Lecture 11: The Ladder Game

Recapping co-classes and space complexity

Which of the following are known to be correct?

Recapping co-classes and space complexity

Which of the following are known to be correct?
(Note: Mentimeter doesn't support latex, so interpret "co-L" as $\overline{L}$.)

Recapping co-classes and space complexity

Which of the following are known to be correct?
(Note: Mentimeter doesn't support latex, so interpret "co-L" as $\overline{L}$.)

- If $\overline{L} \in NP$ then $L \leq_p SAT$.

Recapping co-classes and space complexity

Which of the following are known to be correct?
(Note: Mentimeter doesn't support latex, so interpret "co-L" as $\overline{L}$.)

- If $\overline{L} \in NP$ then $L \leq_p SAT$.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$.

Recapping co-classes and space complexity

Which of the following are known to be correct?
(Note: Mentimeter doesn't support latex, so interpret "co-L" as $\overline{L}$.)

- If $\overline{L} \in NP$ then $L \leq_p SAT$.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$.
- If $\overline{L} \in NP$ then $L \leq_p \overline{SAT}$.

Recapping co-classes and space complexity

Which of the following are known to be correct?
(Note: Mentimeter doesn't support latex, so interpret "co-L" as $\overline{L}$.)

- If $\overline{L} \in NP$ then $L \leq_p SAT$.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$.
- If $\overline{L} \in NP$ then $L \leq_p \overline{SAT}$.
- $PSPACE = co-PSPACE$.

Recapping co-classes and space complexity

Which of the following are known to be correct?
(Note: Mentimeter doesn't support latex, so interpret "co-L" as $\overline{L}$.)

- If $\overline{L} \in NP$ then $L \leq_p SAT$.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$.
- If $\overline{L} \in NP$ then $L \leq_p \overline{SAT}$.
- $PSPACE = co-PSPACE$.
- Any decidable L satisfies $L \in PSPACE$.

Recapping co-classes and space complexity

Which of the following are known to be correct?
(Note: Mentimeter doesn't support latex, so interpret "co-L" as $\overline{L}$.)

- If $\overline{L} \in NP$ then $L \leq_p SAT$.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$.
- If $\overline{L} \in NP$ then $L \leq_p \overline{SAT}$.
- $PSPACE = co-PSPACE$.
- Any decidable L satisfies $L \in PSPACE$.
- $NP \cap co-NP = \emptyset$.

Recapping co-classes and space complexity

Which of the following are known to be correct?
(Note: Mentimeter doesn't support latex, so interpret "co-L" as $\overline{L}$.)

- If $\overline{L} \in NP$ then $L \leq_p SAT$.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$.
- If $\overline{L} \in NP$ then $L \leq_p \overline{SAT}$.
- $PSPACE = co-PSPACE$.
- Any decidable L satisfies $L \in PSPACE$.
- $NP \cap co-NP = \emptyset$.
- $NP \cap co-NP = P$.

Recapping co-classes and space complexity

Which of the following are known to be correct?
(Note: Mentimeter doesn't support latex, so interpret "co-L" as $\overline{L}$.)

- If $\overline{L} \in NP$ then $L \leq_p SAT$.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$.
- If $\overline{L} \in NP$ then $L \leq_p \overline{SAT}$.
- $PSPACE = co-PSPACE$.
- Any decidable L satisfies $L \in PSPACE$.
- $NP \cap co-NP = \emptyset$.
- $NP \cap co-NP = P$.
- If $L \in co-NP$ and $L \leq_p SAT$ then $NP = co-NP$.

Recapping co-classes and space complexity

Which of the following are known to be correct?
(Note: Mentimeter doesn't support latex, so interpret "co-L" as $\overline{L}$.)

- If $\overline{L} \in NP$ then $L \leq_p SAT$.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$.
- If $\overline{L} \in NP$ then $L \leq_p \overline{SAT}$.
- $PSPACE = co-PSPACE$.
- Any decidable L satisfies $L \in PSPACE$.
- $NP \cap co-NP = \emptyset$.
- $NP \cap co-NP = P$.
- If $L \in co-NP$ and $L \leq_p SAT$ then $NP = co-NP$.

Answer on Mentimeter:

Recapping co-classes and space complexity

Which of the following are known to be correct?

- If $\overline{L} \in NP$ then $L \leq_p SAT$.

Recapping co-classes and space complexity

Which of the following are known to be correct?

- If $\overline{L} \in NP$ then $L \leq_p SAT$. No.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$.

Recapping co-classes and space complexity

Which of the following are known to be correct?

- If $\overline{L} \in NP$ then $L \leq_p SAT$. No.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$. Yes, by the definition of NP-completeness.
- If $\overline{L} \in NP$ then $L \leq_p \overline{SAT}$.

Recapping co-classes and space complexity

Which of the following are known to be correct?

- If $\overline{L} \in NP$ then $L \leq_p SAT$. No.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$. Yes, by the definition of NP-completeness.
- If $\overline{L} \in NP$ then $L \leq_p \overline{SAT}$. Yes! with the same reduction.
- $PSPACE = co-PSPACE$.

Recapping co-classes and space complexity

Which of the following are known to be correct?

- If $\overline{L} \in NP$ then $L \leq_p SAT$. No.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$. Yes, by the definition of NP-completeness.
- If $\overline{L} \in NP$ then $L \leq_p \overline{SAT}$. Yes! with the same reduction.
- $PSPACE = co-PSPACE$. Yes, it's a deterministic class and we can flip states Y and N.
- Any decidable L satisfies $L \in PSPACE$.

Recapping co-classes and space complexity

Which of the following are known to be correct?

- If $\overline{L} \in NP$ then $L \leq_p SAT$. No.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$. Yes, by the definition of NP-completeness.
- If $\overline{L} \in NP$ then $L \leq_p \overline{SAT}$. Yes! with the same reduction.
- $PSPACE = co-PSPACE$. Yes, it's a deterministic class and we can flip states Y and N.
- Any decidable L satisfies $L \in PSPACE$. No. Some problems require more space.
- $NP \cap co-NP = \emptyset$.

Recapping co-classes and space complexity

Which of the following are known to be correct?

- If $\overline{L} \in NP$ then $L \leq_p SAT$. No.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$. Yes, by the definition of NP-completeness.
- If $\overline{L} \in NP$ then $L \leq_p \overline{SAT}$. Yes! with the same reduction.
- $PSPACE = co-PSPACE$. Yes, it's a deterministic class and we can flip states Y and N.
- Any decidable L satisfies $L \in PSPACE$. No. Some problems require more space.
- $NP \cap co-NP = \emptyset$. No. For example, $P \subseteq NP \cap co-NP$.
- $NP \cap co-NP = P$.

Recapping co-classes and space complexity

Which of the following are known to be correct?

- If $\overline{L} \in NP$ then $L \leq_p SAT$. No.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$. Yes, by the definition of NP-completeness.
- If $\overline{L} \in NP$ then $L \leq_p \overline{SAT}$. Yes! with the same reduction.
- $PSPACE = co-PSPACE$. Yes, it's a deterministic class and we can flip states Y and N.
- Any decidable L satisfies $L \in PSPACE$. No. Some problems require more space.
- $NP \cap co-NP = \emptyset$. No. For example, $P \subseteq NP \cap co-NP$.
- $NP \cap co-NP = P$. This is an open question.
- If $L \in co-NP$ and $L \leq_p SAT$ then $NP = co-NP$.

Recapping co-classes and space complexity

Which of the following are known to be correct?

- If $\overline{L} \in NP$ then $L \leq_p SAT$. No.
- If $\overline{L} \in NP$ then $\overline{L} \leq_p SAT$. Yes, by the definition of NP-completeness.
- If $\overline{L} \in NP$ then $L \leq_p \overline{SAT}$. Yes! with the same reduction.
- $PSPACE = co-PSPACE$. Yes, it's a deterministic class and we can flip states Y and N.
- Any decidable L satisfies $L \in PSPACE$. No. Some problems require more space.
- $NP \cap co-NP = \emptyset$. No. For example, $P \subseteq NP \cap co-NP$.
- $NP \cap co-NP = P$. This is an open question.
- If $L \in co-NP$ and $L \leq_p SAT$ then $NP = co-NP$. No. For example, any $L \in P \subseteq co-NP$ is reducible to SAT.

Lecture 11: The Ladder Game

We will see:

Lecture 11: The Ladder Game

We will see:

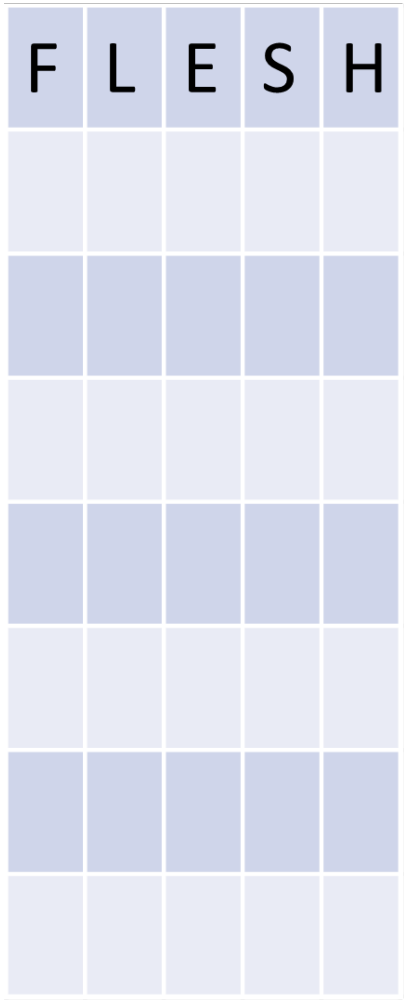
- A game.

Lecture 11: The Ladder Game

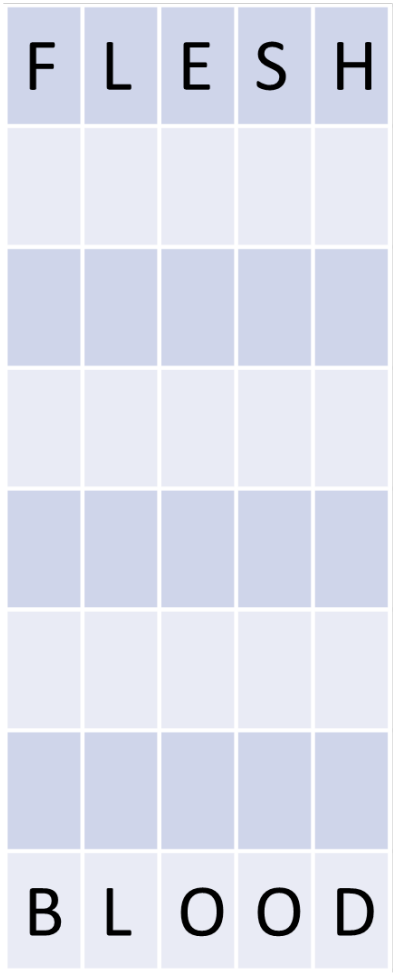
We will see:

- A game.
- And analyze its space complexity.

Word ladder



Word ladder



Word ladder

F	L	E	S	H
F	L	A	S	H
B	L	O	O	D

Word ladder

F	L	E	S	H
F	L	A	S	H
F	L	A	S	K
B	L	O	O	D

Word ladder

F	L	E	S	H
F	L	A	S	H
F	L	A	S	K
F	L	A	N	K
B	L	O	O	D

Word ladder

F	L	E	S	H
F	L	A	S	H
F	L	A	S	K
F	L	A	N	K
B	L	A	N	K
B	L	O	O	D

Word ladder

F	L	E	S	H
F	L	A	S	H
F	L	A	S	K
F	L	A	N	K
B	L	A	N	K
B	L	A	N	D
B	L	O	O	D

Word ladder

F	L	E	S	H
F	L	A	S	H
F	L	A	S	K
F	L	A	N	K
B	L	A	N	K
B	L	A	N	D
B	L	O	N	D
B	L	O	O	D

Word ladder

$$\text{LADDER} = \{\langle M, s, t\rangle \mid M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

F	L	E	S	H
B	L	O	O	D

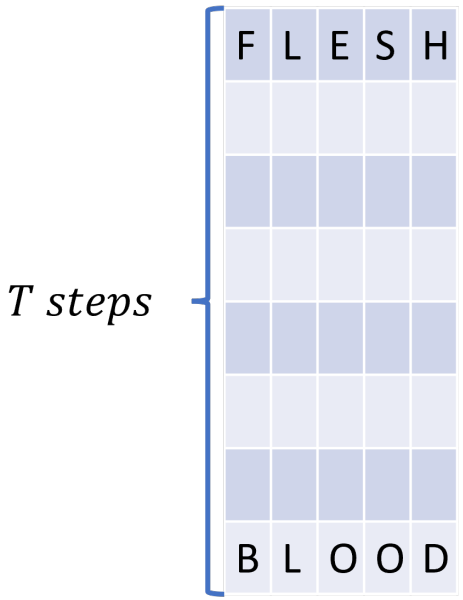
LADDER$\in$ *NPSPACE*

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

LADDER$\in$ *NPSPACE*

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

Observation: it is enough to consider ladders of length $T \leq |\Sigma|^{|s|}$.



LADDER$\in$ *NPSPACE*

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

Proof.

An algorithm:

LADDER$\in$ *NPSPACE*

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

Proof.

An algorithm:

Set $\ell = 0, w = s$

LADDER$\in$ *NPSPACE*

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

Proof.

An algorithm:

Set $\ell = 0, w = s$
While $(\ell \leq T)$

LADDER$\in$ *NPSPACE*

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

Proof.

An algorithm:

Set $\ell = 0, w = s$
While $(\ell \leq T)$
    If $(w = t)$
        Accept

LADDER$\in$ *NPSPACE*

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

Proof.

An algorithm:

Set $\ell = 0, w = s$
While $(\ell \leq T)$
    If $(w = t)$
        Accept
    Guess a word w'

LADDER$\in$ *NPSPACE*

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

Proof.

An algorithm:

Set $\ell = 0, w = s$
While $(\ell \leq T)$
    If $(w = t)$
        Accept
    Guess a word w'
    If ($w' \in L(M)$ and differs from w by one symbol)
        set $w = w', \ell = \ell + 1$

LADDER$\in$ *NPSPACE*

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

Proof.

An algorithm:

Set $\ell = 0, w = s$
While $(\ell \leq T)$
    If $(w = t)$
        Accept
    Guess a word w'
    If ($w' \in L(M)$ and differs from w by one symbol)
        set $w = w', \ell = \ell + 1$
Reject

LADDER$\in$ *NPSPACE*

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

Proof.

An algorithm:

Set $\ell = 0, w = s$
While $(\ell \leq T)$
    If $(w = t)$
        Accept
    Guess a word w'
    If ($w' \in L(M)$ and differs from w by one symbol)
        set $w = w', \ell = \ell + 1$
Reject

LADDER$\in$ *NPSPACE*

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

Proof.

An algorithm:

Set $\ell = 0, w = s$
While $(\ell \leq T)$
    If $(w = t)$
        Accept
    Guess a word w'
    If ($w' \in L(M)$ and differs from w by one symbol)
        set $w = w', \ell = \ell + 1$
Reject

This shows $LADDER \in NSPACE(n)$.

LADDER$\in$ *NPSPACE*

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

Proof.

An algorithm:

Set $\ell = 0, w = s$
While $(\ell \leq T)$
    If $(w = t)$
        Accept
    Guess a word w'
    If ($w' \in L(M)$ and differs from w by one symbol)
        set $w = w', \ell = \ell + 1$
Reject

This shows $LADDER \in NSPACE(n)$.
(We store 2 words and $\log(|\Sigma|^{|s|}) = |s| \log |\Sigma| = O(n)$-bits counter). □

LADDER$\in$ *PSPACE*!

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

LADDER$\in$ *PSPACE*!

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

We saw that $NP \subseteq PSPACE$, so why can't we just guess the ladder and conclude the result?

LADDER$\in$ *PSPACE*!

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

We saw that $NP \subseteq PSPACE$, so why can't we just guess the ladder and conclude the result?

Hint: what's the size of the ladder?

LADDER$\in$ *PSPACE*!

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

An algorithm:

LADDER$\in$ *PSPACE*!

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

An algorithm:

Function Reachable(a, b, k) # Can we reach from a to b in k steps?
    If $(k = 1)$
        Return (b can be reached from a in M)

LADDER$\in$ *PSPACE*!

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

An algorithm:

Function Reachable(a, b, k) # Can we reach from a to b in k steps?
    If $(k = 1)$
        Return (b can be reached from a in M)
    For each $m \in \Sigma^{|s|}$

LADDER$\in$ *PSPACE*!

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

An algorithm:

Function Reachable(a, b, k) # Can we reach from a to b in k steps?
    If $(k = 1)$
        Return (b can be reached from a in M)
    For each $m \in \Sigma^{|s|}$
        If (Reachable$(a, m, \lceil k/2 \rceil) \wedge$ Reachable$(m, b, \lfloor k/2 \rfloor)$)

LADDER$\in$ *PSPACE*!

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

An algorithm:

Function Reachable(a, b, k) # Can we reach from a to b in k steps?
    If $(k = 1)$
        Return (b can be reached from a in M)
    For each $m \in \Sigma^{|s|}$
        If (Reachable$(a, m, \lceil k/2 \rceil) \wedge$ Reachable$(m, b, \lfloor k/2 \rfloor)$)
            Return T
    Return F

LADDER$\in$ *PSPACE*!

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

An algorithm:

Function Reachable(a, b, k) # Can we reach from a to b in k steps?
    If $(k = 1)$
        Return (b can be reached from a in M)
    For each $m \in \Sigma^{|s|}$
        If (Reachable$(a, m, \lceil k/2 \rceil) \wedge$ Reachable$(m, b, \lfloor k/2 \rfloor)$)
            Return T
    Return F

To check if $\langle M, s, t\rangle \in$ LADDER we invoke Reachable$(s, t, |\Sigma|^{|s|})$.

LADDER$\in$ *PSPACE*!

$$\text{LADDER} = \{\langle M, s, t\rangle \quad | \quad M \text{ is a DFA over alphabet } \Sigma \text{ and } L(M) \text{ contains a ladder from } s \text{ to } t\}.$$

An algorithm:

Function Reachable(a, b, k) # Can we reach from a to b in k steps?
    If $(k = 1)$
        Return (b can be reached from a in M) $\vee$ $(a = b)$
    For each $m \in \Sigma^{|s|}$
        If (Reachable$(a, m, \lceil k/2 \rceil) \wedge$ Reachable$(m, b, \lfloor k/2 \rfloor)$)
            Return T
    Return F

To check if $\langle M, s, t\rangle \in$ LADDER we invoke Reachable$(s, t, |\Sigma|^{|s|})$.

How much memory did we use?

Function Reachable(a, b, k) # Can we reach from a to b in k steps?
    If ($k = 1$)
        Return (b can be reached from a in M) $\vee$ ($a = b$)
    For each $m \in \Sigma^{|s|}$
        If (Reachable($a, m, \lceil k/2 \rceil$)$\wedge$ Reachable($m, b, \lfloor k/2 \rfloor$))
            Return T
    Return F

How much memory did we use?

Function Reachable(a, b, k) # Can we reach from a to b in k steps?
    If ($k = 1$)
        Return (b can be reached from a in M) $\vee$ ($a = b$)
    For each $m \in \Sigma^{|s|}$
        If (Reachable($a, m, \lceil k/2 \rceil$)$\wedge$ Reachable($m, b, \lfloor k/2 \rfloor$))
            Return T
    Return F

- The depth of the recursion is $\log_2(|\sigma|^{|s|}) = |s| \log |\Sigma| = O(n)$.

How much memory did we use?

Function Reachable(a, b, k) # Can we reach from a to b in k steps?
  If $(k = 1)$
    Return (b can be reached from a in M) $\vee$ $(a = b)$
  For each $m \in \Sigma^{|s|}$
    If (Reachable$(a, m, \lceil k/2 \rceil) \wedge$ Reachable$(m, b, \lfloor k/2 \rfloor)$)
      Return T
  Return F

- The depth of the recursion is $\log_2(|\sigma|^{|s|}) = |s| \log |\Sigma| = O(n)$.
- Each time we store the word m, where $|m| = |s| = O(n)$.

How much memory did we use?

Function Reachable(a, b, k) # Can we reach from a to b in k steps?
  If $(k = 1)$
    Return (b can be reached from a in M) $\vee$ $(a = b)$
  For each $m \in \Sigma^{|s|}$
    If (Reachable$(a, m, \lceil k/2 \rceil) \wedge$ Reachable$(m, b, \lfloor k/2 \rfloor)$)
      Return T
  Return F

- The depth of the recursion is $\log_2(|\sigma|^{|s|}) = |s| \log |\Sigma| = O(n)$.
- Each time we store the word m, where $|m| = |s| = O(n)$.

Therefore, we showed that $LADDER \in SPACE(n^2)$, and in particular, $LADDER \in PSPACE$.

How much memory did we use?

Function Reachable(a, b, k) # Can we reach from a to b in k steps?
  If $(k = 1)$
    Return (b can be reached from a in M) $\vee$ $(a = b)$
  For each $m \in \Sigma^{|s|}$
    If (Reachable$(a, m, \lceil k/2 \rceil) \wedge$ Reachable$(m, b, \lfloor k/2 \rfloor)$)
      Return T
  Return F

- The depth of the recursion is $\log_2(|\sigma|^{|s|}) = |s| \log |\Sigma| = O(n)$.
- Each time we store the word m, where $|m| = |s| = O(n)$.

Therefore, we showed that $LADDER \in SPACE(n^2)$, and in particular, $LADDER \in PSPACE$. We will see that this phenomenon is much more general.

Lecture 11: Recap

Today we saw:

Lecture 11: Recap

Today we saw:

- That the ladder game is in NSPACE.

Lecture 11: Recap

Today we saw:

- That the ladder game is in NSPACE.
- That using a recursion to break the problem into halves we can solve the problem deterministically while reusing the space.

Lecture 11: Recap

Today we saw:

- That the ladder game is in NSPACE.
- That using a recursion to break the problem into halves we can solve the problem deterministically while reusing the space.

Next: Savitch's Theorem and PSPACE=NPSPACE.