Consider $L$ according to new definition. Construct a new machine $M'$ that simply makes a non-deterministic guess about the next symbol in $y$. Thus, it can simulate the presence of $y$ without any storage with at most constant storage.

The Immerman–Szelepsényi Theorem

$\text{NL} = \text{co-NL}$ Complementation does not change non-deterministic space class!

$\text{NSPACE}(s(n)) = \text{co-NSPACE}(s(n))$ (by padding)

$\text{PATH}$ is $\text{NL}$-complete. So $\text{PATH}$ is $\text{co-NL}$-complete.

If we show $\text{PATH} \in \text{NL}$, then

$\forall A \in \text{co-NL}, A \leq_{\text{P}} \text{PATH}$, so $A \in \text{NL}$

Thus $\text{co-NL} \subseteq \text{NL} \subseteq \text{co-NL}$

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

$\text{PATH} \in \text{NL}$ is obvious, but how can we generate read-once certificate for $\text{PATH}$?

(For $\langle G, s, t \rangle \in \text{PATH}$, read-once certificate is the path from $s$ to $t$.)
Thus: \( \text{PATH} \in \text{NL} \)

**Pf:** Think in terms of certificates. You'll go nuts thinking about a non-deterministic logspace machine.

Define \( \text{PATH}_i = \{ \langle G, s, t \rangle \mid \text{There is no path of length } \leq i \text{ from } s \text{ to } t \} \)

\( \text{PATH}_0 \) is trivial. Why?

\( \text{PATH}_1 \) is in NL. Why?

Consider \( \text{PATH}_2 \). How to certify that \( d(s, t) > 2 \)? (Define \( d(s, v) \) to be shortest path between \( s \) and \( v \).)

If we show: \( \Gamma^-(t) \cap \Gamma^+(s) = \emptyset \), we're done.

First try: Certificate is \( \Gamma^+(s) : v_i, v_{i-1}, \ldots, v_1 \)

1. For \( v \) in \( v_i, v_{i-1}, \ldots, v_1 \) (in order)
   a. Check if \( v \) is in \( \Gamma^+(s) \). (logspace)
   b. Check if \( v \) is in \( \Gamma^-(t) \). (logspace)
   c. If yes, reject
   d. If not, reject

But there could be some \( v \in \Gamma^+(s) \) that is NOT in our certificate.

So, we first compute \( d^+ \) and make sure \( d = d^+ \).

How do we know some \( v \) isn't repeated?

Force \( v_i \)'s to be in ascending order of id.
Thus, for any \( v \), we have certificate for \( d(s,v) > 2 \). There is obvious certificate for \( d(s,v) \leq 2 \).

Now for \( \text{PATH}_2 \). We need to show \( \Gamma(v) \cap B^+(s,2) = \emptyset \).

Certificate is \( B^+(s,2) \) in order. But, after each \( v \in B^+(s,2) \), we need certificate that \( v \notin B^+(s,2) \). Easy, just take the path from \( s \) to \( v \) of length 2.

\[
\begin{aligned}
v_1 &\rightarrow s \rightarrow v_2, \\
v_2 &\rightarrow s \rightarrow v_3, \\
v_3 &\rightarrow s \rightarrow v_4, \\
&\ldots
\end{aligned}
\]

\[\text{init} \]
1. On seeing \( v_i \), store it.
2. Verify \( s \rightarrow v_i \) is correct. (If not, reject.)
3. Check if \( t \) is a neighbor of \( v_i \). If so, reject.

But we also need the sign of \( B^+(s,2) \) (call it \( C_2 \)).

This can be part of the certificate, but how can it be verified?

This is by using the \( d(s,v) > 2 \) certificates!

\[
C_2 \quad s \rightarrow 1 \rightarrow s \rightarrow 2 \quad s \rightarrow 3 \rightarrow s \rightarrow 4 \quad \ldots
\]

Verify that \( C_2 \) is correct. Logspace machine can forget everything.

Now the list of \( B^+(s,2) \) is repeated \( n \) times to certify each \( v \in B^+(s,2) \) or \( v \notin B^+(s,2) \).

\[
\begin{aligned}
v_1 &\rightarrow v_4 \leftarrow v_1, \\
v_1 &\rightarrow v_4 \leftarrow v_1, \\
&\ldots
\end{aligned}
\]

\[n \text{ times}
\]

\[
C_3
\]
Thus, length of certificate for $|G_i|$, denote $L_i$

$$L_i \leq L_{i-1} + O(n^3 \log n) \quad L_i = O(n^4 \log n)$$

For $v_i, v_j, \ldots, v_k$ in $B^+(s, i-1)$

\[ \begin{array}{c|c|c|c}
 v_1 & \ldots & v_i & \ldots & v_k \\
 \hline
 s & \sim & v_i & \sim & s & \sim & v_k \\
 \end{array} \]

Repeat $n$ times