L = SPACE(log n) \quad NL = NSPACE(log n)

L \subseteq NL \subseteq P \subseteq NP

**Def.** \( f \) is logspace-computable \iff \( f \) is computable by a TM of the following form:

![Diagram of a Turing Machine](attachment:image.png)

**Def.** \( A \leq_L^m B \) \iff \( A \leq_m^L B \) via reduction, \( f \) that is logspace-computable

**Def.** \( B \) is \( \text{NL-hard} \) \iff \( \forall A \in \text{NL}, \ A \leq_L^m B \)

**Def.** \( B \) is \( \text{NL-complete} \) \iff (i) \( B \in \text{NL} \) (ii) \( B \) is NL-complete

**Def.** \( \text{PATH} \) is NL-complete

**Proof.** (1) \( \text{PATH} \in \text{NL} \) \iff \( \text{last min} \)

(2) Let \( A \in \text{NL} \), claim \( A \leq_L^m \text{PATH} \)

Reduction from last min:

\[ A \quad \xrightarrow{f} \quad \langle G, x, \text{concept} \rangle \] where \( G \) is logspace NTH deciding \( A \)
Why is \( f \) logspace-computable?
- Each configuration vector of \( G_{n,x} \)
  takes \( O(\log n) \) space so \( O(\log n) \) easy

Producing \( G_{n,x} \):
  Adjacency list forms
  For all configurations \( C \)
    (in lexicographic order, not necessarily reachable)
  Output \( C \) followed by all
  next configurations \( D_i \)
    s.t. \( C \rightarrow M D_i \)
      (i.e. \( C \rightarrow \{ D \}) \)
    i.e. \( C : D_{i_1}, \ldots, D_{i_j} \)
    \( \text{in} \) \( \text{out-} \) neighbors
    only need to store a constant #
      of configurations.
    \( \therefore \) \( O(\log n) \) space

We still need to prove properties of \( \leq L \) that were easy
for \( \leq M \) and \( \leq W \) but are tricky for \( \leq L \).

Then:
- If \( A \leq M B \) and \( B \in L \) then \( A \in L \)
- If \( A \leq W B \) and \( B \in NL \) then \( A \in NL \)
- If \( A \leq M B \) and \( B \leq \leq C \) then \( A \leq \leq C \)

**Proof**
- Usual method

Instead:
Modify $M_B$: If $M_B$ is looking at $y_i$, we have $M_B$ also keep track of the input head position.

Change $M_f$ by reversing its output tape. New machine for $A$ will "call" $M_f$ with index $i$. ($r$ is still on input tape, $i$ is on the control tape.) Each time it does it will run $M_f$ ignoring its output except for the $i$th bit of output. $M_f$ will need to keep track of the # of bits output so far, $j$.

Re-run $M_f$ each time step of $M_B$ to find out the value of $y_i$.

Total space: Space for $M_f$
Space for $M_B$ $+ O(\log n)$

Note: $|f|_{1}^{1}$ is $\log |f|$ if $n = 1$

$|\log |f||_{1}^{1}$ is $O(\log n)$ so still $O(\log n)$ space total

Note: same construction works for $NL$ case.

For $A \leq^L B$ and $B \leq^L C$ $\implies$ $A \leq^L C$
do the same except $M_B$ replaced by $M_f$
do some change as above $\frac{1}{f(n)} = \frac{1}{(4n^3)}$
Con. $\text{PATH} \leq_{m} C \Rightarrow C$ is NL-hard

The following is very surprising.

Then $\text{PATH} \notin \text{NL}$

$\text{PATH} \equiv \{ \langle G, s, t \rangle : G \text{ does not have a path from } s \text{ to } t \}$

Con. $\text{NL} = \text{coNL}$

- complements of languages in NL

Con. For any space bound $S(n) \geq \log_2 n$

$\text{NSPACE}(S(n))$ is closed under complement

Proof. Imagine that we have the value

$\text{Count} = \# \text{ of vertices of } G \text{ reachable from } s$

No Path $(s, t, \text{Count}, i)$

- Reach $\leq O$

For all vertices $v \in G$

Guess whether $v$ is reachable from $s$

if Guess is yes then

Guess & verify a path of length $\leq \text{Count}$ from $s$ to $v$, one vertex at a time

if path found Reach = $\text{Reach} + 1$

else reject

end for

if reach = count then accept else reject
How do we compute $\text{Count}_i$?

Idea: "inductive counting"

Define: $\text{Count}_i = \#$ of vertices reachable from $s$ via paths of length $\leq i$

$\therefore \text{Count}_0 = 1 \quad \& \leq 3$

$\text{count} = \text{Count}_n$

This will be via a nondeterministic algorithm, such an alg with have some paths that reject but any branch that does not reject will compute the correct value.

We can't afford to store all the $\text{Count}_i$ vars but we only need vars for the current layer $i$, $\text{count}_i$, $\text{count}_{i+1}$

$i\leq 0$, $\text{count}_{i} \leftarrow 1$

for $i = 0$ to $n-1$ do

$\text{count}_{i+1} \leftarrow 0$

for all vertices $v \in G$ do

if $v = s$ then

$\text{count}_{i+1} \leftarrow \text{count}_{i+1} + 1$

else

Guess whether $v$ is reachable from $s$ via a path of length $\leq i+1$

if guess for $v$ is yet

Guess & verify a path of length $\leq i+1$

drawn from $s$ to $v$, one vertex at a time

if found then $\text{count}_{i+1} \leftarrow \text{count}_{i+1} + 1$

else reject
If guess for $v$ is no
    for all predecessors $u$ of $v$ in $G$
      "Check that no path of length $e_i$
       from $s$ to $u$ in $G$"
      if No Path$(s, t, \text{count}_{t}, i)$ is false
        then reject
    end for
  end for
Count $\leq \text{count} + 1$

Clearly only a constant # of counters and variables need to be stored. $\mathcal{O}(\log n)$ space.