18.404/6.840 Lecture 20

shrink me →

Last time:

- Games and Quantifiers
- Generalized Geography is PSPACE-complete
- Logspace: L and NL

Today:

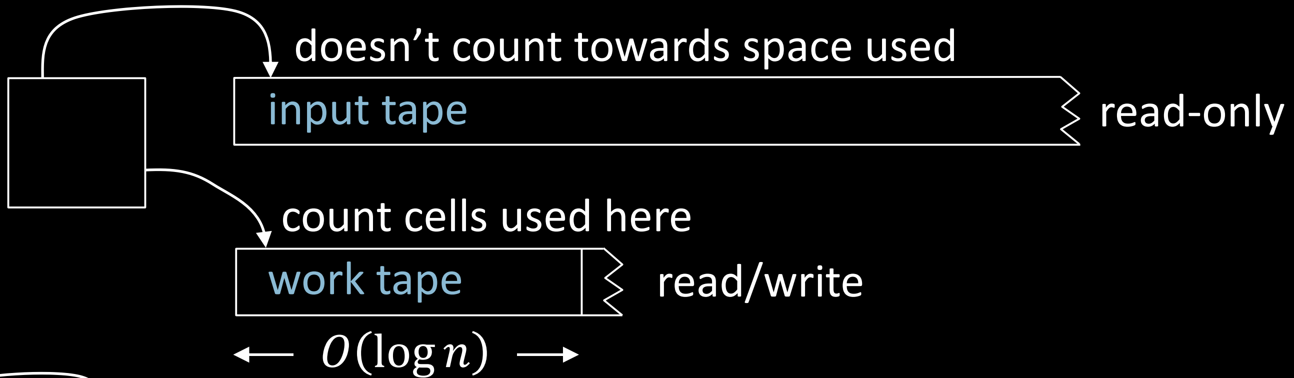
- Review NL $\subseteq$ P
- Review NL $\subseteq$ SPACE$(\log^2 n)$
- NL-completeness
- NL = coNL

Review: log space

Model: 2-tape TM with read-only input tape for defining sublinear space computation.

Defn: L = SPACE$(\log n)$
NL = NSPACE$(\log n)$

Log space can represent a constant number of pointers into the input.

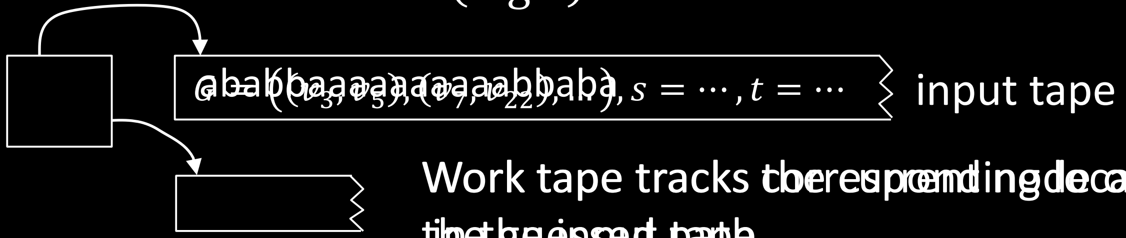


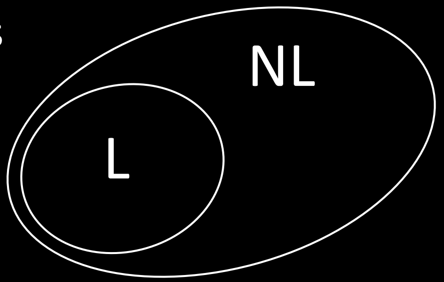
Examples

1. $\{ww^{\mathcal{R}} \mid w \in \Sigma^*\} \in$ L

2. $PATH \in$ NL

Nondeterministically select the nodes of a path connecting s to t.





L = NL? Unsolved

Review: L ⊆ P

Theorem: L ⊆ P

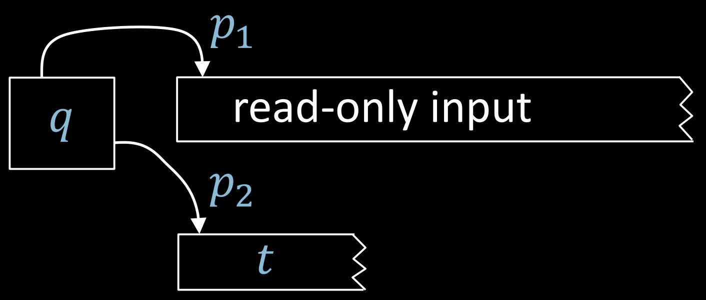
Proof: Say M decides A in space $O(\log n)$.

Defn: A configuration for M on w is (q, p_1, p_2, t) where q is a state, p_1 and p_2 are the tape head positions, and t is the work tape contents.

The number of such configurations is $|Q| \times n \times O(\log n) \times d^{O(\log n)} = O(n^k)$ for some k.

Therefore M runs in polynomial time.

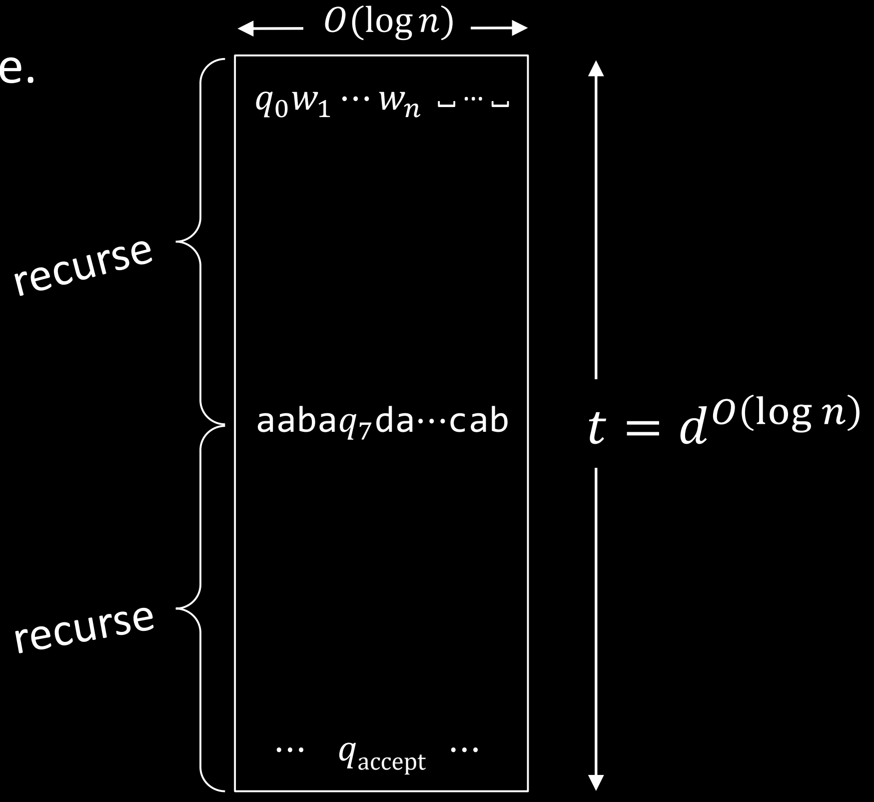
Conclusion: $A \in$ P



Review: NL $\subseteq$ SPACE$(\log^2 n)$

Theorem: NL $\subseteq$ SPACE$(\log^2 n)$
Proof: Savitch's theorem works for log space

Each recursion level stores 1 config = $O(\log n)$ space.
Number of levels = $\log t = O(\log n)$.
Total $O(\log^2 n)$ space.



Review: NL ⊆ P

Theorem: NL ⊆ P

Proof: Say NTM M decides A in space $O(\log n)$.

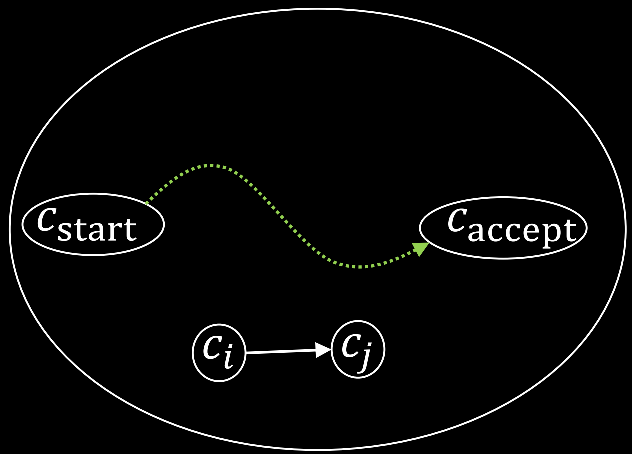
Defn: The configuration graph $G_{M,w}$ for M on w has
- **nodes:** all configurations for M on w
- **edges:** edge from $c_i \to c_j$ if c_i can yield c_j in 1 step.

Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

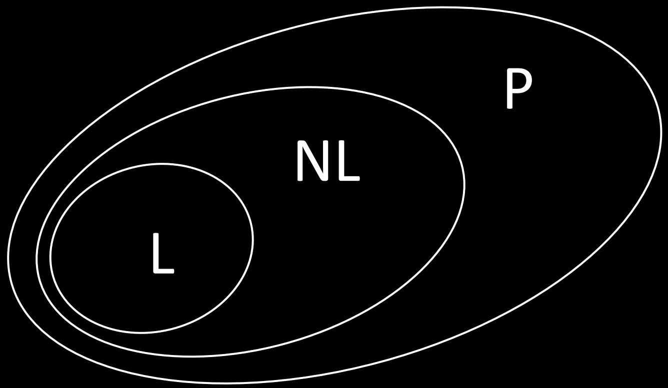
Polynomial time algorithm T for A:

$T =$ "On input w
1. Construct $G_{M,w}$. [polynomial size]
2. *Accept* if there is a path from c_{start} to c_{accept}. *Reject* if not."

configuration graph $G_{M,w}$



iff M accepts w



L = P? Unsolved

NL-completeness

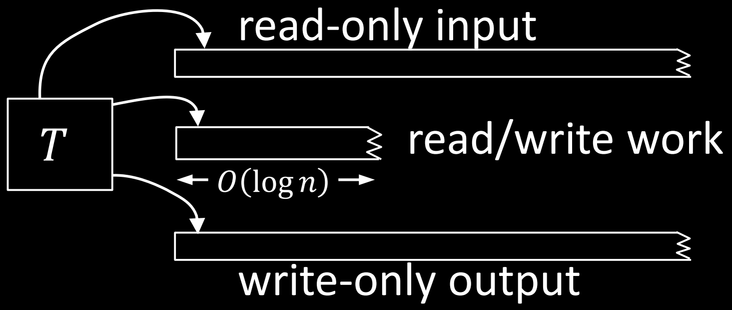
Check-in 20.1

If T is a log-space transducer that computes f, then for inputs w of length n, how long can $f(w)$ be?

(a) at most $O(\log n)$
(b) at most $O(n)$
(c) at most polynomial in n
(d) at most $2^{O(n)}$
(e) any length

Defn: A <u>log-space transducer</u> is a TM with three tapes:

1. read-only input tape of size n
2. read/write work tape of size $O(\log n)$
3. write-only output tape



A log-space transducer T computes a function $f : \Sigma^* \to \Sigma^*$ if T on input w halts with $f(w)$ on its output tape for all w.
Say that f is computable in log-space.

Defn: A is <u>log-space reducible</u> to B $(A \leq_{\mathrm{L}} B)$ if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in log-space.

Theorem: If $A \leq_{\mathrm{L}} B$ and $B \in \mathrm{L}$ then $A \in \mathrm{L}$

Proof: TM for $A =$ "On input w
1. Compute $f(w)$
2. Run decider for B on $f(w)$. Output same."

BUT we don't have space to store $f(w)$.
So, (re-)compute symbols of $f(w)$ as needed.

Check-in 20.1

$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_{\mathrm{L}} PATH$

Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

[Modify M to erase work tape and move heads to left end upon accepting.]

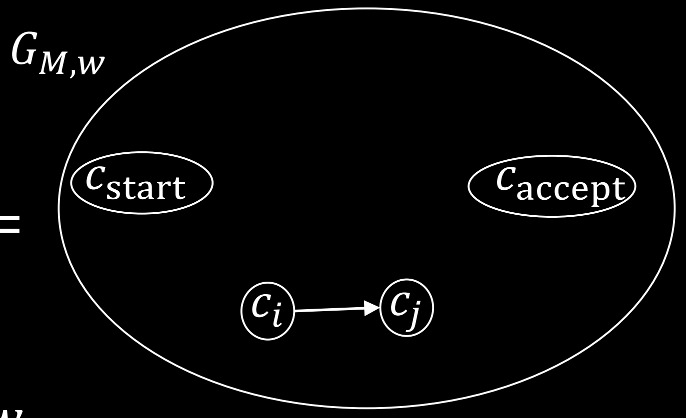
Give a log-space reduction f mapping A to $PATH$.

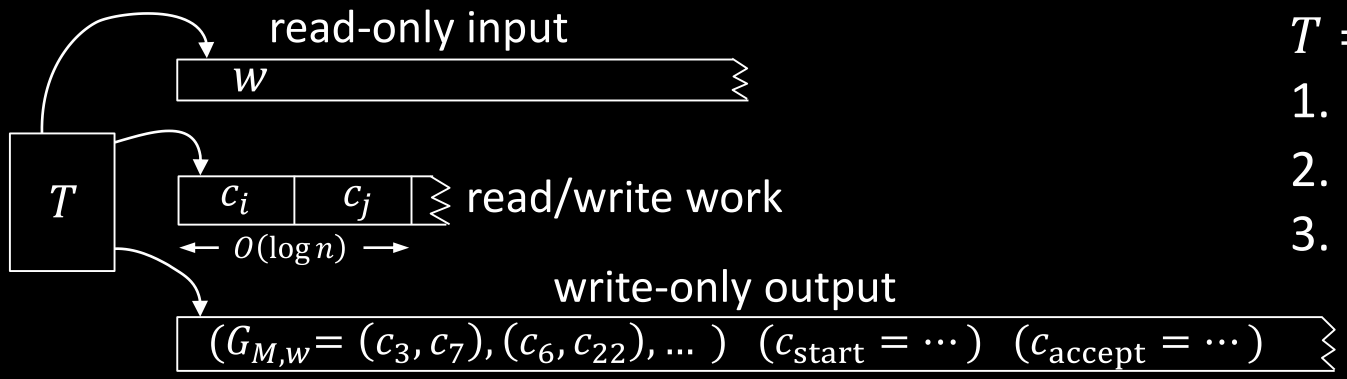
$f(w) = \langle G, s, t \rangle$

$w \in A$ iff G has a path from s to t

Here is a log-space transducer T to compute f in log-space.

$f(w) =$





$T =$ "on input w

1. For all pairs c_i, c_j of configurations of M on w.
2. Output those pairs which are legal moves for M.
3. Output c_{start} and c_{accept}."

$\overline{2SAT}$ is NL-complete

Theorem: $\overline{2SAT}$ is NL-complete

Proof: 1) Show $\overline{2SAT} \in$ NL   good exercise

2) Show $PATH \leq_L \overline{2SAT}$

Give log-space reduction f from $PATH$ to $\overline{2SAT}$.

$f(\langle G, s, t\rangle) = \langle \phi \rangle$

For each node u in G put a variable x_u in ϕ.

For each edge (u, v) in G, put a clause $(x_u \rightarrow x_v)$ in ϕ [equivalent to $(\overline{x_u} \vee x_v)$].

In addition put the clauses $(x_s \vee x_s)$ and $(x_t \rightarrow \overline{x_s})$ in ϕ.

Show G has an path from s to t iff ϕ is unsatisfiable.

(→) Follow implications to get a contradiction.

(←) If G has no path from s to t, then assign all x_u TRUE where u is reachable from s, and all other variables FALSE. That gives a satisfying assignment to ϕ.

Straightforward to show f is computable in log-space.


Coffee Break

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

Proof: Show $\overline{PATH} \in$ NL

Defn: NTM M computes function $f: \Sigma^* \to \Sigma^*$ if for all w

1) All branches of M on w halt with $f(w)$ on the tape or reject.
2) Some branch of M on w does not reject.

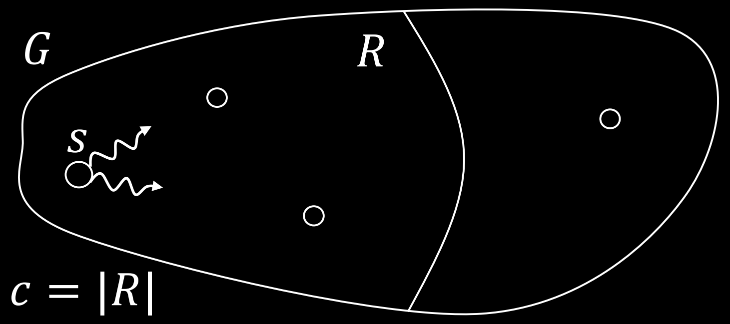
Let $path(G, s, t) = \begin{cases} \text{YES, if } G \text{ has a path from } s \text{ to } t \\ \text{NO, if not} \end{cases}$

Let $R = R(G, s) = \{u \mid path(G, s, u) = \text{YES}\}$

Let $c = c(G, s) = |R|$

R = Reachable nodes

c = # reachable



Check-in 20.2

Consider the statements:

(1) $\overline{PATH} \in$ NL, and

(2) Some NL-machine computes the $path$ function.

What implications can we prove *easily*?

(a) (1) → (2) only

(b) (2) → (1) only

(c) Both implications

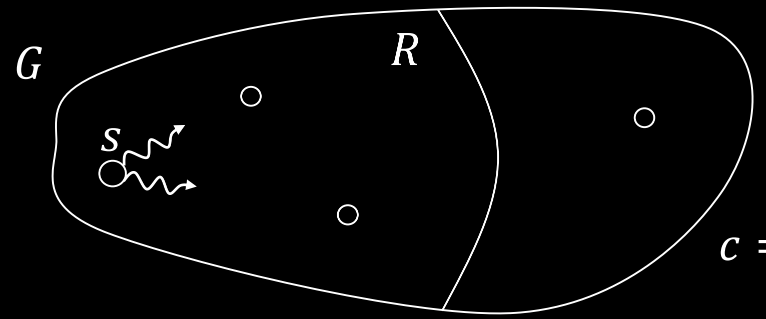
(d) Neither implication

NL = coNL (part 2/4) – key idea

Theorem: If some NL-machine computes c, then some NL-machine computes $path$.

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   - (p) Nondeterministically pick a path from s to u of length $\leq m$.
     If fail, then *reject*.
     If $u = t$, then output YES, else set $k \leftarrow k + 1$.
   - (n) Skip u and continue.
5. If $k \neq c$ then *reject*.
6. Output NO." [found all c reachable nodes and none were t}



$c = |R|$

NL = coNL (part 3/4)

Let $path_d(G, s, t) = \begin{cases} \text{YES, if } G \text{ has a path } s \text{ to } t \text{ of length} \leq d \\ \text{NO, if not} \end{cases}$

Let $R_d = R_d(G, s) = \{u |\ path_d(G, s, u) = \text{YES}\}$

Let $c_d = c_d(G, s) = |R_d|$

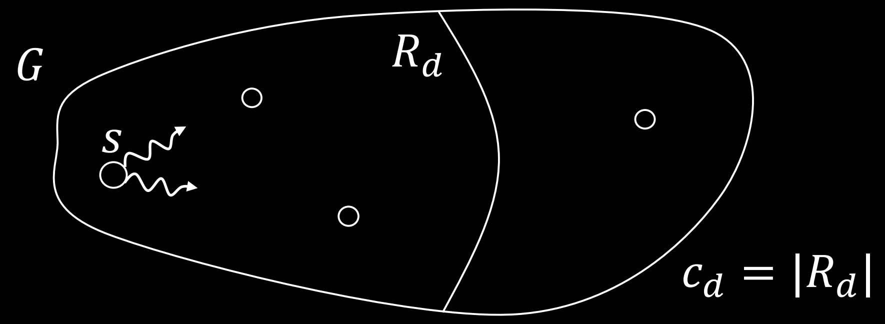
Theorem: If some NL-machine computes c_d, then some NL-machine computes $path_d$.

Proof: "On input $\langle G, s, t\rangle$

1. Compute c_d
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)

   (p) Nondeterministically pick a path from s to u of length $\leq d$.
   If fail, then *reject*.
   If $u = t$, then output YES, else set $k \leftarrow k + 1$.

   (n) Skip u and continue.
5. If $k \neq c_d$ then *reject*.
6. Output NO" [found all c_d reachable nodes and none were t}



NL = coNL (part 4/4)

Theorem: If some NL-machine computes c_d, then some NL-machine computes $path_{d+1}$.

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   - (p) Nondeterministically pick a path from s to u of length $\leq d$.
     If fail, then *reject*.
     If u has an edge to t, then output YES, else set $k \leftarrow k + 1$.
   - (n) Skip u and continue.
5. If $k \neq c_d$ then *reject*.
6. Output NO." [found all c_d reachable nodes and none had an edge to t}

Corollary: Some NL-machine computes c_{d+1} from c_d.

Check-in 20.3

Can we now show $2SAT$ is NL-complete?

(a) No.

(b) Yes.

Yes: $\overline{PATH} \leq_{\mathrm{L}} PATH$ & $PATH \leq_{\mathrm{L}} \overline{2SAT}$

So $\overline{PATH} \leq_{\mathrm{L}} \overline{2SAT}$ thus $PATH \leq_{\mathrm{L}} 2SAT$

Quick review of today

1. Log-space reducibility
2. L = NL? question
3. $PATH$ is NL-complete
4. $\overline{2SAT}$ is NL-complete
5. NL = coNL