n^k)}$  on the left side.

$$d^{(n^k)}$$

hide →

$$v^k)$$
[illegible]

# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

## Tableau for $M$ on $w$

The diagram illustrates the construction of a  $d$ -ary tree from a binary tree. On the left, a binary tree is shown with root  $a$ . On the right, a  $d$ -ary tree is shown with root  $a$ . The  $d$ -ary tree has  $d$  children for each node, labeled  $b, c, d, \dots, z$ . The  $d$ -ary tree is labeled  $d^{(n^k)}$ .

$$d^{(n^k)}$$

# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

## Tableau for $M$ on $w$

For configs  $c_i$  and  $c_j$  construct  $\phi_{c_i, c_j, b}$  which “says”  $c_i \xrightarrow{b} c_j$  recursively.

$d^{(n^k)}$				⌊ ... ⌊
	a			



# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

## Tableau for $M$ on $w$

For configs  $c_i$  and  $c_j$  construct  $\phi_{c_i, c_j, b}$  which “says”  $c_i \xrightarrow{b} c_j$  recursively.

$$\phi_{c_i, c_j, b} = \exists c_{\text{mid}} \left[ \phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2} \right]$$

[illegible]
$$d^{(n^k)}$$

# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

## Tableau for $M$ on $w$

$$n^k$$

For configs  $c_i$  and  $c_j$  construct  $\phi_{c_i, c_j, b}$  which “says”  $c_i \xrightarrow{b} c_j$  recursively.

$$\phi_{c_i, c_j, b} = \underbrace{\exists c_{\text{mid}}}_{\text{mid}} \left[ \phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2} \right]$$

$\exists x_1, x_2, \dots, c_l$   
as in Cook-Levin

$$d^{(n^k)}$$

# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

Tableau for  $M$  on  $w$

$d^{(n^k)}$

$n^k$			
			... $n^k$
a			

For configs  $c_i$  and  $c_j$  construct  $\phi_{c_i, c_j, b}$  which “says”  $c_i \xrightarrow{b} c_j$  recursively.

$$\phi_{c_i, c_j, b} = \underbrace{\exists c_{\text{mid}}}_{\substack{\exists x_1, x_2, \dots, x_l \\ \text{as in Cook-Levin}}} \left[ \phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2} \right]$$

$$\exists c_{\text{mid}} \left[ \phi_{\phantom{c_i}, \phantom{c_j}, b/4} \wedge \phi_{\phantom{c_i}, \phantom{c_j}, b/4} \right]$$

# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

Tableau for  $M$  on  $w$

$d^{(n^k)}$

$n^k$			
			⋮ ⋮ ⋮
a			

For configs  $c_i$  and  $c_j$  construct  $\phi_{c_i, c_j, b}$  which “says”  $c_i \xrightarrow{b} c_j$  recursively.

$$\phi_{c_i, c_j, b} = \underbrace{\exists c_{\text{mid}}}_{\substack{\exists x_1, x_2, \dots, x_l \\ \text{as in Cook-Levin}}} \left[ \phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2} \right]$$

$\swarrow \quad \searrow$   
 $\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right]$

$\swarrow \quad \searrow$   
 $\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right]$

# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

Tableau for  $M$  on  $w$

$d^{(n^k)}$

$n^k$			
			⋮ ⋮ ⋮
a			

For configs  $c_i$  and  $c_j$  construct  $\phi_{c_i, c_j, b}$  which “says”  $c_i \xrightarrow{b} c_j$  recursively.

$$\phi_{c_i, c_j, b} = \underbrace{\exists c_{\text{mid}}}_{\substack{\exists x_1, x_2, \dots, x_l \\ \text{as in Cook-Levin}}} \left[ \phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2} \right]$$

$\swarrow \quad \searrow$   
 $\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right]$

$\swarrow \quad \searrow$   
 $\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right]$   
 $\swarrow \quad \searrow$   
 $\exists c_{\text{mid}} [\phi_{\cdot, \cdot, b/8} \cdots]$

# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

Tableau for  $M$  on  $w$

$d^{(n^k)}$

$n^k$			
			⋮
a			

For configs  $c_i$  and  $c_j$  construct  $\phi_{c_i, c_j, b}$  which “says”  $c_i \xrightarrow{b} c_j$  recursively.

$$\phi_{c_i, c_j, b} = \underbrace{\exists c_{\text{mid}}}_{\substack{\exists x_1, x_2, \dots, x_l \\ \text{as in Cook-Levin}}} \left[ \phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2} \right]$$

$\swarrow \quad \searrow$   
 $\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right]$   
 $\vdots$

$\swarrow \quad \searrow$   
 $\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right]$   
 $\vdots$   
 $\swarrow \quad \searrow$   
 $\exists c_{\text{mid}} [\phi_{\cdot, \cdot, b/8} \cdots]$

# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

Tableau for  $M$  on  $w$

$d^{(n^k)}$

$n^k$			
			...
a			

For configs  $c_i$  and  $c_j$  construct  $\phi_{c_i, c_j, b}$  which “says”  $c_i \xrightarrow{b} c_j$  recursively.

$$\phi_{c_i, c_j, b} = \underbrace{\exists c_{\text{mid}}}_{\substack{\exists x_1, x_2, \dots, c_l \\ \text{as in Cook-Levin}}} \left[ \phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2} \right]$$

$\downarrow \qquad \downarrow$   
 $\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right]$   
 $\vdots$   
 $\phi_{\cdot, \cdot, 1}$  defined as in Cook-Levin

$\downarrow \qquad \downarrow$   
 $\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right]$   
 $\vdots$   
 $\exists c_{\text{mid}} [\phi_{\cdot, \cdot, b/8} \dots]$

# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

## Tableau for $M$ on $w$

For configs  $c_i$  and  $c_j$  construct  $\phi_{c_i, c_j, b}$  which “says”  $c_i \xrightarrow{b} c_j$  recursively.

$$\phi_{c_i, c_j, b} = \underbrace{\exists c_{\text{mid}}}_{\substack{\exists x_1, x_2, \dots, c_l \\ \text{as in Cook-Levin}}} \left[ \phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/8} \wedge \phi_{\cdot, \cdot, b/8} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/16} \wedge \phi_{\cdot, \cdot, b/16} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/32} \wedge \phi_{\cdot, \cdot, b/32} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/64} \wedge \phi_{\cdot, \cdot, b/64} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/128} \wedge \phi_{\cdot, \cdot, b/128} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/256} \wedge \phi_{\cdot, \cdot, b/256} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/512} \wedge \phi_{\cdot, \cdot, b/512} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/1024} \wedge \phi_{\cdot, \cdot, b/1024} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/2048} \wedge \phi_{\cdot, \cdot, b/2048} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4096} \wedge \phi_{\cdot, \cdot, b/4096} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/8192} \wedge \phi_{\cdot, \cdot, b/8192} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/16384} \wedge \phi_{\cdot, \cdot, b/16384} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/32768} \wedge \phi_{\cdot, \cdot, b/32768} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/65536} \wedge \phi_{\cdot, \cdot, b/65536} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/131072} \wedge \phi_{\cdot, \cdot, b/131072} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/262144} \wedge \phi_{\cdot, \cdot, b/262144} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/524288} \wedge \phi_{\cdot, \cdot, b/524288} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/1048576} \wedge \phi_{\cdot, \cdot, b/1048576} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/2097152} \wedge \phi_{\cdot, \cdot, b/2097152} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4194304} \wedge \phi_{\cdot, \cdot, b/4194304} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/8388608} \wedge \phi_{\cdot, \cdot, b/8388608} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/16777216} \wedge \phi_{\cdot, \cdot, b/16777216} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/33554432} \wedge \phi_{\cdot, \cdot, b/33554432} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/67108864} \wedge \phi_{\cdot, \cdot, b/67108864} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/134217728} \wedge \phi_{\cdot, \cdot, b/134217728} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/268435456} \wedge \phi_{\cdot, \cdot, b/268435456} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/536870912} \wedge \phi_{\cdot, \cdot, b/536870912} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/1073741824} \wedge \phi_{\cdot, \cdot, b/1073741824} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/2147483648} \wedge \phi_{\cdot, \cdot, b/2147483648} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4294967296} \wedge \phi_{\cdot, \cdot, b/4294967296} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/8589934592} \wedge \phi_{\cdot, \cdot, b/8589934592} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/17179869184} \wedge \phi_{\cdot, \cdot, b/17179869184} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/34359738368} \wedge \phi_{\cdot, \cdot, b/34359738368} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/68719476736} \wedge \phi_{\cdot, \cdot, b/68719476736} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/137438953472} \wedge \phi_{\cdot, \cdot, b/137438953472} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/274877906944} \wedge \phi_{\cdot, \cdot, b/274877906944} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/549755813888} \wedge \phi_{\cdot, \cdot, b/549755813888} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/1099511627776} \wedge \phi_{\cdot, \cdot, b/1099511627776} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/2199023255552} \wedge \phi_{\cdot, \cdot, b/2199023255552} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4398046511104} \wedge \phi_{\cdot, \cdot, b/4398046511104} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/8796093022208} \wedge \phi_{\cdot, \cdot, b/8796093022208} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/17592186044416} \wedge \phi_{\cdot, \cdot, b/17592186044416} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/35184372088832} \wedge \phi_{\cdot, \cdot, b/35184372088832} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/70368744177664} \wedge \phi_{\cdot, \cdot, b/70368744177664} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/140737488355328} \wedge \phi_{\cdot, \cdot, b/140737488355328} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/281474976710656} \wedge \phi_{\cdot, \cdot, b/281474976710656} \right]$$

$$\vdots \qquad \vdots$$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot$$

 $\phi_{\epsilon, \delta, 1}$  defined as in Cook-Levin

$$\phi_{M,w} = \phi_{c_{\text{start}}, c_{\text{accept}}, t}$$

[illegible]



# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

## Tableau for $M$ on $w$

For configs  $c_i$  and  $c_j$  construct  $\phi_{c_i, c_j, b}$  which “says”  $c_i \xrightarrow{b} c_j$  recursively.

$$\phi_{c_i, c_j, b} = \underbrace{\exists c_{\text{mid}}}_{\substack{\exists x_1, x_2, \dots, c_l \\ \text{as in Cook-Levin}}} \left[ \phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2} \right]$$

$\vdots$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right]$$

$\vdots$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/8} \cdots \right]$$

 $\phi_{\epsilon, \delta, 1}$  defined as in Cook-Levin

$$\phi_{M,w} = \phi_{c_{\text{start}}, c_{\text{accept}}, t}$$

## Size analysis:

Each recursive level doubles number of QBFs.

$d^{(n^k)}$				<div> <div></div> <div>...</div> <div></div> </div>
	a			

# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

## Tableau for $M$ on $w$

For configs  $c_i$  and  $c_j$  construct  $\phi_{c_i, c_j, b}$  which “says”  $c_i \xrightarrow{b} c_j$  recursively.

$$\phi_{c_i, c_j, b} = \underbrace{\exists c_{\text{mid}}}_{\substack{\exists x_1, x_2, \dots, c_l \\ \text{as in Cook-Levin}}} \left[ \phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2} \right]$$

$\vdots$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right]$$

$\vdots$

$$\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/8} \cdots \right]$$

 $\phi_{\epsilon, \delta, 1}$  defined as in Cook-Levin

$$\phi_{M,w} = \phi_{c_{\text{start}}, c_{\text{accept}}, t}$$

## Size analysis:

Each recursive level doubles number of QBFs.

Number of levels is  $\log d^{(n^k)} = O(n^k)$ .

$d^{(n^k)}$				<div> <div></div> <div>...</div> <div></div> </div>
	a			

# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

## Tableau for $M$ on $w$

For configs  $c_i$  and  $c_j$  construct  $\phi_{c_i, c_j, b}$  which “says”  $c_i \xrightarrow{b} c_j$  recursively.

$$\begin{array}{c}
\phi_{c_i, c_j, b} = \underbrace{\exists c_{\text{mid}}}_{\substack{\exists x_1, x_2, \dots, c_l \\ \text{as in Cook-Levin}}} \left[ \phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2} \right] \\
\begin{array}{ccc}
\swarrow & & \searrow \\
\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right] & & \exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right] \\
\vdots & & \vdots \\
\phi_{\cdot, \cdot, 1} \text{ defined as in Cook-Levin} & & \exists c_{\text{mid}} [\phi_{\cdot, \cdot, b/8} \dots]
\end{array}
\end{array}$$

$$\phi_{M,w} = \phi_{c_{\text{start}}, c_{\text{accept}}, t}$$

## Size analysis:

Each recursive level doubles number of QBFs.  
Number of levels is  $\log d^{(n^k)} = O(n^k)$ .

→ Size is exponential. ☹️

# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

## Tableau for $M$ on $w$

For configs  $c_i$  and  $c_j$  construct  $\phi_{c_i, c_j, b}$  which “says”  $c_i \xrightarrow{b} c_j$  recursively.

$$\begin{array}{c}
\phi_{c_i, c_j, b} = \underbrace{\exists c_{\text{mid}}}_{\substack{\exists x_1, x_2, \dots, c_l \\ \text{as in Cook-Levin}}} \left[ \phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2} \right] \\
\begin{array}{ccc}
\swarrow & & \searrow \\
\exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right] & & \exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right] \\
\vdots & & \vdots \\
\phi_{\cdot, \cdot, 1} \text{ defined as in Cook-Levin} & & \exists c_{\text{mid}} [\phi_{\cdot, \cdot, b/8} \dots]
\end{array}
\end{array}$$

$$\phi_{M,w} = \phi_{c_{\text{start}}, c_{\text{accept}}, t}$$

$$t = d^{(n^k)}$$

## Size analysis:

Each recursive level doubles number of QBFs.  
Number of levels is  $\log d^{(n^k)} = O(n^k)$ .

→ Size is exponential. ☹️

## Check-in 18.2

$d^{(n^k)}$				⏪ ⋮ ⏩
	a			

# Constructing $\phi_{M,w}$ : 2<sup>nd</sup> try

· configs  $c_i$  and  $c_j$  construct  $\phi_{c_i, c_j, b}$  which “says”  $c_i \xrightarrow{b} c_j$  recursively.

$$\phi_{c_i, c_j, b} = \underbrace{\exists c_{\text{mid}}}_{\substack{\exists x_1, x_2, \dots, x_l \\ \text{as in Cook-Levin}}} \left[ \phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2} \right]$$

$$\begin{array}{c} \swarrow \quad \searrow \qquad \qquad \swarrow \quad \searrow \\ \exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right] \quad \exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/4} \wedge \phi_{\cdot, \cdot, b/4} \right] \\ \vdots \qquad \qquad \qquad \vdots \\ \exists c_{\text{mid}} \left[ \phi_{\cdot, \cdot, b/8} \dots \right] \end{array}$$

$\phi_{\cdot, \cdot, 1}$  defined as in Cook-Levin

## Check-in 18.2

Why shouldn't we be surprised that this construction fails?

- (a) We can't define a QBF by using recursion.
- (b) It doesn't use  $\forall$  anywhere.
- (c) We know that  $TQBF \notin \text{P}$ .

$$\phi_{M,w} = \phi_{c_{\text{start}}, c_{\text{accept}}, t}$$

$$t = d^{(n^k)}$$

## Size analysis:

Each recursive level doubles number of QBFs.  
Number of levels is  $\log d^{(n^k)} = O(n^k)$ .

→ Size is exponential. ☹

Check-in 18.2

## Constructing $\phi_{M,w}$ : 3<sup>rd</sup> try

$$\phi_{c_i, c_j, b} = \exists c_{\text{mid}} \left[ \phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2} \right]$$

## Constructing $\phi_{M,w}$ : 3<sup>rd</sup> try

$$\phi_{c_i, c_j, b} = \exists c_{\text{mid}} \left[ \underbrace{\phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2}}_{\text{}} \right]$$
$$\forall (c_g, c_h) \in \left\{ (c_i, c_{\text{mid}}), (c_{\text{mid}}, c_j) \right\} \left[ \phi_{c_g, c_h, b/2} \right]$$

# Constructing $\phi_{M,w}$ : 3<sup>rd</sup> try

$$\phi_{c_i, c_j, b} = \exists c_{\text{mid}} \left[ \underbrace{\phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2}}_{\text{...}} \right]$$
$$\forall (c_g, c_h) \in \left\{ (c_i, c_{\text{mid}}), (c_{\text{mid}}, c_j) \right\} \left[ \phi_{c_g, c_h, b/2} \right]$$
$$\vdots$$



# Constructing $\phi_{M,w}$ : 3<sup>rd</sup> try

$$\phi_{c_i, c_j, b} = \exists c_{\text{mid}} \left[ \underbrace{\phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2}}_{\text{...}} \right]$$

$$\forall (c_g, c_h) \in \left\{ (c_i, c_{\text{mid}}), (c_{\text{mid}}, c_j) \right\} \left[ \phi_{c_g, c_h, b/2} \right]$$

$$\vdots$$

$\phi_{, , 1}$  defined as in Cook-Levin

# Constructing $\phi_{M,w}$ : 3<sup>rd</sup> try

$$\phi_{c_i, c_j, b} = \exists c_{\text{mid}} \left[ \underbrace{\phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2}} \right]$$

$$\forall (c_g, c_h) \in \left\{ (c_i, c_{\text{mid}}), (c_{\text{mid}}, c_j) \right\} \left[ \phi_{c_g, c_h, b/2} \right]$$

$$\vdots$$

$\forall (x \in S) [\psi]$   
is equivalent to  
 $\forall x [(x \in S) \rightarrow \psi]$

$\phi_{, , 1}$  defined as in Cook-Levin

# Constructing $\phi_{M,w}$ : 3<sup>rd</sup> try

$$\phi_{c_i, c_j, b} = \exists c_{\text{mid}} \left[ \underbrace{\phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2}} \right]$$

$$\forall (c_g, c_h) \in \left\{ (c_i, c_{\text{mid}}), (c_{\text{mid}}, c_j) \right\} \left[ \phi_{c_g, c_h, b/2} \right]$$

$$\vdots$$

$\forall (x \in S) [\psi]$   
is equivalent to  
 $\forall x [(x \in S) \rightarrow \psi]$

$$\phi_{M,w} = \phi_{c_{\text{start}}, c_{\text{accept}}, t}$$

$$t = d(n^k)$$

$\phi_{, , 1}$  defined as in Cook-Levin

# Constructing $\phi_{M,w}$ : 3<sup>rd</sup> try

$$\phi_{c_i, c_j, b} = \exists c_{\text{mid}} \left[ \underbrace{\phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2}} \right]$$

$$\forall (c_g, c_h) \in \left\{ (c_i, c_{\text{mid}}), (c_{\text{mid}}, c_j) \right\} \left[ \phi_{c_g, c_h, b/2} \right]$$

$$\vdots$$

$\forall (x \in S) [\psi]$   
is equivalent to  
 $\forall x [(x \in S) \rightarrow \psi]$

$$\phi_{M,w} = \phi_{c_{\text{start}}, c_{\text{accept}}, t}$$

$$t = d^{(n^k)}$$

$\phi_{, , 1}$  defined as in Cook-Levin

## Size analysis:

Each recursive level adds  $O(n^k)$  to the QBF.

Number of levels is  $\log d^{(n^k)} = O(n^k)$ .

→ Size is  $O(n^k \times n^k) = O(n^{2k})$  ☺

# Constructing $\phi_{M,w}$ : 3<sup>rd</sup> try

$$\phi_{c_i, c_j, b} = \exists c_{\text{mid}} \left[ \underbrace{\phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2}} \right]$$

$$\forall (c_g, c_h) \in \left\{ (c_i, c_{\text{mid}}), (c_{\text{mid}}, c_j) \right\} \left[ \phi_{c_g, c_h, b/2} \right]$$

$$\vdots$$

$\forall (x \in S) [\psi]$   
is equivalent to  
 $\forall x [(x \in S) \rightarrow \psi]$

$$\phi_{M,w} = \phi_{c_{\text{start}}, c_{\text{accept}}, t}$$

$$t = d^{(n^k)}$$

$\phi_{, , 1}$  defined as in Cook-Levin

## Size analysis:

Each recursive level adds  $O(n^k)$  to the QBF.

Number of levels is  $\log d^{(n^k)} = O(n^k)$ .

→ Size is  $O(n^k \times n^k) = O(n^{2k})$  ☺

# Constructing $\phi_{M,w}$ : 3<sup>rd</sup> try

$$\phi_{c_i, c_j, b} = \exists c_{\text{mid}} \left[ \underbrace{\phi_{c_i, c_{\text{mid}}, b/2} \wedge \phi_{c_{\text{mid}}, c_j, b/2}} \right]$$

$$\forall (c_g, c_h) \in \left\{ (c_i, c_{\text{mid}}), (c_{\text{mid}}, c_j) \right\} \left[ \phi_{c_g, c_h, b/2} \right]$$

$$\vdots$$

$\forall (x \in S) [\psi]$   
is equivalent to  
 $\forall x [(x \in S) \rightarrow \psi]$

$$\phi_{M,w} = \phi_{c_{\text{start}}, c_{\text{accept}}, t}$$

$$t = d^{(n^k)}$$

$\phi_{, , 1}$  defined as in Cook-Levin

## Size analysis:

Each recursive level adds  $O(n^k)$  to the QBF.  
Number of levels is  $\log d^{(n^k)} = O(n^k)$ .

→ Size is  $O(n^k \times n^k) = O(n^{2k})$  ☺

## Check-in 18.3

Would this construction still work if  $M$  were nondeterministic?

- (a) Yes.
- (b) No.

Check-in 18.3

# Quick review of today

1. Space complexity
2.  $\text{SPACE}(f(n))$ ,  $\text{NSPACE}(f(n))$
3. PSPACE, NPSPACE
4. Relationship with TIME classes
5.  $TQBF \in \text{PSPACE}$
6.  $LADDER_{\text{DFA}} \in \text{NSPACE}(n)$
7.  $LADDER_{\text{DFA}} \in \text{SPACE}(n^2)$
8. Savitch's Theorem:  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$
9.  $TQBF$  is PSPACE-complete

# Quick review of today

1. Space complexity
2.  $\text{SPACE}(f(n))$ ,  $\text{NSPACE}(f(n))$
3. PSPACE, NPSPACE
4. Relationship with TIME classes
5.  $TQBF \in \text{PSPACE}$
6.  $LADDER_{\text{DFA}} \in \text{NSPACE}(n)$
7.  $LADDER_{\text{DFA}} \in \text{SPACE}(n^2)$
8. Savitch's Theorem:  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$
9.  $TQBF$  is PSPACE-complete