NL- Completeness

$L \subseteq NL \subseteq P \subseteq NP$

Consider the language PATH.

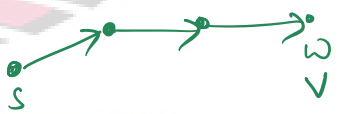
$PATH = \{ \langle G, s, t \rangle \mid G$ is a directed graph and has an s-t directed path $\}$

Example 8.19: $PATH \in NL$.

We know $PATH \in P$. But traversal algorithms like BFS / DFS require linear space. Instead, we can do the "guess & verify" approach.

Input $G = (V, E)$, $|V| = n$.

Set $v = s$. Count = 0.



For Count = 0 to n

Guess $w \in V$

If $v = t$ accept.

If $v \to w$ (that is, (v, w) is an edge), assign value of w to v.

Else, (v, w) not an edge), reject

Count = Count + 1.

Reject if not accepted.

The above algorithm needs to store v, w, count. Each requires $\log n$ space.

Hence $PATH \in NL$.

LOGSPACE REDUCTIONS

$A \leq_L B \Rightarrow A \leq_p B$

$A \leq_p B \not\Rightarrow A \leq_L B$

Def 8.21: We say $A \leq_L B$, A reduces to B in logspace, if there is a deterministic logspace machine M that computes f such that

$$w \in A \iff f(w) \in B.$$

Def 8.22 (NL-Completeness): A language B is NL-Complete if

(1) $B \in NL$, and

(2) $\forall A \in NL$, $A \leq_L B$.

→ Would it make sense if we used polytime reduction instead of logspace reduction?

No! We saw $NL \subseteq P$. So polytime can be used to decide A.

→ For polytime reductions, we had

$$A \leq_p B, \quad B \in P \implies A \in P.$$

$A \to B$ n^2 time

B decides $|f(w)|^3$ time

$\Rightarrow$ A decided in n^6 time

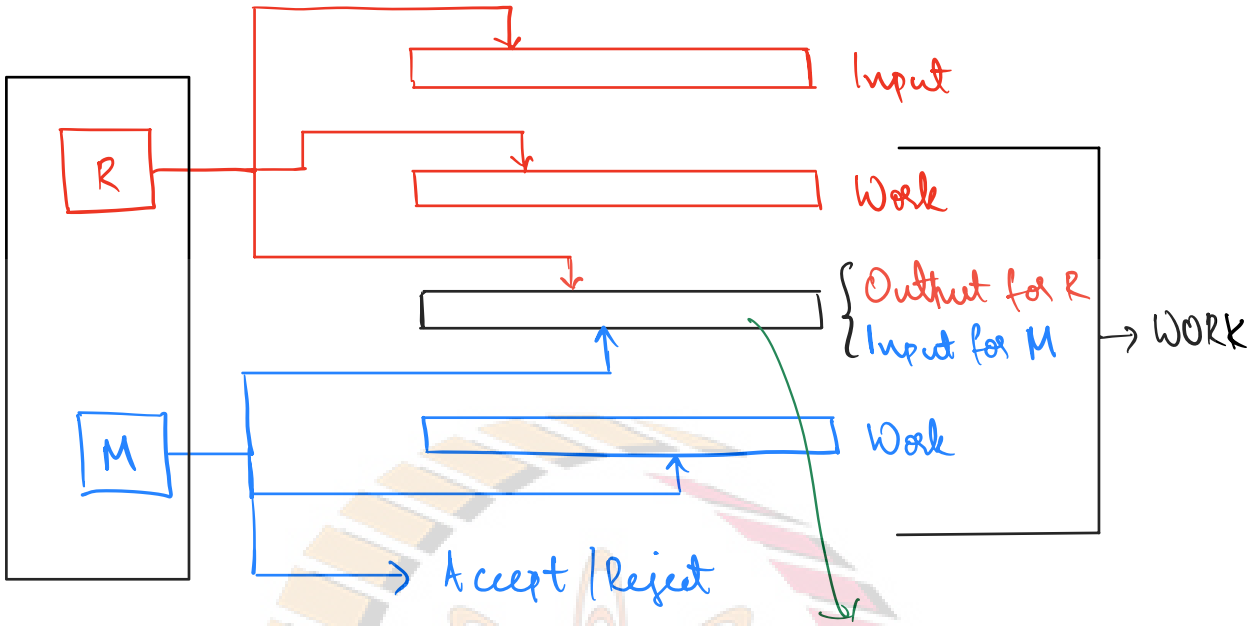
Given $A \leq_L B$ and $B \in L$, can we infer that $A \in L$?

First attempt, given w, we compute $f(w)$ where f is the reduction from A to B. But what is the upper bound on $|f(w)|$?

This could be n^c, where $n = |w|$.

Let R be the machine that computes the reduction from A to B. Let M be the logspace decider for B.

If we try to make a decider for A by combining R and M, we get three work tapes. The space used by $f(w)$ in the output tape of R could be as much as $2^{O(\log n)} \approx \text{poly}(n)$. Hence the combined machine is not a logspace decider for A.



May take $\leq n^c$ space

Work tape of M requires $\log(n^c) = O(\log n)$ space

So we do not have space to write f(w) down.

We do the following:

* Start M, the decider for B. Suppose the input (A-instance) is x.
* When M needs the j^{th} symbol of f(x),
  - Remember j
  - Run R till the j^{th} symbol of f(x) gets output
  - Use j^{th} symbol

This is repeated whenever we need a symbol from $f(x)$. We may end up recomputing many symbols of $f(x)$ but that does not cost us in space. Thus we can conclude that

Theorem: $A \leq_L B$ and $B \in L \Rightarrow A \in L$

Def 8.22 (NL-Completeness): A language B is NL-Complete if

(1) $B \in NL$, and

(2) $\forall A \in NL$, $A \leq_L B$.

Hardest problems in NL!

Theorem: PATH is NL-complete.

Earlier, we saw that $PATH \in NL$. We now need to show that $\forall A \in NL$, $A \leq_L PATH$.

Proof idea: Construct the configuration graph of the NL decider for A and check if there is a path from C_{start} to C_{accept}.

Exercise: Show that we can construct an equivalent NTM with a single accepting configuration.

No. of vertices in the config graph

$= $ No. of configurations

$\leq \underbrace{n * \log n}_{\text{Head positions}} * \underbrace{|\Gamma|^{\log n}}_{\text{Tape Contents}} * \underbrace{|Q|}_{\text{States}}$

$\leq 2^{O(\log n)} = n^{O(1)}$

Each configuration has a label of size $O(\log n)$.

Now we need to explain the construction of the PATH instance such that

$$x \in A \iff \langle G, s, t \rangle \in PATH.$$

The reduction machine does the following:

(1) list down all the configurations of the NL decider of A.

- Needs an $O(\log n)$ counter.
- Check each string if it is a legal config.

(2) list down all the edges.

- Go over all the config (Counter 1)
- list all possible successors (Counter 2)
- For each possible successor, check if it is a legal successor.

(3) G is defined. $s = c_{start}$, $t = c_{accept}$.

The space requirement is $O(\log n)$. But the output size is poly(n).

(Read the detailed proof in Sipser)

Exercise: Show that A_{NFA} is NL-complete.

$A_{NFA} = \{\langle N, w\rangle \mid N \text{ is an NFA that accepts } w\}$

Hint: We can show $PATH \leq_L A_{NFA}$ instead of showing $\forall A \in NL,\ A \leq A_{NFA}$. The idea is to recreate the graph as an NFA.

Theorem: If $C \in NL$, and B is NL-complete and $B \leq_L C$, then C is also NL-complete.

Theorem: $A \leq_L B$ and $B \leq_L C \Rightarrow A \leq_L C$.