Lecture 17

NL = coNL

Definition: $\operatorname{coNL} = \{L \mid \overline{L} \in \operatorname{NL}\}$

Definition: $\operatorname{coNL} = \{L \mid \overline{L} \in \operatorname{NL}\}$

Example: $\overline{PATH} \in coNL$.

Definition: $\operatorname{coNL} = \{L \mid \overline{L} \in \operatorname{NL}\}$

Example: $\overline{PATH} \in coNL$.

Definition: $coNL = \{L \mid \overline{L} \in NL\}$

Example: $\overline{PATH} \in coNL$.

Claim: $\overline{PATH} \in NL \implies NL = coNL$

Definition: $\operatorname{coNL} = \{L \mid \overline{L} \in \operatorname{NL}\}$

Example: $\overline{PATH} \in coNL$.

 $\overline{PATH} = \{ \langle G, s, t \rangle \mid G \text{ is a directed graph s.t. there is no path from s to t} \}$

Claim: $\overrightarrow{PATH} \in NL \implies NL = coNL$ **Proof:**

Definition: $coNL = \{L \mid \overline{L} \in NL\}$

Example: $\overline{PATH} \in coNL$.

Claim: $\overline{PATH} \in NL \implies NL = coNL$ **Proof:** $coNL \subseteq NL$:

Definition: $coNL = \{L \mid \overline{L} \in NL\}$

Example: $\overline{PATH} \in coNL$.

Claim: $\overline{PATH} \in NL \implies NL = coNL$ **Proof:** $coNL \subseteq NL$:

Let $L \in coNL$.

Definition: $coNL = \{L \mid \overline{L} \in NL\}$

Example: $\overline{PATH} \in coNL$.

Claim: $\overline{PATH} \in NL \implies NL = coNL$ **Proof:** $coNL \subseteq NL$:

Let $L \in coNL$. Then, $L \leq_l \overline{PATH}$.

Definition: $coNL = \{L \mid \overline{L} \in NL\}$

Example: $\overline{PATH} \in coNL$.

Claim: $\overline{PATH} \in NL \implies NL = coNL$ **Proof:** $coNL \subseteq NL$:

Let $L \in coNL$. Then, $L \leq_l \overline{PATH}$. ($\because \overline{PATH}$ is coNL-complete)

Definition: $coNL = \{L \mid \overline{L} \in NL\}$

Example: $\overline{PATH} \in coNL$.

Claim: $\overline{PATH} \in NL \implies NL = coNL$ **Proof:** $coNL \subseteq NL$:

Let $L \in coNL$. Then, $L \leq_1 \overline{PATH}$. ($\because \overline{PATH}$ is coNL-complete) **NL** machine for L will first reduce L to **PATH** and then use **NL** machine of **PATH**.



Definition: $coNL = \{L \mid \overline{L} \in NL\}$

Example: $\overline{PATH} \in coNL$.

Claim: $\overline{PATH} \in NL \implies NL = coNL$ **Proof:** $coNL \subseteq NL$:

Let $L \in coNL$. Then, $L \leq_1 \overline{PATH}$. ($\because \overline{PATH}$ is coNL-complete)

NL machine for L will first reduce L to \overline{PATH} and then use **NL** machine of \overline{PATH} .

 $NL \subseteq coNL$:



Definition: $coNL = \{L \mid \overline{L} \in NL\}$

Example: $\overline{PATH} \in coNL$.

Claim: $\overline{PATH} \in NL \implies NL = coNL$ **Proof:** $coNL \subseteq NL$:

Let $L \in coNL$. Then, $L \leq_1 \overline{PATH}$. ($\because \overline{PATH}$ is coNL-complete)

NL machine for L will first reduce L to \overline{PATH} and then use **NL** machine of \overline{PATH} .

 $NL \subseteq coNL$:

 $L \in \mathsf{NL}$



Definition: $\operatorname{coNL} = \{L \mid \overline{L} \in \operatorname{NL}\}$

Example: $\overline{PATH} \in coNL$.

Claim: $\overline{PATH} \in NL \implies NL = coNL$ **Proof:** $coNL \subseteq NL$:

Let $L \in coNL$. Then, $L \leq_1 \overline{PATH}$. ($\because \overline{PATH}$ is coNL-complete)

NL machine for L will first reduce L to \overline{PATH} and then use **NL** machine of \overline{PATH} .

 $NL \subseteq coNL$:

 $L \in \mathsf{NL} \Longrightarrow \overline{L} \in \mathsf{coNL}$



Definition: $coNL = \{L \mid \overline{L} \in NL\}$

Example: $\overline{PATH} \in coNL$.

Claim: $\overline{PATH} \in NL \implies NL = coNL$ **Proof:** $coNL \subseteq NL$:

Let $L \in coNL$. Then, $L \leq_1 \overline{PATH}$. ($\because \overline{PATH}$ is coNL-complete)

NL machine for L will first reduce L to \overline{PATH} and then use **NL** machine of \overline{PATH} .

 $NL \subseteq coNL$:

 $L \in \mathsf{NL} \Longrightarrow \overline{L} \in \mathsf{coNL} \Longrightarrow \overline{L} \in \mathsf{NL}$



Definition: $coNL = \{L \mid \overline{L} \in NL\}$

Example: $\overline{PATH} \in coNL$.

Claim: $\overline{PATH} \in NL \implies NL = coNL$ **Proof:** $coNL \subseteq NL$:

Let $L \in coNL$. Then, $L \leq_1 \overline{PATH}$. ($\because \overline{PATH}$ is coNL-complete)

NL machine for L will first reduce L to \overline{PATH} and then use **NL** machine of \overline{PATH} .

 $NL \subseteq coNL$:

 $L \in \mathsf{NL} \Longrightarrow \overline{L} \in \mathsf{coNL} \Longrightarrow \overline{L} \in \mathsf{NL} \Longrightarrow L \in \mathsf{coNL}$



Definition: $coNL = \{L \mid \overline{L} \in NL\}$

Example: $\overline{PATH} \in coNL$.

Claim: $\overline{PATH} \in NL \implies NL = coNL$ **Proof:** $coNL \subseteq NL$:

Let $L \in coNL$. Then, $L \leq_1 \overline{PATH}$. ($\because \overline{PATH}$ is coNL-complete)

NL machine for L will first reduce L to \overline{PATH} and then use **NL** machine of \overline{PATH} .

 $NL \subseteq coNL$:

 $L \in \mathsf{NL} \Longrightarrow \overline{L} \in \mathsf{coNL} \Longrightarrow \overline{L} \in \mathsf{NL} \Longrightarrow L \in \mathsf{coNL}$



Immerman-Szelepcsényi Theorem: $\overline{PATH} \in NL$.

Immerman-Szelepcsényi Theorem: $\overline{PATH} \in NL$.

Proof:

Immerman-Szelepcsényi Theorem: *PATH* ∈ NL. **Proof:** Idea: Make a read-once certificate for $\langle G, s, t \rangle \in \overline{PATH}$ that is verifiable in logspace.



Immerman-Szelepcsényi Theorem: $\overline{PATH} \in NL$.

- **Proof:** Idea: Make a read-once certificate for $\langle G, s, t \rangle \in \overline{PATH}$ that is verifiable in logspace.
 - Let $C_i = \text{Set of all the vertices of } G$ that have a path of length $\leq i$ from s.



Immerman-Szelepcsényi Theorem: $\overline{PATH} \in NL$.

Formally, we want a certificate for $t \notin C_n$.

- **Proof:** Idea: Make a read-once certificate for $\langle G, s, t \rangle \in \overline{PATH}$ that is verifiable in logspace.
 - Let $C_i = \text{Set of all the vertices of } G$ that have a path of length $\leq i$ from s.



Immerman-Szelepcsényi Theorem: $\overline{PATH} \in NL$.

Formally, we want a certificate for $t \notin C_n$.

Q. For a vertex v, how can someone prove that $v \in C_i$?

- **Proof:** Idea: Make a read-once certificate for $\langle G, s, t \rangle \in \overline{PATH}$ that is verifiable in logspace.
 - Let $C_i = \text{Set of all the vertices of } G$ that have a path of length $\leq i$ from s.



- Immerman-Szelepcsényi Theorem: $\overline{PATH} \in NL$.
- **Proof:** Idea: Make a read-once certificate for $\langle G, s, t \rangle \in \overline{PATH}$ that is verifiable in logspace.
 - Let $C_i = \text{Set of all the vertices of } G$ that have a path of length $\leq i$ from s. Formally, we want a certificate for $t \notin C_n$.

 - **Q.** For a vertex v, how can someone prove that $v \in C_i$?
 - **Proposed solution:** A sequence of vertices, $(v_1, v_2, ..., v_k)$.



Immerman-Szelepcsényi Theorem: $\overline{PATH} \in NL$.

Formally, we want a certificate for $t \notin C_n$.

Q. For a vertex v, how can someone prove that $v \in C_i$?

Proposed solution: A sequence of vertices, $(v_1, v_2, ..., v_k)$.

Verification: Check $v_1 = s, v_k = v$

- **Proof:** Idea: Make a read-once certificate for $\langle G, s, t \rangle \in \overline{PATH}$ that is verifiable in logspace.
 - Let $C_i = \text{Set of all the vertices of } G$ that have a path of length $\leq i$ from s.



Immerman-Szelepcsényi Theorem: $\overline{PATH} \in NL$.

Formally, we want a certificate for $t \notin C_n$.

Q. For a vertex v, how can someone prove that $v \in C_i$?

Proposed solution: A sequence of vertices, $(v_1, v_2, ..., v_k)$.

- **Proof:** Idea: Make a read-once certificate for $\langle G, s, t \rangle \in \overline{PATH}$ that is verifiable in logspace.
 - Let $C_i = \text{Set of all the vertices of } G$ that have a path of length $\leq i$ from s.
 - **Verification:** Check $v_1 = s$, $v_k = v$, $v_i v_{i+1}$ is an edge, and $k \le i + 1$.



Immerman-Szelepcsényi Theorem: $\overline{PATH} \in NL$.

Formally, we want a certificate for $t \notin C_n$.

Q. For a vertex v, how can someone prove that $v \in C_i$?

Proposed solution: A sequence of vertices, $(v_1, v_2, ..., v_k)$.

Verification: Check $v_1 = s$, $v_k = v$, $v_i v_{i+1}$ is an edge, and $k \le i+1$.

- **Proof:** Idea: Make a read-once certificate for $\langle G, s, t \rangle \in \overline{PATH}$ that is verifiable in logspace.
 - Let $C_i = \text{Set of all the vertices of } G$ that have a path of length $\leq i$ from s.

verification is doable in logspace, read-once manner.



- Immerman-Szelepcsényi Theorem: $\overline{PATH} \in NL$.
- **Proof:** Idea: Make a read-once certificate for $\langle G, s, t \rangle \in \overline{PATH}$ that is verifiable in logspace.
 - Let $C_i = \text{Set of all the vertices of } G$ that have a path of length $\leq i$ from s.
 - Formally, we want a certificate for $t \notin C_n$.
 - **Q.** For a vertex v, how can someone prove that $v \in C_i$?
 - **Proposed solution:** A sequence of vertices, $(v_1, v_2, ..., v_k)$. **doable in Logspace,** read-once manner. **Verification:** Check $v_1 = s$, $v_k = v$, $v_i v_{i+1}$ is an edge, and $k \le i+1$.
 - **Certificate:** A walk of length at most *i* from *s* to *v*, denoted $path_i(v)$.

verification is



- Immerman-Szelepcsényi Theorem: $\overline{PATH} \in NL$.
- **Proof:** Idea: Make a read-once certificate for $\langle G, s, t \rangle \in \overline{PATH}$ that is verifiable in logspace.
 - Let $C_i = \text{Set of all the vertices of } G$ that have a path of length $\leq i$ from s.
 - Formally, we want a certificate for $t \notin C_n$.
 - **Q.** For a vertex v, how can someone prove that $v \in C_i$?
 - **Proposed solution:** A sequence of vertices, $(v_1, v_2, ..., v_k)$. **doable in Logspace,** read-once manner. **Verification:** Check $v_1 = s$, $v_k = v$, $v_i v_{i+1}$ is an edge, and $k \le i+1$.
 - **Certificate:** A walk of length at most *i* from *s* to *v*, denoted $path_i(v)$.

verification is



• • •

Q. For a vertex v, how can someone prove that $v \notin C_i$

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_i|$?

Proof of NL = coNL ...

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_i|$? **Idea:** For every $u \in C_i$, prove that $u \in C_i$ and $u \neq v$.

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_i|$?

Idea: For every $u \in C_i$, prove that $u \in C_i$ and $u \neq v$.

Proposed Solution: v_1 : seq_1
Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_i|$?

Idea: For every $u \in C_i$, prove that $u \in C_i$ and $u \neq v$.

Proposed Solution: v_1 : seq_1 , v_2 : seq_2

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_i|$?

Idea: For every $u \in C_i$, prove that $u \in C_i$ and $u \neq v$.

Proposed Solution: v_1 : seq_1, v_2 : seq_2, \ldots, v_k : seq_k

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_i|$? **Idea:** For every $u \in C_i$, prove that $u \in C_i$ and $u \neq v$.

Proposed Solution: $v_1 : seq_1, v_2 : seq_2, ..., v_k : seq_k$, where $v_j \in V(G)$ and seq_j is a

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_i|$? **Idea:** For every $u \in C_i$, prove that $u \in C_i$ and $u \neq v$.

sequence of vertices.

Proposed Solution: v_1 : seq_1, v_2 : $seq_2, ..., v_k$: seq_k , where $v_j \in V(G)$ and seq_j is a

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_i|$? **Idea:** For every $u \in C_i$, prove that $u \in C_i$ and $u \neq v$.

sequence of vertices.

Verification: Check whether:

Proposed Solution: v_1 : seq_1, v_2 : $seq_2, ..., v_k$: seq_k , where $v_j \in V(G)$ and seq_j is a

- **Q.** For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_i|$? **Idea:** For every $u \in C_i$, prove that $u \in C_i$ and $u \neq v$.
- **Proposed Solution:** v_1 : seq₁, v_2 : seq₂, ..., v_k : seq_k, where $v_j \in V(G)$ and seq_j is a sequence of vertices.
- **Verification:** Check whether:
 - $\forall j, seq_i$ is a walk of length at most *i* from *s* to v_i .

- **Q.** For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_i|$? **Idea:** For every $u \in C_i$, prove that $u \in C_i$ and $u \neq v$.
- **Proposed Solution:** v_1 : seq_1, v_2 : $seq_2, ..., v_k$: seq_k , where $v_j \in V(G)$ and seq_j is a sequence of vertices.
- **Verification:** Check whether:
 - $\forall j, seq_i$ is a walk of length at most *i* from *s* to v_i . • $\forall j, v_j \neq v$.

- **Q.** For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_i|$? **Idea:** For every $u \in C_i$, prove that $u \in C_i$ and $u \neq v$.
- **Proposed Solution:** $v_1 : seq_1, v_2 : seq_2, ..., v_k : seq_k$, where $v_j \in V(G)$ and seq_j is a sequence of vertices.
- **Verification:** Check whether:
 - $\forall j, seq_i$ is a walk of length at most *i* from *s* to v_i . • $\forall j, v_j \neq v$. • $k = |C_i|$.

- **Q.** For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_i|$? **Idea:** For every $u \in C_i$, prove that $u \in C_i$ and $u \neq v$.
- **Proposed Solution:** $v_1 : seq_1, v_2 : seq_2, ..., v_k : seq_k$, where $v_i \in V(G)$ and seq_i is a sequence of vertices.
- **Verification:** Check whether:
 - $\forall j, seq_i$ is a walk of length at most *i* from *s* to v_i . • $\forall j, v_j \neq v$. • $k = |C_i|$.
 - $v_1 < v_2 < \ldots < v_k$.

- **Q.** For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_i|$? **Idea:** For every $u \in C_i$, prove that $u \in C_i$ and $u \neq v$.
- **Proposed Solution:** v_1 : seq₁, v_2 : seq₂, ..., v_k : seq_k, where $v_j \in V(G)$ and seq_j is a sequence of vertices.
- **Verification:** Check whether:
 - $\forall j, seq_i$ is a walk of length at most *i* from *s* to v_i . • $\forall j, v_j \neq v$. • $k = |C_i|$.
 - $v_1 < v_2 < \ldots < v_k$.



- **Q.** For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_i|$? **Idea:** For every $u \in C_i$, prove that $u \in C_i$ and $u \neq v$.
- **Proposed Solution:** v_1 : seq₁, v_2 : seq₂, ..., v_k : seq_k, where $v_j \in V(G)$ and seq_j is a sequence of vertices.
- **Verification:** Check whether:
 - $\forall j, seq_i$ is a walk of length at most *i* from *s* to v_i . • $\forall j, v_j \neq v$.
 - $k = |C_i|$.
 - $v_1 < v_2 < \ldots < v_k$.

Certificate: $v_1 : path_i(v_1), v_2 : path_i(v_2), ..., v_{|C_i|} : path_i(v_{|C_i|}), where v_i < v_{i+1}$.



Q. For a vertex v, how can someone prove that $v \notin C_i$

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$?

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$? **Idea:** For every $u \in C_{i-1}$, prove that $u \in C_{i-1}$ and $u \neq v$ and u has no edge to v.

Proof of NL = coNL ...

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$?

Idea: For every $u \in C_{i-1}$, prove that $u \in C_{i-1}$ and $u \neq v$ and u has no edge to v.

Proposed Solution: v_1 : seq_1

ve that $v \notin C_i$, if you know $|C_{i-1}|$? C_{i-1} and $u \neq v$ and u has no edge to v.

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$?

Proposed Solution: v_1 : seq_1 , v_2 : seq_2

Idea: For every $u \in C_{i-1}$, prove that $u \in C_{i-1}$ and $u \neq v$ and u has no edge to v.

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$? **Idea:** For every $u \in C_{i-1}$, prove that $u \in C_{i-1}$ and $u \neq v$ and u has no edge to v.

Proposed Solution: v_1 : seq_1, v_2 : seq_2, \ldots, v_k : seq_k

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$?

Idea: For every $u \in C_{i-1}$, prove that $u \in C_{i-1}$ and $u \neq v$ and u has no edge to v.

Proposed Solution: $v_1 : seq_1, v_2 : seq_2, ..., v_k : seq_k$, where $v_j \in V(G)$ and seq_j is a

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$? **Idea:** For every $u \in C_{i-1}$, prove that $u \in C_{i-1}$ and $u \neq v$ and u has no edge to v.

sequence of vertices.

Proposed Solution: v_1 : seq_1, v_2 : $seq_2, ..., v_k$: seq_k , where $v_j \in V(G)$ and seq_j is a

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$? **Idea:** For every $u \in C_{i-1}$, prove that $u \in C_{i-1}$ and $u \neq v$ and u has no edge to v.

Verification: Check whether:

Proposed Solution: v_1 : seq_1, v_2 : $seq_2, ..., v_k$: seq_k , where $v_j \in V(G)$ and seq_j is a sequence of vertices.

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$?

- **Idea:** For every $u \in C_{i-1}$, prove that $u \in C_{i-1}$ and $u \neq v$ and u has no edge to v.
- sequence of vertices.
- **Verification:** Check whether:
 - $\forall j, seq_i$ is a walk of length at most i 1 from s to v_i .

Proposed Solution: v_1 : seq_1, v_2 : $seq_2, ..., v_k$: seq_k , where $v_j \in V(G)$ and seq_j is a

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$?

sequence of vertices.

Verification: Check whether:

• $\forall j, seq_i$ is a walk of length at most i - 1 from s to v_i . • $\forall j, v_i \neq v$ and v_i has no edge to v.

Idea: For every $u \in C_{i-1}$, prove that $u \in C_{i-1}$ and $u \neq v$ and u has no edge to v. **Proposed Solution:** v_1 : seq₁, v_2 : seq₂, ..., v_k : seq_k, where $v_i \in V(G)$ and seq_i is a

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$? **Idea:** For every $u \in C_{i-1}$, prove that $u \in C_{i-1}$ and $u \neq v$ and u has no edge to v.

sequence of vertices.

Verification: Check whether:

- $\forall j, seq_i$ is a walk of length at most i 1 from s to v_i .
- $\forall j, v_i \neq v$ and v_i has no edge to v.
- $k = |C_{i-1}|$.

Proposed Solution: v_1 : seq₁, v_2 : seq₂, ..., v_k : seq_k, where $v_i \in V(G)$ and seq_i is a

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$? **Idea:** For every $u \in C_{i-1}$, prove that $u \in C_{i-1}$ and $u \neq v$ and u has no edge to v.

sequence of vertices.

Verification: Check whether:

- $\forall j, seq_i$ is a walk of length at most i 1 from s to v_i .
- $\forall j, v_i \neq v$ and v_i has no edge to v.
- $k = |C_{i-1}|$.
- $v_1 < v_2 < \ldots < v_k$.

Proposed Solution: v_1 : seq₁, v_2 : seq₂, ..., v_k : seq_k, where $v_i \in V(G)$ and seq_i is a

Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$? **Idea:** For every $u \in C_{i-1}$, prove that $u \in C_{i-1}$ and $u \neq v$ and u has no edge to v.

sequence of vertices.

Verification: Check whether:

- $\forall j, seq_i$ is a walk of length at most i 1 from s to v_i .
- $\forall j, v_j \neq v \text{ and } v_j \text{ has no edge to } v$.
- $k = |C_{i-1}|$.
- $v_1 < v_2 < \ldots < v_k$.

Proposed Solution: v_1 : seq_1, v_2 : $seq_2, ..., v_k$: seq_k , where $v_i \in V(G)$ and seq_i is a



Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$? **Idea:** For every $u \in C_{i-1}$, prove that $u \in C_{i-1}$ and $u \neq v$ and u has no edge to v.

sequence of vertices.

Verification: Check whether:

- $\forall j, seq_i$ is a walk of length at most i 1 from s to v_i .
- $\forall j, v_j \neq v \text{ and } v_j \text{ has no edge to } v$.
- $k = |C_{i-1}|$.
- $v_1 < v_2 < \ldots < v_k$.

Certificate: $nopath_i(v)$

Proposed Solution: v_1 : seq_1, v_2 : $seq_2, ..., v_k$: seq_k , where $v_i \in V(G)$ and seq_i is a



Q. For a vertex v, how can someone prove that $v \notin C_i$, if you know $|C_{i-1}|$? **Idea:** For every $u \in C_{i-1}$, prove that $u \in C_{i-1}$ and $u \neq v$ and u has no edge to v.

sequence of vertices.

Verification: Check whether:

- $\forall j, seq_i$ is a walk of length at most i 1 from s to v_i .
- $\forall j, v_j \neq v \text{ and } v_j \text{ has no edge to } v$.
- $k = |C_{i-1}|$.

• $v_1 < v_2 < \ldots < v_k$.

Proposed Solution: v_1 : seq_1, v_2 : $seq_2, ..., v_k$: seq_k , where $v_i \in V(G)$ and seq_i is a

verification is doable in logspace, read-once manner.

Certificate: $nopath_i(v) = v_1 : path_{i-1}(v_1), ..., v_{|C_{i-1}|} : path_{i-1}(v_{|C_{i-1}|}), where v_i < v_{i+1}$.



Q. How can someone prove that $|C_i| = c$

Q. How can someone prove that $|C_i| = c$, if you know $|C_{i-1}|$?

Q. How can someone prove that $|C_i| = c$, if you know $|C_{i-1}|$? **Idea:** For every $u \in V(G)$, prove that $u \in C_i$ or $u \notin C_i$ (whichever is true).

Q. How can someone prove that $|C_i| = c$, if you know $|C_{i-1}|$?

Idea: For every $u \in V(G)$, prove that $u \in C_i$ or $u \notin C_i$ (whichever is true).

Proposed Solution: v_1 : seq_1

Proof of NL = coNL ...

Q. How can someone prove that $|C_i| = c$, if you know $|C_{i-1}|$? **Idea:** For every $u \in V(G)$, prove that $u \in C_i$ or $u \notin C_i$ (whichever is true).

Proposed Solution: v_1 : seq_1 , v_2 : seq_2

Q. How can someone prove that $|C_i| = c$, if you know $|C_{i-1}|$? **Idea:** For every $u \in V(G)$, prove that $u \in C_i$ or $u \notin C_i$ (whichever is true).

Proposed Solution: v_1 : seq_1, v_2 : seq_2, \ldots, v_k : seq_k

Q. How can someone prove that $|C_i| = c$, if you know $|C_{i-1}|$? **Idea:** For every $u \in V(G)$, prove that $u \in C_i$ or $u \notin C_i$ (whichever is true).

- **Proposed Solution:** $v_1 : seq_1, v_2 : seq_2, ..., v_k : seq_k$, where $v_j \in V(G)$ and seq_j is a
Q. How can someone prove that $|C_i| = c$, if you know $|C_{i-1}|$? **Idea:** For every $u \in V(G)$, prove that $u \in C_i$ or $u \notin C_i$ (whichever is true). **Proposed Solution:** v_1 : seq_1, v_2 : $seq_2, ..., v_k$: seq_k , where $v_j \in V(G)$ and seq_j is a sequence of vertices.

Q. How can someone prove that $|C_i| = c$, if you know $|C_{i-1}|$? **Idea:** For every $u \in V(G)$, prove that $u \in C_i$ or $u \notin C_i$ (whichever is true). **Proposed Solution:** v_1 : seq_1, v_2 : $seq_2, ..., v_k$: seq_k , where $v_j \in V(G)$ and seq_j is a sequence of vertices.

Q. How can someone prove that $|C_i| = c$, if you know $|C_{i-1}|$? **Idea:** For every $u \in V(G)$, prove that $u \in C_i$ or $u \notin C_i$ (whichever is true).

sequence of vertices.

Verification: Check whether:

• $\forall j, seq_i \text{ is either } path_i(v_i) \text{ or } nopath_i(v_i).$

- **Proposed Solution:** v_1 : seq₁, v_2 : seq₂, ..., v_k : seq_k, where $v_i \in V(G)$ and seq_i is a

Q. How can someone prove that $|C_i| = c$, if you know $|C_{i-1}|$? **Idea:** For every $u \in V(G)$, prove that $u \in C_i$ or $u \notin C_i$ (whichever is true).

sequence of vertices.

- $\forall j, seq_i$ is either $path_i(v_i)$ or $nopath_i(v_j)$.
- # of v_i s for which $seq_i = path_i(v_i)$ is c.

- **Proposed Solution:** v_1 : seq₁, v_2 : seq₂, ..., v_k : seq_k, where $v_i \in V(G)$ and seq_i is a

Q. How can someone prove that $|C_i| = c$, if you know $|C_{i-1}|$? **Idea:** For every $u \in V(G)$, prove that $u \in C_i$ or $u \notin C_i$ (whichever is true).

sequence of vertices.

- $\forall j, seq_i \text{ is either } path_i(v_i) \text{ or } nopath_i(v_i).$
- # of v_i s for which $seq_i = path_i(v_i)$ is c.
- k = n.

- **Proposed Solution:** v_1 : seq₁, v_2 : seq₂, ..., v_k : seq_k, where $v_i \in V(G)$ and seq_i is a

Q. How can someone prove that $|C_i| = c$, if you know $|C_{i-1}|$? **Idea:** For every $u \in V(G)$, prove that $u \in C_i$ or $u \notin C_i$ (whichever is true). **Proposed Solution:** v_1 : seq_1, v_2 : $seq_2, ..., v_k$: seq_k , where $v_i \in V(G)$ and seq_i is a

sequence of vertices.

- $\forall j, seq_i$ is either $path_i(v_i)$ or $nopath_i(v_j)$.
- # of v_i s for which $seq_i = path_i(v_i)$ is c.
- k = n.
- $v_1 < v_2 < \ldots < v_k$.

Q. How can someone prove that $|C_i| = c$, if you know $|C_{i-1}|$? **Idea:** For every $u \in V(G)$, prove that $u \in C_i$ or $u \notin C_i$ (whichever is true).

sequence of vertices.

Verification: Check whether:

- $\forall j, seq_i \text{ is either } path_i(v_i) \text{ or } nopath_i(v_j).$
- # of v_i s for which $seq_i = path_i(v_i)$ is c.
- k = n.
- $v_1 < v_2 < \ldots < v_k$.

- **Proposed Solution:** v_1 : seq₁, v_2 : seq₂, ..., v_k : seq_k, where $v_i \in V(G)$ and seq_i is a



verification is doable in logspace, read-once manner.

Q. How can someone prove that $|C_i| = c$, if you know $|C_{i-1}|$? **Idea:** For every $u \in V(G)$, prove that $u \in C_i$ or $u \notin C_i$ (whichever is true).

sequence of vertices.

Verification: Check whether:

- $\forall j, seq_j \text{ is either } path_i(v_j) \text{ or } nopath_i(v_j).$
- # of v_i s for which $seq_i = path_i(v_i)$ is c.
- k = n.

•
$$v_1 < v_2 < \ldots < v_k$$
.

Certificate: $size_i(c) = \dots$

- **Proposed Solution:** v_1 : seq₁, v_2 : seq₂, ..., v_k : seq_k, where $v_i \in V(G)$ and seq_i is a



verification is doable in logspace, read-once manner.

There exists a read-once logspace verifiable certificate that:

There exists a read-once logspace verifiable certificate that:

• Proves $v \in C_i$.

There exists a read-once logspace verifiable certificate that:

- Proves $v \in C_i$.
- Proves $v \notin C_i$, if $|C_{i-1}|$ is already known. (certificate is $nopath_i(v)$)

There exists a read-once logspace verifiable certificate that:

- Proves $v \in C_i$.
- Proves $v \notin C_i$, if $|C_{i-1}|$ is already known. (certificate is $nopath_i(v)$)
- Proves $|C_i| = c$, if $|C_{i-1}|$ is already known. (certificate is $size_i(c)$)

There exists a read-once logspace verifiable certificate that:

- Proves $v \in C_i$.
- Proves $v \notin C_i$, if $|C_{i-1}|$ is already known. (certificate is $nopath_i(v)$)
- Proves $|C_i| = c$, if $|C_{i-1}|$ is already known. (certificate is $size_i(c)$)

We know $C_0 = \{s\}, |C_0| = 1.$

There exists a read-once logspace verifiable certificate that:

- Proves $v \in C_i$.
- Proves $v \notin C_i$, if $|C_{i-1}|$ is already known. (certificate is $nopath_i(v)$)
- Proves $|C_i| = c$, if $|C_{i-1}|$ is already known. (certificate is $size_i(c)$)
- We know $C_0 = \{s\}, |C_0| = 1.$
- Final certificate for **PATH**:

There exists a read-once logspace verifiable certificate that:

- Proves $v \in C_i$.
- Proves $v \notin C_i$, if $|C_{i-1}|$ is already known. (certificate is $nopath_i(v)$)
- Proves $|C_i| = c$, if $|C_{i-1}|$ is already known. (certificate is $size_i(c)$)
- We know $C_0 = \{s\}, |C_0| = 1.$

Final certificate for **PATH**:

 $size_1(|C_1|)$

There exists a read-once logspace verifiable certificate that:

- Proves $v \in C_i$.
- Proves $v \notin C_i$, if $|C_{i-1}|$ is already known. (certificate is $nopath_i(v)$)
- Proves $|C_i| = c$, if $|C_{i-1}|$ is already known. (certificate is $size_i(c)$)
- We know $C_0 = \{s\}, |C_0| = 1.$
- Final certificate for **PATH**:

 $size_1(|C_1|)$

vn. (certificate is $nopath_i(v)$) nown. (certificate is $size_i(c)$)

Certificate that proves the size of C_1 knowing $|C_0|$.

There exists a read-once logspace verifiable certificate that:

- Proves $v \in C_i$.
- Proves $v \notin C_i$, if $|C_{i-1}|$ is already known. (certificate is $nopath_i(v)$)
- Proves $|C_i| = c$, if $|C_{i-1}|$ is already known. (certificate is $size_i(c)$)
- We know $C_0 = \{s\}, |C_0| = 1.$
- Final certificate for **PATH**:

 $size_1(|C_1|), size_2(|C_2|)$

vn. (certificate is $nopath_i(v)$) nown. (certificate is $size_i(c)$)

Certificate that proves the size of C_1 knowing $|C_0|$.

There exists a read-once logspace verifiable certificate that:

- Proves $v \in C_i$.
- Proves $v \notin C_i$, if $|C_{i-1}|$ is already known. (certificate is *nopath*_i(v))
- Proves $|C_i| = c$, if $|C_{i-1}|$ is already known. (certificate is $size_i(c)$)
- We know $C_0 = \{s\}, |C_0| = 1.$
- Final certificate for PATH:

 $size_1(|C_1|), size_2(|C_2|), \dots, size_{n-1}(|C_{n-1}|)$

. Certificate that proves the size of C_1 knowing $|C_0|$.

There exists a read-once logspace verifiable certificate that:

- Proves $v \in C_i$.
- Proves $v \notin C_i$, if $|C_{i-1}|$ is already known. (certificate is *nopath*_i(v))
- Proves $|C_i| = c$, if $|C_{i-1}|$ is already known. (certificate is $size_i(c)$)
- We know $C_0 = \{s\}, |C_0| = 1.$
- Final certificate for PATH:

. Certificate that proves the size of C_1 knowing $|C_0|$.

 $size_1(|C_1|), size_2(|C_2|), \dots, size_{n-1}(|C_{n-1}|), nopath_n(t)$

There exists a read-once logspace verifiable certificate that:

- Proves $v \in C_i$.
- Proves $v \notin C_i$, if $|C_{i-1}|$ is already known. (certificate is *nopath*_i(v))
- Proves $|C_i| = c$, if $|C_{i-1}|$ is already known. (certificate is $size_i(c)$)
- We know $C_0 = \{s\}, |C_0| = 1.$
- Final certificate for PATH:
- **Corollary:** For every space-constructible function $S(n) \ge \log n$,

. Certificate that proves the size of C_1 knowing $|C_0|$.

 $size_1(|C_1|), size_2(|C_2|), \dots, size_{n-1}(|C_{n-1}|), nopath_n(t)$

NSPACE(S(n)) = coNSPACE(S(n)).