Last week: Space Complexity

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

We know $L \subsetneq PSPACE$ (in fact also $NL \subsetneq PSPACE$)

but suspect most (all?) inclusions are strict.

(i.e $L \subsetneq NL \subsetneq P \subsetneq NP \subsetneq PSPACE$)

Antagonist in today's lecture: NL

$$NL = \{ L \subseteq \{0,1\}^* \mid L \text{ decided by an NTM running in space } O(\log n) \}$$

Recall: Reductions in NL are defined via implicitly computable log-space functions

i.e. $f: \{0,1\}^* \rightarrow \{0,1\}^*$ such that

① $|f(x)| \leq poly(|x|)$

② $\{\langle x, i\rangle \mid f(x)_i = 1\} \in L$

③ $\{\langle x, i\rangle \mid i \leq |f(x)|\} \in L$

$A \leq_l B$: A reduces to B via an (implicitly computable) log-space reduction

Properties: ① $A \leq_l B$ & $B \in L \Rightarrow A \in L$
log space

② $A \leq_l B$ & $B \leq_l C \Rightarrow A \leq_l C$

Thm 1: $\text{Path} = \{\langle G, s, t\rangle \mid \exists$ a path from s to t in G $\}$

is NL-complete. (via log-space reduction)

Need to show 2 things

① Path $\in$ NL

② Any language $A \in$ NL reduces to Path.

For ②, fix $A \in$ NL & let M be an NTM using space $S(n) = O(\log n)$ deciding A.

Define the reduction as follows:

$$f : \{0,1\}^* \to \{0,1\}^*$$

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

G: configuration graph of M on x

s: start configuration

t: accept configuration

$$|f(x)| \leq O(|G| + \log S(n))$$

(assume $|x| = n$) $= O(|G| + \log n)$

Let us assume that we represent G as an adjacency matrix.

$$|G| = O(|V(G)|^2) = 2^{O(\log n)} = \text{poly}(n)$$

$$\Rightarrow |f(x)| \leq \text{poly}(n)$$

The length of $f(x)$ is easily computable in space $O(\log n)$

[exercise]

Finally, to show f is implicitly log space computable, we need to able to compute any bit of G, s or t in space $O(\log n)$.

→ s & t are easy (they are only $O(\log n)$ bits long)

→ For G, we need to recall that for 2 vertices $u, v \in G$, we can check if $(u,v) \in E(G)$ or not in space $O(\log n)$. Finishes part ② of Thm 1.

On to ① in Thm 1.

→ Easy to do via NTMs.

→ Let's see another way, via **certificates**

Recall: $A \in NP$ if & only if there is a DTM M (verifier) running in polynomial time such that

$$x \in A \iff \exists y \quad |y| \le poly(|x|) \text{ \& } M(x,y) = 1.$$

How can we modify this to give a similar characterization of NL?

Two changes:

(i) Make M a DTM running in logspace. (i.e Space$(M, x) = O(\log n)$ where $n = |x|$)

(ii) Make the certificate read-once

i.e the TM M receives the certificate on a new tape ("certificate tape") where the head can only move right.

[Without the "read-once" condition on the certificate tape, this definition actually captures all of NP.]

Claim: $A \in NL$ if & only if

there is a DTM M with a read-once certificate running in logspace such that

$$x \in A \iff \exists y \; |y| \le poly(|x|)$$

$$M(x, y) = 1$$

input tape ↲ ↳ on certificate tape

The proof is quite easy & sketched below:

($\Rightarrow$) If $A \in NL$, then it has an NTM N in logspace. Use it to create a DTM M that simulates N using the certificate tape to simulate the non-deterministic choices

($\Leftarrow$) Given a DTM M with certificates, we can create an NTM N simulating M by guessing the next bit on the certificate tape.

Showing Path $\in$ NL via certificates

To show $\langle G, s, t \rangle$ is Path, the certificate is just a path (or more specifically a walk) from s to t in G with at most n vertices where $n = |V(G)|$. The machine M just needs to check

(a) the first vertex is s & the last is t

(b) If the path is $v_0 = s, v_1, \ldots, v_k = t$ then $(v_i, v_{i+1}) \in E(G)$ for all $i \in \{0, \ldots, k-1\}$.

Finally, we are done with steps ① & ② of Thm 1, meaning that Path is NL-complete

We can now define the class co-NL analogously to co-NP.

$$\boxed{\text{co-NL} = \{A \mid \overline{A} \in \text{NL}\}}$$

Since any $A \in \text{NL}$ reduces to Path,

$\overline{A} \in \text{co-NL}$ reduces to $\overline{\text{Path}}$

Thus, $\overline{\text{Path}}$ is co-NL complete (using logspace reductions).

Like NP vs co-NP, we can also ask if $NL = co\text{-}NL$. Here, we have a surprising (?) answer.

$$\boxed{\begin{array}{l}\underline{\text{Thm}}\ 2\ \text{(Immerman - Szelepcsényi theorem)} \\ \qquad NL = co\text{-}NL\end{array}}$$

To show this, we only need to show that $co\text{-}NL \subseteq NL$ (exercise!)

& as $\overrightarrow{\text{Path}}$ is co-NL complete, it is enough to show that

$$\boxed{\begin{array}{l}\text{Thm 3: (also the I-S theorem)} \\ \qquad \overline{\text{Path}} \in NL.\end{array}}$$

<u>Proof</u>: Need to give a certificate for $\langle G, s, t\rangle \in \overline{\text{Path}}$

(i.e. the certificate shows that there is <u>no path</u> from s to t.)

Moreover, the certificate is checkable by a DTM M that uses log space & reads the certificate on a <u>read-once tape</u>.

<u>Two steps</u>:

① Assume we know

c = # of vertices reachable from <u>s</u>

& give a certificate for $\langle G, s, t\rangle \in \overline{\text{Path}}$

② Give a certificate for c

Step ①:

→ let n = # of vertices of G

& $V(G) = \{v_1, \dots, v_n\}$

→ For vertex v_i:

→ A bit $b_i \in \{0,1\}$ that tells us whether v_i is reachable from s or not [$b_i = 1 \iff v_i$ reachable]

→ If $b_i = 1$ (i.e. v_i supposedly reachable)

a path p_i from s to v_i.

(If $b_i = 0$, p_i = empty)

The overall certificate

$$\underbrace{(\underbrace{b_1, p_1}_{O(n \log n) \text{ bits}}, b_2, p_2, \dots, b_n, p_n)}_{O(n^2 \log n) \text{ bits}}$$

Verifying this certificate:

The verification process needs to check 3 things:

(i) $b_1 + \cdots + b_n = c$

(ii) If $t = v_i$, $b_i = 0$

(iii) Each p_i gives a valid path from s to v_i.

Each can be checked in parallel by a log-space machine in read-once fashion!

Do all the three checks in parallel.

Takes space $O(\log n) \times 3 = O(\log n)$.

Step (2):

let $C_j = \{v_i \mid v_i$ reachable from s by a path of length $\leq j\}$

Obs: $C_0 = \{s\} \subseteq C_1 \subseteq C_2 \cdots \subseteq C_n$

& $|C_n| = c$

(any vertex reachable from s is reachable by path of length $\leq n$)

So $|C_0| = 1$ is known & we want to certify $|C_n| = c$.

For each $j < n$, we will certify $|C_{j+1}|$ given $|C_j|$.

Certificate:

→ For each $i \in \{1, \ldots, n\}$ supposedly

→ a bit $b_i \in \{0, 1\}$, indicating if $\underbrace{v_i \notin C_{j+1}}_{b_i = 0}$ or $\underbrace{v_i \in C_{j+1}}_{b_i = 1}$

→ If $b_i = 1$, certify by a path of length $\leq j$

→ If $b_i = 0$, a certificate showing $v_i \notin C_{j+1}$ (next page)

Overall certificate

$(b_1, \sigma_1, b_2, \sigma_2, \ldots, b_n, \sigma_n)$

certifying if $v_i \notin C_{j+1}$ or $v_i \in C_{j+1}$

If everything is correct, $b_1 + \cdots + b_n = |C_{j+1}|$.

Certifying $v_i \notin C_{j+1}$ given $|C_j|$.

→ For each $k \in \{1, \dots, n\}$.

→ a bit $b_k' \in \{0,1\}$ that indicates if $v_k \in C_j$ or not.

→ If $b_k' = 1$, a path p_k from s to v_k of length $\leq j$.

So certificate $= (b_1', p_1, \dots, b_n', p_n)$

Verification:

(i) $b_1' + b_2' + \dots + b_n' = |C_j|$

(ii) v_k an in-neighbour of v_i

$$\implies b_k' = 0.$$

(iii) $b_k' = 1 \implies p_k$ a path of length $\leq j$ from s to v_k.

Each can be done in log-space as before!

Verification of overall certificate for $|C_{j+1}|$ given $|C_j|$.

→ For each $i \in \{1, \ldots, n\}$

→ If $b_i = 1$, check if σ_i is a path from s to v_i of length $\leq j+1$

→ If $b_i = 0$, check σ_i as on the previous page.

→ Compute $|C_{j+1}| = b_1 + \cdots + b_{n+1}$.

Note: Space re-used between iterations.

The only additional space we need is to store $\underbrace{i,\ c_j,\ \text{the current sum } b_1 + \cdots + b_i}_{O(\log n) \text{ bits}}$

Final <u>certificate</u> for path

$(y_1, y_2, \ldots, y_n, y)$

y_j - certifies $|C_j|$ given $|C_{j-1}|$

y - certifies $\langle G, s, t\rangle \in \overline{\text{Path}}$ given $|C_n| = c$.

$|y_j| = O(n^3 \log n)$

$\Rightarrow$ Overall length $= O(n^4 \log n)$
$= \text{poly}(n)$.