Hardness HW 1

TAMREF
tamref.yun@snu.ac.kr

Due: September 29, 2022

Rules

- You can just link your former post instead of the solution. Otherwise, it is recommended to write the proof in your language.
- Subproblems are separated for explanatory convenience. You can elaborate the answer for each subproblem, or just provide the whole solution if possible.
- You are qualified if you **read** all the problems, and answered **at least 3** of them. Most easy problems belong to the first page.

Question 1. Integer Program

(a) For $A \in \mathbb{Z}^{N \times M}$ and $b \in \mathbb{Z}^N$, **Zero-One Integer Program** decideds if there is an $x \in \{0,1\}^M$ such that $Ax \geq b$. Prove **SAT** $\leq_p$ **Zero-One Integer Program** to show that it is $\mathbb{NP}-\text{complete}$.

(b) (Optional) If the candidate of $x$is relaxed into $\mathbb{Z}^M_{\geq 0}$, the problem is called **Integer Program (IP)**. What differs in $\mathbb{NP}-\text{complete}$ness of **IP**? Fill the gap to prove it.

Question 2. Pratt's theorem

PRIMES is to test the primality of given integer, the only input.

(a) Show that n is prime if and only if there is an integer g such that $n-1$ is the smallest exponent that $g^{n-1} \equiv 1 \pmod{n}$.

(b) From (a), deduce that n is prime if and only if $g^{(n-1)/q} \not\equiv 1 \pmod{n}$, for every prime divisor q of n.

(c) From (b), provide the polynomial size witness for **PRIMES**.

Question 3. $\mathbb{ZPP} = \mathbb{RP} \cap \text{co}-\mathbb{RP}$

Following the definitions in the slide, prove that $\mathbb{ZPP} = \mathbb{RP} \cap \text{co}-\mathbb{RP}$.

Question 4. Lousy $\mathbb{RP}$ and $\mathbb{BPP}$

For the original definition of $\mathbb{RP}$ and $\mathbb{BPP}$, refer the slide.

(a) Let $\mathbb{RP}_\alpha$ denote the complexity class, where $\Pr[M(x) = \text{yes} \mid x \in A] \geq \alpha$. For all constant $0 < \alpha < 1$, Show that $\mathbb{RP}_\alpha = \mathbb{RP}$.

(b) For $\mathbb{RP}_{1/n^2}$ and $\mathbb{RP}_{1/2^n}$ defined similarly, where n is size of the input, determine whether or not it is equal to $\mathbb{RP}$.

(c) Similarly define $\mathbb{BPP}_\alpha$. For which α we can insist that $\mathbb{BPP}_\alpha = \mathbb{BPP}$?

Question 5. Classic inclusions

Show that $\mathbb{NP} \subseteq \mathbb{PSPACE} \subseteq \mathbb{EXPTIME}$.

Question 1. Integer Program

(a) For $A \in \mathbb{Z}^{N \times M}$ and $b \in \mathbb{Z}^N$, **Zero-One Integer Program** decideds if there is an $x \in \{0,1\}^M$ such that $Ax \geq b$. Prove **SAT** $\leq_p$ **Zero-One Integer Program** to show that it is $\mathbb{NP}-\text{complete}$.

(b) (Optional) If the candidate of $x$is relaxed into $\mathbb{Z}^M_{\geq 0}$, the problem is called **Integer Program (IP)**. What differs in $\mathbb{NP}-\text{complete}$ness of **IP**? Fill the gap to prove it.

(a) Let $0 \leq v_i \leq 1$ represent the i-th variable x_i of SAT. Then $\neg x_i$ corresponds to $1 - v_i$.

The clause $(x_{i_1} \vee \cdots \vee x_{i_n} = \text{true})$ corresponds to the constraint $v_{i_1} + \cdots + v_{i_n} \geq 1$.

Hence, we show that SAT $\leq_p$ ZERO-ONE Integer program to prove that ZOIP is NP-hard.

Indeed. ZOIP is trivially in NP, thus it's in NP-complete. □

(b) IP differs from ZOIP, because IP is not guaranteed to have a witness that is not so big

Employing some additional variables, we replace the inequality cond's to equality conds.

$(Ax)_i \geq b_i \Rightarrow (Ax)_i + s_i = b_i$, with $s_i \geq 0$.

Also, letting $x_i := x_i^+ - x_i^-$ removes the negative solutions. Now we re-write the problem

$$\text{find} \quad Ax = b$$
$$x \geq 0.$$

By Cramer's rule, $x_i = \dfrac{\det(A_i)}{\det(A)}$ where $A_i = (a_1, \cdots, a_{i-1}, b, a_{i+1}, \cdots, a_n)$

given $A = (a_1, \cdots, a_n)$ as column vectors.

Note that $\left|\dfrac{\det A_i}{\det A}\right| \leq |\det A_i| \leq n! \times W^n$ given. $|a_{ij}|, |b_i| \leq W$.

Thus we have a witness x_i, fits in $n \times \log(W^n \times n!) = \mathcal{O}(n^2 \log(nW))$.

Thus IP $\in$ NP. □

Question 2. Pratt's theorem

PRIMES is to test the primality of given integer, the only input.

(a) Show that n is prime if and only if there is an integer g such that $n-1$ is the smallest exponent that $g^{n-1} \equiv 1 \pmod{n}$.

(b) From (a), deduce that n is prime if and only if $g^{(n-1)/q} \not\equiv 1 \pmod{n}$, for every prime divisor q of $n-1$.

(c) From (b), provide the polynomial size witness for **PRIMES**.

(a) For the existence of such g, g must be the primitive root of $\mathbb{Z}_n^{\times}$ and $\varphi(n) = n-1$ should hold. It's enough to show that n is prime. The converse is obvious, as all prime numbers have a primitive root. □

(b). If $g^k \equiv 1 \pmod{n}$ for $k < n-1$, $g^{\gcd(n,k)} \equiv 1 \pmod{n}$ implies the existence of q: prime divisor of n, such that $g^{\frac{n-1}{q}} \equiv 1 \pmod{n}$. The converse is obvious. □

(c). Witness(n) is constructed by:

Witness(2) = {}. (base case).

For $q_1, \cdots, q_r$, the all prime divisors of $n-1$.

$$\text{Witness}(n) = \left(g, q_1, \cdots, q_r, \text{Witness}(q_1), \cdots, \text{Witness}(q_r)\right).$$

Where g is a primitive root.

As $|\text{Witness}(n)| = \sum_{i=1}^{r} |\text{Witness}(q_i)| + O(\log n) = O(\log^2 n)$,

it works as a polynomial-size witness to prove that PRIMES $\in$ NP. □

Question 3. $\mathbb{ZPP} = \mathbb{RP} \cap \text{co}-\mathbb{RP}$

Following the definitions in the slide, prove that $\mathbb{ZPP} = \mathbb{RP} \cap \text{co}-\mathbb{RP}$.

ZPP $\subseteq$ RP : Output "no" if $M(x)$ = "idk".

ZPP $\subseteq$ co-RP : Output "yes" if $M(x)$ = "idk".

RP $\cap$ co-RP $\subseteq$ ZPP : For a problem $A \in$ RP $\cap$ co-RP, suppose there is an algorithm M_{RP} & $M_{co\text{-}RP}$ meeting the requirements of RP & co-RP, respectively.

Then a hybrid algorithm M goes like:

$M(x)$:

① $M_{RP}(x)$ = "yes" $\Rightarrow$ return "yes"

② $M_{co\text{-}RP}(x)$ = "no" $\Rightarrow$ return "no".

③ return "idk"

Output of M other than "idk" should be correct. The probability of "idk" is

$$\Pr[M(x) = \text{idk}] = \Pr[M_{RP}(x) = \text{no} \mid x \in A] \Pr[x \in A] + \Pr[M_{co\text{-}RP}(x) = \text{yes} \mid x \notin A] \Pr[x \notin A]$$

$$\leq \frac{1}{2}\left(\Pr[x \in A] + \Pr[x \notin A]\right) = \frac{1}{2}. \quad \square$$

Question 4. Lousy $\mathbb{RP}$ and $\mathbb{BPP}$

For the original definition of $\mathbb{RP}$ and $\mathbb{BPP}$, refer the slide.

(a) Let $\mathbb{RP}_\alpha$ denote the complexity class, where $\Pr[M(x) = \text{yes} \mid x \in A] \geq \alpha$. For all constant $0 < \alpha < 1$, Show that $\mathbb{RP}_\alpha = \mathbb{RP}$.

(b) For $\mathbb{RP}_{1/n^2}$ and $\mathbb{RP}_{1/2^n}$ defined similarly, where n is size of the input, determine whether or not it is equal to $\mathbb{RP}$.

(c) Similarly define $\mathbb{BPP}_\alpha$. For which α we can insist that $\mathbb{BPP}_\alpha = \mathbb{BPP}$?

(a) Trivial.

(b). $RP_{\frac{1}{n^2}} = RP$, by repeating algorithm $\Omega(n^3)$ time.

$RP_{1/2^n}$ is unlikely to be equal to RP. For SAT, trying a random assignment and guessing yes/no gives at least $\frac{1}{2^n}$ probability. Accepting this leads to RP = NP which is quirky.

(c) For $1/2 < \alpha < 1$, $BPP_\alpha = BPP$.

For $\alpha \leq \frac{1}{2}$, Coin flip gives the strategy. Picking an EXPSPACE-complete problem and applying Coin-Flip works. Thus $BPP \neq BPP_{1/2^n}$, since $BPP \subseteq PSPACE \neq EXPSPACE$. □

Question 5. Classic inclusions

Show that NP ⊆ PSPACE ⊆ EXPTIME.

① NP ⊆ PSPACE.

Pick a polynomial p such that witness has size $\leq p(|x|)$. (x : input.) Since the verifier runs in polynomial time, running verifier with $2^{p(|x|)}$ all possible witnesses uses at most polynomial space. □

② PSPACE ⊆ EXPTIME.

Pick a polynomial p such that the program is constrained in $\leq p(|x|)$ memory. The advance of program is only determined by the memory state. Construct a directed graph with $2^{p(|x|)}$ vertices, adding an outgoing edge to the next memory state for each non-terminal state.

Running algorithm for $2^{p(|x|)} + 1$ steps using $\mathcal{O}(p(|x|) \cdot 2^{p(|x|)})$ time, the algorithm terminates or fall into the infinite loop. Since the algorithm ($\in$ PSPACE) must terminate, only the first case is reachable. □

Question 6. Diagonal Argument

Assume the following facts.

- Given a TM M, deciding(computing) whether M terminates in polynomial time is **undecidable**.
- Given a TM M and input x, there is a **Universal TM** $\mathcal{U}$ taking the pair (M, x) as input, and simulate $M(x)$ for T discrete steps in $T \log T$ time.
- The set of TMs terminating in $\mathcal{O}(f(n))$ time is **countable** – so we may give them an enumeration.

(a) For $f(n)$ and $g(n)$ such that $f(n) \log f(n) = o(g(n))$, there is a problem could be solved in $\mathcal{O}(g(n))$ time, but never in $\mathcal{O}(f(n))$ time. To show that, Design a TM D terminates in $\mathcal{O}(g(n))$ time, which never could produce identical output with another TM M, terminating in $\mathcal{O}(f(n))$ time. The result is called the **Time Hierarchy Theorem.**

(b) From (a), prove that $\mathbb{P} \neq$ EXPTIME.

(a) D is designed as following:

- D takes TM m and the string x as input,
- D runs $\mathcal{U}(m, x)$ for $K \cdot g(n)$ for some constant K. If $\mathcal{U}(m, x)$ terminates with output "yes", output "no". Otherwise, output "yes".

we claim that there is no TM M terminating in $\mathcal{O}(f(n))$ time that runs just same as D.

Otherwise, take such TM and denote it M. Assume M terminates in $L \cdot f(|x|)$ time for input x. Then for sufficiently large n, take $x = (M, 1^n)$ where 1^n is the bit-string with n 1's.

Since M terminates in $L \cdot |M|^{O(1)} \cdot f(n)$ time, $\mathcal{U}(M, x)$ can successfully terminate for large K & n. But then the outputs from M and D disagree, contradiction. ↯.

(b). There's a problem can't be solved in $\mathcal{O}(1.9^n)$, by (a). □.

Question 6. Diagonal Argument

Assume the following facts.

- Given a TM M, deciding(computing) whether M terminates in polynomial time is **undecidable**.
- Given a TM M and input x, there is a **Universal TM** $\mathcal{U}$ taking the pair (M, x) as input, and simulate $M(x)$ for T discrete steps in $T \log T$ time.
- The set of TMs terminating in $\mathcal{O}(f(n))$ time is **countable** – so we may give them an enumeration.

(a) For $f(n)$ and $g(n)$ such that $f(n) \log f(n) = o(g(n))$, there is a problem could be solved in $\mathcal{O}(g(n))$ time, but never in $\mathcal{O}(f(n))$ time. To show that, Design a TM D terminates in $\mathcal{O}(g(n))$ time, which never could produce identical output with another TM M, terminating in $\mathcal{O}(f(n))$ time. The result is called the **Time Hierarchy Theorem.**

(b) From (a), prove that $\mathbb{P} \neq \mathbb{EXPTIME}$.

Question 7. $\mathbb{BPP} \subseteq \mathbb{PSPACE}$

There's an alternative definition for BPP.

Definition. $A \in \mathbb{BPP}$ if there's a polynomial algorithm M and another polynomial p, takes the original input x attached with the random string $r \in \{0,1\}^{p(|x|)}$, having $\Pr[M(x,r) = \text{yes} \mid x \in A] \geq \frac{3}{4}$ and $\Pr[M(x,r) = \text{no} \mid x \notin A] \geq \frac{3}{4}$.

Relying on the definition, prove that $\mathbb{BPP} \subseteq \mathbb{PSPACE}$. You may try to prove the equivalence of the definition given above, to the definition given in the slide.

pf Exploring all possible r consumes just polynomial memory. And then just follow the majority.

Equivalence of two definitions:

① Given a deterministic algorithm $M : (x, r) \mapsto M(x, r)$, we construct a randomized algorithm $M' : x \mapsto M'(x)$ by picking r in random and give $M(x, r)$. □

② Given a randomized polynomial-time algorithm $M : x \mapsto M(x)$, $\exists\, p$: polynomial such that M requires at most $p(|x|)$ random choices — since M is polynomial time. Thus design algorithm $\widetilde{M} : (x, r) \mapsto \widetilde{M}(x, r)$, by fixing the choice in random to r. □

Question 8. Equivalence of PH

Recall the oracle definition of PH classes, corrected from the lecture.

- $\Sigma_{i+1} := \mathbb{NP}^{\Sigma_i}$
- $\Pi_{i+1} := \text{co}-\mathbb{NP}^{\Pi_i}$

(a) Prove by induction, that the definition above is equivalent to the definition with alternating quantifiers.

(b) Show that if $\mathbb{P} = \mathbb{NP}$, $P = \Sigma_i$ for all $i \geq 1$.

(c) Show that if $\mathbb{NP} = \text{co}-\mathbb{NP}$, $\mathbb{NP}^{\mathbb{NP}} \subseteq \mathbb{NP}$. (Heavy!)

(d) Assuming (c), show that if $\mathbb{NP} = \text{co}-\mathbb{NP}$, $\Sigma_i = \Pi_i = \mathbb{NP}$ for all $i \geq 1$. This is a tremendous subcase of Polynomial Hierarchy Collapse.

Question 9. Hardness Results from Directed Graph Modeling

These are the class of problems could be solved in similar way. Give a survey to these problems:

(a) Show that the problem **QBF** is $\mathbb{PSPACE}-\text{complete}$.

(b) Show that $\mathbb{NPSPACE} = \mathbb{PSPACE}$. (Savitch's theorem)

(c) $\mathbb{NL}$ is the class of problems could be solved non-deterministically, with $\mathcal{O}(\log n)$ extra r/w memory. Be careful that the 'witness' is bounded in the read-only memory along the input, and it does not really restricted to be logarithmic size. (But bounded by polynomial) Show that the problem *"Given a directed graph G and $s, t \in V(G)$, is there a path from s to t?"* (So called **REACHIBILITY**) is $\mathbb{NL}-\text{complete}$.

(d) From (c), deduce that **2-SAT** is $\mathbb{NL}-\text{complete}$.

Question 10. **2-QBF**

Given a 2-CNF ϕ, the problem $\exists x_1 \forall x_2 \cdots Q_k x_k$ s.t. $\phi(x_1, \cdots, x_k)$ is called **2-QBF**.

Find the linear-time algorithm for **2-QBF**, and solve NERC 2018 Harder Satisfiability. (BOJ 16667)

Question 8. Equivalence of PH

Recall the oracle definition of PH classes, corrected from the lecture.

- $\Sigma_{i+1} := \mathrm{NP}^{\Sigma_i}$
- $\Pi_{i+1} := \mathrm{co\text{-}NP}^{\Pi_i}$

(a) Prove by induction, that the definition above is equivalent to the definition with alternating quantifiers.

(b) Show that if $\mathbb{P} = \mathbb{NP}$, $P = \Sigma_i$ for all $i \geq 1$.

(c) Show that if $\mathbb{NP} = \mathrm{co\text{-}NP}$, $\mathbb{NP}^{\mathbb{NP}} \subseteq \mathbb{NP}$. (Heavy!)

(d) Assuming (c), show that if $\mathbb{NP} = \mathrm{co\text{-}NP}$, $\Sigma_i = \Pi_i = \mathbb{NP}$ for all $i \geq 1$. This is a tremendous subcase of Polynomial Hierarchy Collapse.

(a). First, note that $\mathrm{NP}^{\Sigma_i} = \mathrm{NP}^{\Pi_i}$ & $\mathrm{co\text{-}NP}^{\Sigma_i} = \mathrm{co\text{-}NP}^{\Pi_i}$. As $A \in \Sigma_i \Leftrightarrow \bar{A} \in \Pi_i$, Σ_i and Π_i are identical for constant-time oracle. Let us recall the definition of Σ_i and Π_i using quantifiers:

$$\Sigma_n^Q := \{ A : \exists M \text{ s.t. } x \in A \Leftrightarrow \exists y_1 \forall y_2 \exists y_3 \cdots Q y_n \;\; M(x, y_1, \cdots, y_n) = 1 \}$$

(TM.)

$$\Pi_n^Q := \{ A : \exists M \text{ s.t. } x \in A \Leftrightarrow \forall y_1 \exists y_2 \forall y_3 \cdots Q y_n \;\; M(x, y_1, \cdots, y_n) = 1 \}.$$

Also, denote Σ_i, Π_i defined using oracles by Σ_i^0 and Π_i^0, respectively. By showing $\Sigma_i^0 = \Sigma_i^Q$, $\Pi_i^0 = \Pi_i^Q$ immediately follows since $\Pi_{i+1}^0 = \overline{\mathrm{NP}^{\Sigma_i^0}} = \overline{\mathrm{NP}^{\Sigma_i^0}} = \overline{\Sigma_{i+1}^0}$ and $\Pi_{i+1}^Q = \overline{\Sigma_{i+1}^Q}$ obviously.

$\Sigma_n^Q \subseteq \Sigma_n^0$) From $A \in \Sigma_n^Q$ and corresponding TM M, define $\tilde{A}$ by

$$\tilde{A} := \{ (x, y_1) : \forall y_2 \exists y_3 \cdots Q y_n \;\; M(x, y_1, y_2, \cdots, y_n) = 1 \} \in \Pi_{n-1}^Q.$$

Since $x \in A$ can be decided in NP by membership oracle in $\tilde{A}$, $A \in \mathrm{NP}^{\Pi_{n-1}^Q} = \Sigma_n^0$.

$\Sigma_n^0 \subseteq \Sigma_n^Q$) From $A \in \Sigma_n^0$, there is a non-deterministic TM M and a problem $B \in \Sigma_{n-1}^Q$, given oracle access to B to decide $x \in A$. Let N be the TM corresponding to B.

$$x \in A \Leftrightarrow \exists q_1, \cdots, q_\ell, r_1, \cdots, r_t \text{ such that } q_i \in B,\; r_i \notin B, \quad M(x, q_1, \cdots, q_\ell, r_1, \cdots, r_t) = 1.$$

$$\Leftrightarrow \exists q_1, \cdots, q_\ell, r_1, \cdots, r_t, q_1^{(2)}, \cdots, q_\ell^{(2)} \text{ such that}$$

$$\forall r_1^{(2)}, \cdots, r_t^{(2)} \;\; \exists r_1^{(3)}, \cdots, r_t^{(3)} \;\; \cdots \cdots$$

$$\begin{cases} M(x, q_1, \cdots, q_\ell, r_1, \cdots, r_t) = 1. \\ N(q_i, q_i^{(2)}, \cdots, q_i^{(m)}) = 1. \\ N(r_i, r_i^{(2)}, \cdots, r_i^{(m)}) = 0. \end{cases}$$

$q_1^{(3)}, \cdots, q_{l_2}^{(3)}$ $\quad$ $q_1^{(4)}, \cdots, q_{l_2}^{(4)}$ $\quad$ $)$

which fits into definition of Σ_i-problem. □

(b) $\Sigma_1 = NP = P$.

Assuming $\Sigma_i = P$, $\Sigma_{i+1} = NP^P = P^P = P$. □

(c). $NP^{NP} \subseteq NP$. assuming $NP = \overline{NP}$.

Note that $NP^{NP} = NP^{SAT}$, so enough to show that $NP^{SAT} \subseteq NP$.

Note that $\overline{SAT} \leq_p SAT$ since $NP = \overline{NP}$, for a polynomial time turing machine M

$x \in A \iff \exists_{\substack{q_1, \cdots, q_n \in SAT \\ e_1, \cdots, e_n}}$ such that $M(x, q_1, \cdots, q_n) = 1$ and $q_i(e_i) = 1$

(we can hide $t_1, \cdots, t_l \in \overline{SAT}$ used as oracles, as they are reducible to SAT).

thus $A \in NP$. □

(d). $\Sigma_2 = NP^{NP} = NP$
$\Pi_2 = co\text{-}NP^{NP} = NP$. (base case)

Assuming $\Sigma_{i-1} = \Pi_{i-1} = NP$, $\Sigma_i = \Pi_i = NP^{NP} = NP$. □

Question 9. Hardness Results from Directed Graph Modeling

These are the class of problems could be solved in similar way. Give a survey to these problems:

(a) Show that the problem **QBF** is $\mathbb{PSPACE}-\text{complete}$.

(b) Show that $\mathbb{NPSPACE} = \mathbb{PSPACE}$. (Savitch's theorem)

(c) $\mathbb{NL}$ is the class of problems could be solved non-deterministically, with $\mathcal{O}(\log n)$ extra r/w memory. Be careful that the 'witness' is bounded in the read-only memory along the input, and it does not really restricted to be logarithmic size. (But bounded by polynomial) Show that the problem *"Given a directed graph G and $s, t \in V(G)$, is there a path from s to t?"* (So called **REACHIBILITY**) is $\mathbb{NL}-\text{complete}$.

(d) From (c), deduce that **2-SAT** is $\mathbb{NL}-\text{complete}$.

(a) QBF $\in$ PSPACE is obvious. We may try all assignments within polynomial memory by recursion.

For $A \in$ PSPACE, assume A uses $\leq p(n)$ space.

There are at most $2^{p(n)}$ memory states, put the states as vertices of the graph, and draw directed edge along the process in the program.

An input x is accepted if and only if there is a path from the starting state $s(x)$ to the accepting state t, with length $\leq 2^{p(n)}$. Let $P(u,v,k)$ denote the statement that "there is a path from u to v with length $\leq k$". First we try the recursive QBF:

$$P(u,v,k) := \exists w \ (P(u,w,k-1) \wedge P(w,v,k-1)).$$

But this is unacceptable, requiring exponential terms to construct desired $P(s_x, t, p(n))$.

We simplify the QBF as below:

$$P(u,v,k) := \exists w \forall x \forall y \ [((x,y) = (u,w) \vee (x,y) = (w,v)) \rightarrow P(x,y,k-1)].$$

Which is a bundle of (bitwise) boolean formula. The clause $(x,y) = (u,w)$ consumes at most $\leq p(n)$ boolean clauses, so $P(s_x, t, p(n))$ is QBF with $\leq p(n)^2$ length. $\square$

$f(n) \geq n$.

(b). Let NPSPACE$(f(n))$ denote the class of problem can be solved with $O(f(n))$ extra r/w-memory, non-deterministically. Constructing the same directed graph, nothing changes but ① there are multiple outgoing edges from single state and ② there are at most $2^{f(n)}$ states. In the same way, we construct QBF with $O((n+f(n))^2) = O(f(n)^2)$ length to solve the original problem.

$\therefore$ NPSPACE $= \bigcup_k$ NPSPACE$(n^k) \subseteq$ PSPACE. $\square$

(c) REACH $\in$ NL, as we can traverse the graph non-deterministically if there's a path from s to t. In converse, given a problem $A \in NL$, with $\leq c \cdot \log n$ memory consumption, convert the problem into a directed graph with $O(n^c)$ vertices, and the reachibility is equivalent to whether the input is accepted. This reachibility can be solved within $O(\log n^c) = O(\log n)$. $\square$

(d). Lemma. (Immerman, 1987) NL = co-NL.

pf) Enough to show that $\overline{\text{REACH}} \in$ NL. Basically, we provide the "count" of reachable vertices. If there are C_ℓ vertices in $V(G)$ which can reach t with at most ℓ edges, we can determine if v can reach t within $\leq \ell$ edges, non-deterministically for arbitrary v.

For every vertex $u \in V(G) - \{v\}$, guess if u can reach t within ℓ steps. Then there are $C_\ell - 1$ or C_ℓ such vertices among $V(G) - \{v\}$. For the first case, v must be able to reach t. Otherwise, v is unreachable.

Denote such program as PATH(v, ℓ, C_ℓ). PATH(v, ℓ, C_ℓ) is NL-TuringMachine given the exact C_ℓ. We compute C_ℓ from $C_{\ell-1}$ and PATH$(v, \ell-1, C_{\ell-1})$. For each $u \in V(G)$ and its out-neighbor v, Determine if u can reach t within $\leq \ell$ steps by running PATH$(u, \ell-1, C_{\ell-1})$ & PATH$(v, \ell-1, C_{\ell-1})$. Thus for $\ell = 1, \cdots, n-1$, Compute C_ℓ and then run PATH$(s, n-1, C_{n-1})$ gives the reachibility (or un-reachibility) of s to t. Since REACH $\in$ co-NL, NL = co-NL. $\square$

Now we prove that 2-SAT is co-NL complete. Take the implication graph, and then 2-SAT reduces to un-reachibility from x_i to $\neg x_i$, for each variable x_i.

Now, we reduce $\overline{\text{REACH}}$ into 2-SAT. Take x_u denote the literal "u is unreachable to t". We assert $\neg x_t$ and x_s, and $x_u \to x_v$ for each $(u, v) \in E(G)$. Then the SAT instance is accepted iff s is unreachable. $\square$

Question 10. **2-QBF**

Given a 2-CNF ϕ, the problem $\exists x_1 \forall x_2 \cdots Q_k x_k$ s.t. $\phi(x_1, \cdots, x_k)$ is called **2-QBF**.
Find the linear-time algorithm for **2-QBF**, and solve NERC 2018 Harder Satisfiability. (BOJ 16667)

Construct the implication graph, and denote $x \longrightarrow y$ by "y is reachable from x" and $x \not\longrightarrow y$ by its negation. First we add the constraint we learned from 2-SAT.

- if $Q_i = \exists$, $(x_i \not\longrightarrow \neg x_i$ or $\neg x_i \not\longrightarrow x_i)$. $\Rightarrow$ $x_i, \neg x_i$ belong to different SCC. .... ①

If $Q_i = \forall$ and $Q_j = \forall$, $x_i \not\longrightarrow x_j$ and $x_i \not\longrightarrow \neg x_j$. $\Rightarrow$ For all $i_1, \cdots, i_n$ s.t. $Q_{i_j} = \forall$, $x_{i_j}, \neg x_{i_j}$ make antichain with size $2n$ .... ②

If $Q_i = \exists$ and $Q_j = \forall$ for $i < j$, $x_i \longrightarrow x_j$ forces x_i to be false, $x_j \longrightarrow x_i$ forces x_i to be true, $\Rightarrow$ x_i, x_j must belong to different SCC. ... ③

If ① – ③ are satisfied, we show that 2-QBF is satisfiable.

Decompose the implication graph into SCC, and there are at most 2 kinds of them:

① SCC with only literal with quantifier $\forall$.

② Literals with quantifier $\exists$, not containing both x_i and $\neg x_i$.

Note that for any SCC S_0 of type ②, there can't be two SCC T_0, T_1 of type 1 s.t. $T_0 \longrightarrow S_0$, $S_0 \longrightarrow T_1$ since it implies $T_0 \longrightarrow T_1$. WLOG, suppose $S_0 \longrightarrow T_1$ and $x_i \in S_0$, and $\neg x_i \in \overline{S_0} \neq S_0$.

Then $\overline{T_1} \longrightarrow \overline{S_0}$, thus $\overline{S_0} \not\longrightarrow S_0$ as it implies $\overline{T_1} \longrightarrow T_1$. Hence putting $x_i \leftarrow$ false is consistent (implication graph) to all other constraints.

For literals with quantifier $\exists$ in SCC of type ①, just assign along the only $\forall$-literal, since the component is totally uncorrelated with the component containing their negations.

For components of type ② without any correlation, just assign along topological sort like 2-SAT. □