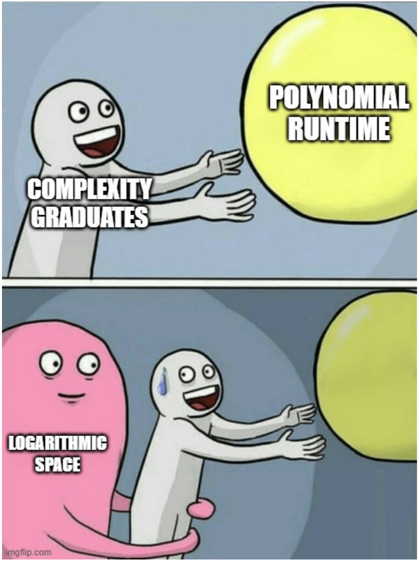
Lecture 13: Log space



Lecture 13: Log space

We will:

Lecture 13: Log space

We will:

- Define log-space (deterministic and ND) Turing machines.

Lecture 13: Log space

We will:

- Define log-space (deterministic and ND) Turing machines.

- Show that $\boldsymbol{L} \subseteq P$, We will:

- Define log-space (deterministic and ND) Turing machines.
- Show that $\boldsymbol{L} \subseteq P$,
- And also that $\boldsymbol{NL} \subseteq P$.

Log-space

We have defined the space complexity as the maximal entry on the tape written to.

Log-space

We have defined the space complexity as the maximal entry on the tape written to.

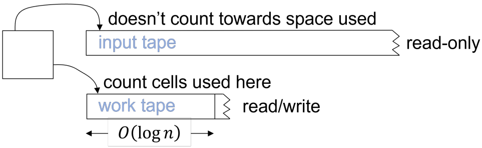
But the input itself has n characters, making it unreasonable to discuss $o(n)$ space with the current definition.

Log-space

We have defined the space complexity as the maximal entry on the tape written to.

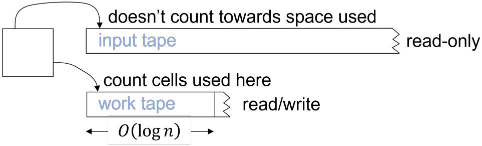
But the input itself has n characters, making it unreasonable to discuss $o(n)$ space with the current definition.

Model: a 2-tape TM with a read-only tape for the input and a RW-tape for computation.



Credit: Sipser

Log-space

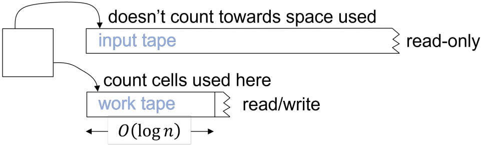


Credit: Sipser

Definition

$\boldsymbol{L} = SPACE(\log n)$

Log-space

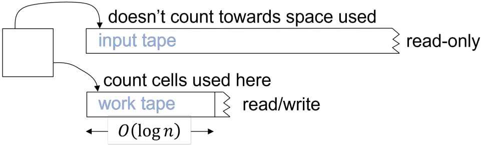


Credit: Sipser

Definition

$\boldsymbol{L} = SPACE(\log n) \qquad \boldsymbol{NL} = NSPACE(\log n)$

Log-space



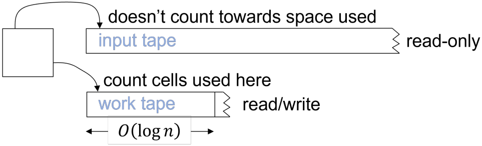
Credit: Sipser

Definition

$\mathbf{L} = SPACE(\log n)$ $\quad$ $\mathbf{NL} = NSPACE(\log n)$

For example, we can store $O(1)$ pointers.

Log-space



Credit: Sipser

Definition

$\mathbf{L} = SPACE(\log n) \qquad \mathbf{NL} = NSPACE(\log n)$

For example, we can store $O(1)$ pointers.

Another example: $\{ww^R | w \in \Sigma^*\} \in \mathbf{L}$.

A more interesting example

We show that finding the maximum of an array is in $\boldsymbol{L}$.

A more interesting example

We show that finding the maximum of an array is in $\boldsymbol{L}$.
Our input has parameters w, q, A such that $A = A[1], \ldots, A[q]$ is an array of q integers, each with w bits.

Algorithm.

A more interesting example

We show that finding the maximum of an array is in $\boldsymbol{L}$.
Our input has parameters w, q, A such that $A = A[1], \ldots, A[q]$ is an array of q integers, each with w bits.

Algorithm.

- Set $M \leftarrow A[1], m \leftarrow 1, i \leftarrow 2$.

A more interesting example

We show that finding the maximum of an array is in $\boldsymbol{L}$.
Our input has parameters w, q, A such that $A = A[1], \ldots, A[q]$ is an array of q integers, each with w bits.

Algorithm.

- Set $M \leftarrow A[1], m \leftarrow 1, i \leftarrow 2$.
- For $i = 2, 3, \ldots, q$

A more interesting example

We show that finding the maximum of an array is in $\boldsymbol{L}$.
Our input has parameters w, q, A such that $A = A[1], \ldots, A[q]$ is an array of q integers, each with w bits.

Algorithm.

- Set $M \leftarrow A[1], m \leftarrow 1, i \leftarrow 2$.
- For $i = 2, 3, \ldots, q$
  If $(A[i] > M)$

A more interesting example

We show that finding the maximum of an array is in $\boldsymbol{L}$.
Our input has parameters w, q, A such that $A = A[1], \ldots, A[q]$ is an array of q integers, each with w bits.

Algorithm.

- Set $M \leftarrow A[1], m \leftarrow 1, i \leftarrow 2$.
- For $i = 2, 3, \ldots, q$
  If $(A[i] > M)$
  $M \leftarrow A[i]$
  $m \leftarrow i$

A more interesting example

We show that finding the maximum of an array is in $\boldsymbol{L}$.
Our input has parameters w, q, A such that $A = A[1], \ldots, A[q]$ is an array of q integers, each with w bits.

Algorithm.

- Set $M \leftarrow A[1], m \leftarrow 1, i \leftarrow 2.$
- For $i = 2, 3, \ldots, q$
  If $(A[i] > M)$
    $M \leftarrow A[i]$
    $m \leftarrow i$
  Return i

A more interesting example

We show that finding the maximum of an array is in $\boldsymbol{L}$.
Our input has parameters w, q, A such that $A = A[1], \ldots, A[q]$ is an array of q integers, each with w bits.

Algorithm.

- Set $M \leftarrow A[1], m \leftarrow 1, i \leftarrow 2$.
- For $i = 2, 3, \ldots, q$
  If $(A[i] > M)$
    $M \leftarrow A[i]$
    $m \leftarrow i$
  Return i

Seems easy enough.

A more interesting example

We show that finding the maximum of an array is in $\boldsymbol{L}$.
Our input has parameters w, q, A such that $A = A[1], \ldots, A[q]$ is an array of q integers, each with w bits.

Algorithm.

- Set $M \leftarrow A[1], m \leftarrow 1, i \leftarrow 2$.
- For $i = 2, 3, \ldots, q$
  If $(A[i] > M)$
  $M \leftarrow A[i]$
  $m \leftarrow i$
  Return i

Seems easy enough.
But how is this implemented?

Our input size is $n = O(qw)$.
We are not allowed to keep w bits (for M) in memory!

A more interesting example

Algorithm.

- Set $m \leftarrow 1, i \leftarrow 2$.
- For $i = 2, 3, \ldots, q$

  If $(A[i] > A[m])$

  $m \leftarrow i$

  Return i

A more interesting example

Algorithm.

- Set $m \leftarrow 1, i \leftarrow 2$.
- For $i = 2, 3, \ldots, q$

  If $(A[i] > A[m])$

  $m \leftarrow i$

  Return i

$(A[i] > A[m])$ is checked in log space by going bit by bit from the MSB.

A more interesting example

Algorithm.

- Set $m \leftarrow 1, i \leftarrow 2$.
- For $i = 2, 3, \ldots, q$

  If $(A[i] > A[m])$

  $m \leftarrow i$

  Return i

$(A[i] > A[m])$ is checked in log space by going bit by bit from the MSB.

0	0	0	1	1	0	1	0

0	0	0	1	0	0	1	0

A more interesting example

Algorithm.

- Set $m \leftarrow 1, i \leftarrow 2$.
- For $i = 2, 3, \dots, q$

  If $(A[i] > A[m])$

  $m \leftarrow i$

  Return i

$(A[i] > A[m])$ is checked in log space by going bit by bit from the MSB.

0	0	0	1	1	0	1	0

0	0	0	1	0	0	1	0

A more interesting example

Algorithm.

- Set $m \leftarrow 1, i \leftarrow 2$.
- For $i = 2, 3, \ldots, q$

  If $(A[i] > A[m])$

  $m \leftarrow i$

  Return i

$(A[i] > A[m])$ is checked in log space by going bit by bit from the MSB.

0	0	0	1	1	0	1	0

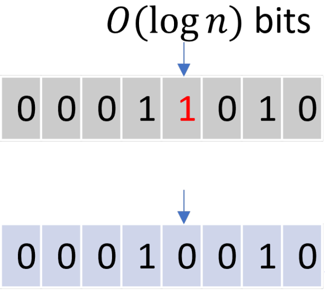
0	0	0	1	0	0	1	0

A more interesting example

Algorithm.

- Set $m \leftarrow 1, i \leftarrow 2$.
- For $i = 2, 3, \ldots, q$
  If $(A[i] > A[m])$
  $m \leftarrow i$
  Return i

$(A[i] > A[m])$ is checked in log space by going bit by bit from the MSB.



Another example

Definition

Given a directed graph $G = (V, E)$ and two vertices s, t, the STCON problem decides whether there exists a path from s to t.

Another example

Definition

Given a directed graph $G = (V, E)$ and two vertices s, t, the STCON problem decides whether there exists a path from s to t.

Lemma

$STCON \in \boldsymbol{NL}$.

Another example

Lemma

$STCON \in \textbf{\textit{NL}}$.

Proof.

Another example

Lemma

$STCON \in \textbf{NL}$.

Proof.

We are given $G = (V, E)$ and $s, t \in V$.

Another example

Lemma

$STCON \in \mathbf{NL}$.

Proof.

We are given $G = (V, E)$ and $s, t \in V$. Here's an algorithm:

Algorithm 1 STCON

1: Set $v \leftarrow s$.

Another example

Lemma

$STCON \in \textbf{NL}$.

Proof.

We are given $G = (V, E)$ and $s, t \in V$. Here's an algorithm:

Algorithm 1 STCON

1: Set $v \leftarrow s$.
2: **for** $\ell = 0$ to $|V| - 1$ **do**

Another example

Lemma

$STCON \in \textbf{NL}$.

Proof.

We are given $G = (V, E)$ and $s, t \in V$. Here's an algorithm:

Algorithm 1 STCON

```
Set v ← s.
for ℓ = 0 to |V| − 1 do
    if v = t then
```

Another example

Lemma

STCON $\in$ ***NL***.

Proof.

We are given $G = (V, E)$ and $s, t \in V$. Here's an algorithm:

Algorithm 1 STCON

```
Set v ← s.
for ℓ = 0 to |V| − 1 do
    if v = t then
        accept
```

Another example

Lemma

$STCON \in \boldsymbol{NL}$.

Proof.

We are given $G = (V, E)$ and $s, t \in V$. Here's an algorithm:

Algorithm 1 STCON

```
Set v ← s.
for ℓ = 0 to |V| − 1 do
    if v = t then
        accept
    Guess u ∈ V such that (v, u) ∈ E.
```

Another example

Lemma

$STCON \in \boldsymbol{NL}$.

Proof.

We are given $G = (V, E)$ and $s, t \in V$. Here's an algorithm:

Algorithm 1 STCON

Set $v \leftarrow s$.
for $\ell = 0$ to $|V| - 1$ **do**
  if $v = t$ **then**
    accept
  Guess $u \in V$ such that $(v, u) \in E$.
  Set $v \leftarrow u$
Reject

Another example

Lemma

STCON $\in$ ***NL***.

Proof.

We are given $G = (V, E)$ and $s, t \in V$. Here's an algorithm:

Algorithm 1 STCON

```
Set v ← s.
for ℓ = 0 to |V| − 1 do
    if v = t then
        accept
    Guess u ∈ V such that (v, u) ∈ E.
    Set v ← u
Reject
```

Notice that ℓ and v both require $O(\log n)$ bits and thus this can be implemented on an ***L*** machine. $\square$

$\boldsymbol{L} \subseteq P$

Theorem

$\boldsymbol{L} \subseteq P$.

$\boldsymbol{L} \subseteq P$

Theorem

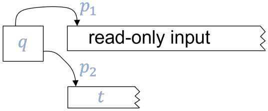
$\boldsymbol{L} \subseteq P$.

For $\boldsymbol{L}$ machines, we can refine the definition of a configuration.

Theorem

$L \subseteq P$.

For L machines, we can refine the definition of a configuration. Since the input tape is read-only, we denote a configuration by (q, p_1, p_2, t), where:



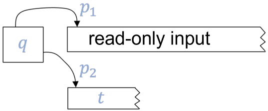
Credit: Sipser

$L \subseteq P$

Theorem

$\boldsymbol{L} \subseteq P$.

For $\boldsymbol{L}$ machines, we can refine the definition of a configuration. Since the input tape is read-only, we denote a configuration by (q, p_1, p_2, t), where:



Credit: Sipser

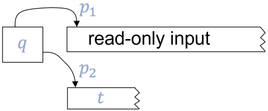
- q is the current state.

$\boldsymbol{L} \subseteq P$

Theorem

$\boldsymbol{L} \subseteq P$.

For $\boldsymbol{L}$ machines, we can refine the definition of a configuration. Since the input tape is read-only, we denote a configuration by (q, p_1, p_2, t), where:



Credit: Sipser

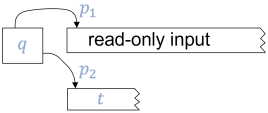
- q is the current state.
- p_1 is the location of the input-tape-head.

$\boldsymbol{L} \subseteq P$

Theorem

$\boldsymbol{L} \subseteq P$.

For $\boldsymbol{L}$ machines, we can refine the definition of a configuration. Since the input tape is read-only, we denote a configuration by (q, p_1, p_2, t), where:



Credit: Sipser

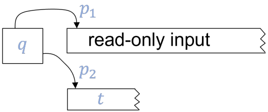
- q is the current state.
- p_1 is the location of the input-tape-head.
- p_2 is the location of the RW-tape-head.

$\boldsymbol{L} \subseteq P$

Theorem

$\boldsymbol{L} \subseteq P$.

For $\boldsymbol{L}$ machines, we can refine the definition of a configuration. Since the input tape is read-only, we denote a configuration by (q, p_1, p_2, t), where:



Credit: Sipser

- q is the current state.
- p_1 is the location of the input-tape-head.
- p_2 is the location of the RW-tape-head.
- $t \in \Gamma^{O(\log n)}$ is the contents of the RW tape.

$L \subseteq P$

Theorem

$\boldsymbol{L} \subseteq P$.

We denote a configuration by (q, p_1, p_2, t).

$\boldsymbol{L} \subseteq P$

Theorem

$\boldsymbol{L} \subseteq P$.

We denote a configuration by (q, p_1, p_2, t).

Proof.

How many configurations can we have?

$L \subseteq P$

Theorem

$\boldsymbol{L} \subseteq P$.

We denote a configuration by (q, p_1, p_2, t).

Proof.

How many configurations can we have?
$|Q| \times n \times \log n \times |\Gamma|^{O(\log n)} = n^{O(1)}$. □

$\boldsymbol{NL} \subseteq SPACE(\log^2 n)$

Theorem

$\boldsymbol{NL} \subseteq SPACE(\log^2 n)$.

$\boldsymbol{NL} \subseteq \mathit{SPACE}(\log^2 n)$

Theorem

$\boldsymbol{NL} \subseteq \mathit{SPACE}(\log^2 n)$.

Proof.

Immediate from Savitch's theorem. $\square$

$\boldsymbol{NL} \subseteq P$

Theorem

$\boldsymbol{NL} \subseteq P$.

$\boldsymbol{NL} \subseteq P$

Theorem

$\boldsymbol{NL} \subseteq P$.

Note that a straightforward simulation does not work here.

$\boldsymbol{NL} \subseteq P$

Theorem

$\boldsymbol{NL} \subseteq P$.

Note that a straightforward simulation does not work here.

To simulate a general $SPACE(\log^2 n)$ machine we need $2^{\Omega(\log^2 n)}$

$\boldsymbol{NL} \subseteq P$

Theorem

$\boldsymbol{NL} \subseteq P$.

Note that a straightforward simulation does not work here.

To simulate a general $SPACE(\log^2 n)$ machine we need $2^{\Omega(\log^2 n)} = n^{\Omega(\log n)}$ time, which is not polynomial.

$\boldsymbol{NL} \subseteq P$

Theorem

$\boldsymbol{NL} \subseteq P$.

Proof.

Let $L \in \boldsymbol{NL}$ and let M be an $\boldsymbol{NL}$ NDTM for it.

$\textbf{\textit{NL}} \subseteq P$

Theorem

$\textbf{\textit{NL}} \subseteq P$.

Proof.

Let $L \in \textbf{\textit{NL}}$ and let M be an $\textbf{\textit{NL}}$ NDTM for it. We will show $L \in P$.

$\boldsymbol{NL} \subseteq P$

Theorem

$\boldsymbol{NL} \subseteq P$.

Proof.

Let $L \in \boldsymbol{NL}$ and let M be an $\boldsymbol{NL}$ NDTM for it. We will show $L \in P$.
Consider the set of configurations

$$C = \{(q, p_1, p_2, t) \in Q \times \{0, \dots, n-1\} \times \{0, \dots, O(\log n)\} \times \Gamma^{O(\log n)}\}.$$

$NL \subseteq P$

Theorem

$\boldsymbol{NL} \subseteq P$.

Proof.

Let $L \in \boldsymbol{NL}$ and let M be an $\boldsymbol{NL}$ NDTM for it. We will show $L \in P$. Consider the set of configurations

$$C = \{(q, p_1, p_2, t) \in Q \times \{0, \ldots, n-1\} \times \{0, \ldots, O(\log n)\} \times \Gamma^{O(\log n)}\}.$$

Construct a graph with C as vertices and connect each pair $(c_i, c_j) \in C^2$ if c_j follows from c_i in one step.

$\boldsymbol{NL} \subseteq P$

Theorem

$\boldsymbol{NL} \subseteq P$.

Proof.

Let $L \in \boldsymbol{NL}$ and let M be an $\boldsymbol{NL}$ NDTM for it. We will show $L \in P$.
Consider the set of configurations

$$C = \{(q, p_1, p_2, t) \in Q \times \{0, \dots, n-1\} \times \{0, \dots, O(\log n)\} \times \Gamma^{O(\log n)}\}.$$

Construct a graph with C as vertices and connect each pair $(c_i, c_j) \in C^2$ if c_j follows from c_i in one step.
Notice that $|C| = 2^{O(\log n)} = n^{O(1)}$, and thus the resulting graph is of size $O(|C|^2) = n^{O(1)}$.

Theorem

$\textbf{\textit{NL}} \subseteq P$.

Proof.

Let $L \in \textbf{\textit{NL}}$ and let M be an $\textbf{\textit{NL}}$ NDTM for it. We will show $L \in P$.
Consider the set of configurations

$$C = \{(q, p_1, p_2, t) \in Q \times \{0, \ldots, n-1\} \times \{0, \ldots, O(\log n)\} \times \Gamma^{O(\log n)}\}.$$

Construct a graph with C as vertices and connect each pair $(c_i, c_j) \in C^2$ if c_j follows from c_i in one step.
Notice that $|C| = 2^{O(\log n)} = n^{O(1)}$, and thus the resulting graph is of size $O(|C|^2) = n^{O(1)}$.
We then run a BFS to check if c_Y is reachable from c_0 and terminate accordingly. $\square$

Trivia

(Not required for the exam.)

Trivia

(Not required for the exam.)

- There is a notion of NL-completeness and hardness

Trivia

(Not required for the exam.)

- There is a notion of NL-completeness and hardness(using $\boldsymbol{L}$-reductions.)

Trivia

(Not required for the exam.)

- There is a notion of NL-completeness and hardness(using $\boldsymbol{L}$-reductions.)
- STCON is NL-complete.

Trivia

(Not required for the exam.)

- There is a notion of NL-completeness and hardness(using $\boldsymbol{L}$-reductions.)
- STCON is NL-complete.
- STCON was shown to also be in co-NL (by showing that $\overline{STCON} \in \boldsymbol{NL}$), thus proving that $\boldsymbol{NL} = co-\boldsymbol{NL}$.

Trivia

(Not required for the exam.)

- There is a notion of NL-completeness and hardness(using $\boldsymbol{L}$-reductions.)
- STCON is NL-complete.
- STCON was shown to also be in co-NL (by showing that $\overline{STCON} \in \boldsymbol{NL}$), thus proving that $\boldsymbol{NL} = co-\boldsymbol{NL}$.
- It was shown that undirected STCON is in $\boldsymbol{L}$.

Trivia

(Not required for the exam.)

- There is a notion of NL-completeness and hardness(using $\boldsymbol{L}$-reductions.)
- STCON is NL-complete.
- STCON was shown to also be in co-NL (by showing that $\overline{STCON} \in \boldsymbol{NL}$), thus proving that $\boldsymbol{NL} = co{-}\boldsymbol{NL}$.
- It was shown that undirected STCON is in $\boldsymbol{L}$.
- There are also P-complete problems (using $\boldsymbol{L}$-reductions) such as max-flow.

Let's recap

Which of the following are correct?

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps.
- $STCON \in \boldsymbol{L}$.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps.
- $STCON \in \boldsymbol{L}$.
- $\overline{STCON} \in \boldsymbol{L}$.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps.
- $STCON \in \boldsymbol{L}$.
- $\overline{STCON} \in \boldsymbol{L}$.
- Deciding if there exists a path (not necessarily simple) of length k in a graph is in $\boldsymbol{NL}$.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps.
- $STCON \in \boldsymbol{L}$.
- $\overline{STCON} \in \boldsymbol{L}$.
- Deciding if there exists a path (not necessarily simple) of length k in a graph is in $\boldsymbol{NL}$.
- Given $G = (V, E)$ and $v \in V$, deciding if v is on a cycle is in $\boldsymbol{NL}$.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps.
- $STCON \in \boldsymbol{L}$.
- $\overline{STCON} \in \boldsymbol{L}$.
- Deciding if there exists a path (not necessarily simple) of length k in a graph is in $\boldsymbol{NL}$.
- Given $G = (V, E)$ and $v \in V$, deciding if v is on a cycle is in $\boldsymbol{NL}$.

Answer on Mentimeter: .

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$. Correct.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$. Correct.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping. No, after $n^{O(1)}$ steps it must be in a loop from configuration count.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$. Correct.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping. No, after $n^{O(1)}$ steps it must be in a loop from configuration count.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps. No, it can make $n^{O(1)}$ steps without looping.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$. Correct.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping. No, after $n^{O(1)}$ steps it must be in a loop from configuration count.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps. No, it can make $n^{O(1)}$ steps without looping.
- $STCON \in \boldsymbol{L}$. Probably not; all we know is that $STCON \in \boldsymbol{NL}$.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$. Correct.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping. No, after $n^{O(1)}$ steps it must be in a loop from configuration count.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps. No, it can make $n^{O(1)}$ steps without looping.
- $STCON \in \boldsymbol{L}$. Probably not; all we know is that $STCON \in \boldsymbol{NL}$.
- $\overline{STCON} \in \boldsymbol{L}$. Probably not; it's the same as with the last one, $\boldsymbol{L}$ is a deterministic class.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$. Correct.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping. No, after $n^{O(1)}$ steps it must be in a loop from configuration count.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps. No, it can make $n^{O(1)}$ steps without looping.
- $STCON \in \boldsymbol{L}$. Probably not; all we know is that $STCON \in \boldsymbol{NL}$.
- $\overline{STCON} \in \boldsymbol{L}$. Probably not; it's the same as with the last one, $\boldsymbol{L}$ is a deterministic class.
- Deciding if there is a path (not necessarily simple) of length k in a graph is in $\boldsymbol{NL}$. Correct.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$. Correct.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping. No, after $n^{O(1)}$ steps it must be in a loop from configuration count.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps. No, it can make $n^{O(1)}$ steps without looping.
- $STCON \in \boldsymbol{L}$. Probably not; all we know is that $STCON \in \boldsymbol{NL}$.
- $\overline{STCON} \in \boldsymbol{L}$. Probably not; it's the same as with the last one, $\boldsymbol{L}$ is a deterministic class.
- Deciding if there is a path (not necessarily simple) of length k in a graph is in $\boldsymbol{NL}$. Correct. But tricky.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$. Correct.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping. No, after $n^{O(1)}$ steps it must be in a loop from configuration count.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps. No, it can make $n^{O(1)}$ steps without looping.
- $STCON \in \boldsymbol{L}$. Probably not; all we know is that $STCON \in \boldsymbol{NL}$.
- $\overline{STCON} \in \boldsymbol{L}$. Probably not; it's the same as with the last one, $\boldsymbol{L}$ is a deterministic class.
- Deciding if there is a path (not necessarily simple) of length k in a graph is in $\boldsymbol{NL}$. Correct. But tricky. k can be represented using $\log k$ bits (thus $n = O(|V| + |E| + \log k)$);

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$. Correct.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping. No, after $n^{O(1)}$ steps it must be in a loop from configuration count.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps. No, it can make $n^{O(1)}$ steps without looping.
- $STCON \in \boldsymbol{L}$. Probably not; all we know is that $STCON \in \boldsymbol{NL}$.
- $\overline{STCON} \in \boldsymbol{L}$. Probably not; it's the same as with the last one, $\boldsymbol{L}$ is a deterministic class.
- Deciding if there is a path (not necessarily simple) of length k in a graph is in $\boldsymbol{NL}$. Correct. But tricky. k can be represented using $\log k$ bits (thus $n = O(|V| + |E| + \log k)$); but if $k \leq |V|$, we can still use $\log |V|$ bits counter;

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$. Correct.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping. No, after $n^{O(1)}$ steps it must be in a loop from configuration count.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps. No, it can make $n^{O(1)}$ steps without looping.
- $STCON \in \boldsymbol{L}$. Probably not; all we know is that $STCON \in \boldsymbol{NL}$.
- $\overline{STCON} \in \boldsymbol{L}$. Probably not; it's the same as with the last one, $\boldsymbol{L}$ is a deterministic class.
- Deciding if there is a path (not necessarily simple) of length k in a graph is in $\boldsymbol{NL}$. Correct. But tricky. k can be represented using $\log k$ bits (thus $n = O(|V| + |E| + \log k)$); but if $k \leq |V|$, we can still use $\log |V|$ bits counter; and if $k \geq |V|$, it's enough to check for path of length $|V|$.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$. Correct.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping. No, after $n^{O(1)}$ steps it must be in a loop from configuration count.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps. No, it can make $n^{O(1)}$ steps without looping.
- $STCON \in \boldsymbol{L}$. Probably not; all we know is that $STCON \in \boldsymbol{NL}$.
- $\overline{STCON} \in \boldsymbol{L}$. Probably not; it's the same as with the last one, $\boldsymbol{L}$ is a deterministic class.
- Deciding if there is a path (not necessarily simple) of length k in a graph is in $\boldsymbol{NL}$. Correct. But tricky. k can be represented using $\log k$ bits (thus $n = O(|V| + |E| + \log k)$); but if $k \leq |V|$, we can still use $\log |V|$ bits counter; and if $k \geq |V|$, it's enough to check for path of length $|V|$. Remember to think about the length of the input!

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$. Correct.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping. No, after $n^{O(1)}$ steps it must be in a loop from configuration count.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps. No, it can make $n^{O(1)}$ steps without looping.
- $STCON \in \boldsymbol{L}$. Probably not; all we know is that $STCON \in \boldsymbol{NL}$.
- $\overline{STCON} \in \boldsymbol{L}$. Probably not; it's the same as with the last one, $\boldsymbol{L}$ is a deterministic class.
- Deciding if there is a path (not necessarily simple) of length k in a graph is in $\boldsymbol{NL}$. Correct. But tricky. k can be represented using $\log k$ bits (thus $n = O(|V| + |E| + \log k)$); but if $k \leq |V|$, we can still use $\log |V|$ bits counter; and if $k \geq |V|$, it's enough to check for path of length $|V|$. Remember to think about the length of the input!
- Given $G = (V, E)$ and $v \in V$, deciding if v is on a cycle is in $\boldsymbol{NL}$. Correct.

Let's recap

Which of the following are correct?

- $\boldsymbol{L} \subseteq \boldsymbol{NL} \subseteq P \subseteq NP \subseteq PSPACE$. Correct.
- An $\boldsymbol{L}$ machine can run $2^{O(n)}$ steps without looping. No, after $n^{O(1)}$ steps it must be in a loop from configuration count.
- An $\boldsymbol{L}$ machine must be looping after $\log^2 n$ steps. No, it can make $n^{O(1)}$ steps without looping.
- $STCON \in \boldsymbol{L}$. Probably not; all we know is that $STCON \in \boldsymbol{NL}$.
- $\overline{STCON} \in \boldsymbol{L}$. Probably not; it's the same as with the last one, $\boldsymbol{L}$ is a deterministic class.
- Deciding if there is a path (not necessarily simple) of length k in a graph is in $\boldsymbol{NL}$. Correct. But tricky. k can be represented using $\log k$ bits (thus $n = O(|V| + |E| + \log k)$); but if $k \leq |V|$, we can still use $\log |V|$ bits counter; and if $k \geq |V|$, it's enough to check for path of length $|V|$. Remember to think about the length of the input!
- Given $G = (V, E)$ and $v \in V$, deciding if v is on a cycle is in $\boldsymbol{NL}$. Correct.

Next time: hierarchy theorems and EXPTIME-completeness.