

Solution to Introduction to the Theory of Computation

Lwins_Lights

Contents

1	Regular Languages	2
2	Context-Free Languages	3
3	The Church–Turing Thesis	4
4	Decidability	5
5	Reducibility	7
6	Advanced Topics in Computability Theory	9
7	Time Complexity	11
8	Space Complexity	13

1 Regular Languages

2 Context-Free Languages

3 The Church–Turing Thesis

4 Decidability

- 4.10 *Refer to the textbook.*
- 4.11 By the pumping lemma, $L(M)$ is an infinite language $\iff L(M)$ includes a string of at least length p . Since $R = \Sigma^p \Sigma^*$ is a regular language, it is easy to design a PDA P recognizing $L(M) \cap R$. Then use E_{PDA} 's decider to decide whether $L(M) \cap R = \emptyset$.
- 4.12 *Refer to the textbook.*
- 4.13 $L(R) \subseteq L(S) \iff L(R) \cup L(S) = L(S)$, which can be decided by EQ_{DFA} 's decider.
- 4.14 *Refer to the textbook.*
- *4.15 By the pumping lemma, $1^k \in L(G) \implies 1^{k+p!} \in L(G)$ for every $k \geq p$. Therefore $1^* \subseteq L(G) \iff \{1^k \mid k \leq p + p!\} \subseteq L(G)$, which can be easily checked by a TM in finite time.
- 4.16 Since $A = \{\langle R \rangle \mid R \text{ is a regular expression, } R \cap \Sigma^* 111 \Sigma^* \neq \emptyset\}$, we only need an E_{DFA} 's decider.
- 4.17 Suppose we have two DFAs D_1 and D_2 . Let D be a DFA recognizing $L(D_1) \oplus L(D_2)$, where \oplus means symmetric difference. By the pumping lemma, $L(D) \cap (\Sigma \cup \epsilon)^p = \emptyset \implies L(D) = \emptyset$, thus p can be the required length.
- *4.18 \Leftarrow : It is enough to design a TM, which recognizes C by checking whether $\langle x, y \rangle \in D$ for all possible y one by one.
 \Rightarrow : Suppose TM M recognizes C . Let $D = \{\langle x, y \rangle \mid x \in C \text{ and } y \text{ is the computation history of } M \text{ on input } x\}$, which is obviously decidable.
- *4.19 Let C be a recognizable but undecidable language, e.g., A_{TM} . Construct D provided by Problem 4.18. Letting homomorphism f satisfy $f(\langle x, y \rangle) = x$ for every x, y we obtain $f(D) = C$.
- 4.20 Let M be a TM which runs both \bar{A} 's recognizer and \bar{B} 's recognizer on its own input. M accepts when \bar{B} 's recognizer accepts and rejects when \bar{A} 's recognizer accepts. Clearly $C = L(M)$ separates A and B .
- 4.21 M is a DFA that accepts $w^{\mathcal{R}}$ whenever it accepts $w \iff L(M) = L(M)^{\mathcal{R}}$, which can be decided by EQ_{DFA} 's decider.
- 4.22 In order to determine whether $L(R)$ is prefix-free, it suffices to check whether the DFA recognizing $L(R)$ has the property that from a reachable accept state we could arrive an accept state again by several transitions.
- *4.23 *Refer to the textbook.*
- 4.24 A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ has an useless state $q \in Q$ if and only if PDA $P' = (Q, \Sigma, \Gamma, \delta, q_0, \{q\})$ recognizes \emptyset . Therefore we can use E_{PDA} 's decider to solve it.
- *4.25 *Refer to the textbook.*
- *4.26 M is a DFA that accepts some palindrome $\iff \text{CFL } \{w \mid w = w^{\mathcal{R}}\} \cap L(M) \text{ is not empty, which can be decided by } E_{\text{PDA}} \text{'s decider.}$
- *4.27 Refer to the solution to Problem 4.26.
- 4.28 x is a substring of some $y \in L(G) \iff L(G) \cap \Sigma^* x \Sigma^* \neq \emptyset$, which can be decided by E_{PDA} 's decider.
- 4.29 By the pumping lemma, $L(G)$ is an infinite language $\iff L(G)$ includes a string of at least length p . So we can first use $INFINITE_{\text{PDA}}$ from Problem 4.11 to check whether $|L(G)| = \infty$, and then check whether $x \in L(G)$ for all x of less-than- p length, one by one.
- 4.30 Let E be A 's enumerator which generates $\langle M_1 \rangle, \langle M_2 \rangle, \dots$ in order. Construct $D = \{\langle n \rangle \mid \langle n \rangle \notin L(M_n)\}$.

- 4.31 First, recursively determine whether $V \xRightarrow{*} w$ with some $w \in \Sigma^*$, for all variables V . Then, recursively determine whether $V \xRightarrow{*} xAy$ with some $x, y \in \Sigma^*$, for all variables V .
- 4.32 Construct DPDA $P'_{(q,x)}$ by modifying P , such that $L(P'_{(q,x)}) = \emptyset$ if and only if (q, x) is a looping situation for P . Then we only need E_{PDA} .

5 Reducibility

- 5.9 Reduce from A_{TM} . To determine whether TM M accepts w , construct TM N which always accepts 01 but accepts 10 if and only if M accepts w .
- 5.10 Refer to the textbook.
- 5.11 Refer to the textbook.
- 5.12 Reduce from E_{TM} . To determine whether TM M accepts nothing, construct TM N which simulates M on N 's own input w but never writes a blank symbol over a nonblank symbol unless when M accepts.
- 5.13 Reduce from E_{TM} . To determine whether TM M accepts nothing, construct TM N which simulates M on N 's own input w and obviously has no useless state except the one q_{accept} .
- 5.14 Reduce from A_{TM} . To determine whether TM M accepts w , construct TM N which simulates M on N 's own input w but never attempts to move its head left when its head is on the left-most tape cell unless when M accepts.
- 5.15 Let M' be M after modifying all its transitions $\delta(q_i, a) = (q_{\text{accept}}, b, X)$ to $\delta(q_i, a) = (q_{\text{reject}}, b, X)$, and then modifying all $\delta(q_i, a) = (q_j, b, L)$ to $\delta(q_i, a) = (q_{\text{accept}}, b, R)$. The problem is now reduced to checking whether M' , as a TM with stay put instead of left described in Problem 3.13, accepts w . It is easy since $L(M')$ is regular by the solution to Problem 3.13.
- 5.16 Suppose for the sake of contradiction that BB is computable. Then obviously there exists a TM M having k states (let k be sufficiently large), which writes $BB(n) + 1$ 1s on the tape, when given input $\langle n \rangle$. Further, using M we can construct a series of TMs M_n having exactly $k + n/2$ states for large n , which writes $BB(n) + 1$ 1s on the tape when started with a blank tape. However, then M_{2k} , as a $2k$ -state TM, would write $BB(2k) + 1$ 1s. Absurd.
- 5.17 Obviously, in this case a PCP instance P always has a match unless all dominos in P have longer top strings, or they all have longer bottom strings.
- 5.18 Reduce from PCP , since any string with finite alphabet can be encoded to a binary one.
- 5.19 Trivial.
- 5.20 We can encode any language to the one over the unary alphabet.
- 5.21 The hint in the textbook is sufficient.
- 5.22 \Leftarrow : Trivial.
 \Rightarrow : Let $f(x) = \langle M, x \rangle$, where TM M is A 's recognizer.
- 5.23 \Leftarrow : Trivial, since 0^*1^* is surely decidable.
 \Rightarrow : Suppose there is an A 's decider, then f defined as follows is computable.
- $$f(x) = \begin{cases} 01, & x \in A \\ 10, & x \notin A \end{cases}$$
- 5.24 It immediately follows from $A_{\text{TM}}, \overline{A_{\text{TM}}} \leq_m J$.
- 5.25 $\overline{E_{\text{TM}}} \leq_m A_{\text{TM}}$ by Problem 5.22 and it is well known that $A_{\text{TM}} \leq_m E_{\text{TM}}$. By the way, it is not difficult to construct an undecidable language B such that $B =_m \overline{B}$.
- 5.26 The idea is the same as how we deal with A_{LBA} and E_{LBA} .
- 5.27 Prove that $A_{\text{TM}} \leq_m E_{2\text{DIM-DFA}} \leq_m EQ_{2\text{DIM-DFA}}$, in which the former reduction can be done by computation history method.

*5.28 Refer to the textbook.

5.29 The case P is not nontrivial is trivial. As for the second condition, let $P = \{\langle M \rangle \mid \text{TM } M \text{ has 100 states}\}$.

5.30 Trivial.

5.31 Let M be a TM which on input $\langle x \rangle$ ($x \in \mathbb{Z}^+$) calculates $x, f(x), f(f(x)), \dots$ until it finds some $f^{(n)}(x) = 1$, and then accepts. Let N be a TM uses H to calculate whether $\langle M, x \rangle \in A_{\text{TM}}$ for $x = 1, 2, \dots$ in order, until it finds some $\langle M, x \rangle \notin A_{\text{TM}}$, and then accepts. We have the positive answer to the $3x + 1$ problem if and only if $\langle N, 0 \rangle \notin A_{\text{TM}}$.

5.32 a. As hinted, reduce from PCP .

b. Reduce from $OVERLAP_{\text{CFG}}$ by constructing a grammar whose rules are G 's and H 's rules and $S \rightarrow S_G \$ \mid S_H \$ \$$, where S_G and S_H are G 's and H 's start variables.

5.33 Refer to the proof of undecidability of ALL_{CFG} . Let $w = \#C_1\#C_3\#C_5\#\dots\#C_6^R\#C_4^R\#C_2^R\#$.

5.34 Reduce from A_{TM} . To determine whether TM N accepts w , construct TM M which simulates N on M 's own input w but never modifies the portion of the tape that contains the input w unless when N accepts.

5.35 a. Just enumerate w .

b. Reduce from ALL_{CFG} . In order to determine whether $L(G) = \Sigma^*$, construct a grammar whose rules are G 's rules and $S \rightarrow S_G \mid T$; $T \rightarrow aT \mid \epsilon$ ($a \in \Sigma$), where S_G are G 's start variable.

*5.36 See <https://cstheory.stackexchange.com/q/39407/46760> for two different solutions.

6 Advanced Topics in Computability Theory

6.6 Let $M = P_{\langle N \rangle}$ and N print $q(\langle N \rangle) = \langle M \rangle$.

6.7 A TM that always loops.

*6.8 Suppose for the sake of contradiction that f is a reduction from EQ_{TM} to $\overline{EQ_{TM}}$. It is easy to generalize the fixed-point version of the recursion theorem to find $f(\langle M, N \rangle) = \langle M', N' \rangle$ such that M, N simulate M', N' respectively. Then $\langle M, N \rangle \in EQ_{TM} \iff \langle M', N' \rangle \in \overline{EQ_{TM}} \iff \langle M, N \rangle \in \overline{EQ_{TM}}$. Absurd.

6.9 Refer to the textbook.

6.10 Refer to the textbook.

*6.11 $(\mathbb{R}, =, <)$.

6.12 Refer to the textbook.

6.13 Since \mathbb{Z}_m is finite, any sentence in the language of \mathcal{F}_m can be decided by brute-force checking.

6.14 Let $J = 0A \cup 1B$.

6.15 Let $B = A_{TM^A} = \{\langle M^A, w \rangle \mid M^A \text{ accepts } w\}$. Then apply any classical method used in proving undecidability of A_{TM} .

*6.16 (**Kleene–Post**) For convenience let any language L be a subset of \mathbb{N} instead of Σ^* . Denote $\{0, 1, \dots, m\}$ by $[m]$. Define

$$\mathcal{L}_m(A) = \{L \subseteq \mathbb{N} \mid L \cap [m] = A\} \quad (A \subseteq [m])$$

Let M_0, M_1, M_2, \dots be all possible oracle TMs. We will give two series of families of languages $\mathcal{A}_0 \supseteq \mathcal{A}_1 \supseteq \mathcal{A}_2 \supseteq \dots$ and $\mathcal{B}_0 \supseteq \mathcal{B}_1 \supseteq \mathcal{B}_2 \supseteq \dots$ such that

$$\forall A \in \mathcal{A}_n, B \in \mathcal{B}_n, M_n^A \text{ is not } B\text{'s decider and } M_n^B \text{ is not } A\text{'s decider.}$$

Then taking arbitrary $A \in \mathcal{A} = \bigcap_{n \in \mathbb{N}} \mathcal{A}_n$ and $B \in \mathcal{B} = \bigcap_{n \in \mathbb{N}} \mathcal{B}_n$ we have $A \not\leq_T B$ and $B \not\leq_T A$. We build them by induction. Given $\mathcal{A}_{n-1} = \mathcal{L}_m(X)$ and $\mathcal{B}_{n-1} = \mathcal{L}_m(Y)$, we first find $\mathcal{A}'_n \subseteq \mathcal{A}_{n-1}$ and $\mathcal{B}'_n \subseteq \mathcal{B}_{n-1}$ such that $\forall A \in \mathcal{A}'_n, B \in \mathcal{B}'_n, M_n^A$ is not B 's decider.

- If there is no $A \in \mathcal{A}_{n-1}$ such that M_n^A is a decider, let $\mathcal{A}'_n = \mathcal{A}_{n-1}$ and $\mathcal{B}'_n = \mathcal{B}_{n-1}$.
- Suppose M_n^A is a decider with some $A \in \mathcal{A}_{n-1}$, then there exists an $m' > m$ such that

$$\forall A' \in \mathcal{L}_{m'}(A \cap [m']), m+1 \in L(M_n^A) \iff m+1 \in L(M_n^{A'}).$$

Then, let $\mathcal{A}'_n = \mathcal{L}_{m'}(A \cap [m'])$, $\mathcal{B}'_n = \mathcal{L}_{m'}(Y)$ or $\mathcal{L}_{m'}(Y \cup \{m+1\})$, depending on whether $m+1 \in L(M_n^A)$.

The same method can be also used to find $\mathcal{A}_n \subseteq \mathcal{A}'_n$ and $\mathcal{B}_n \subseteq \mathcal{B}'_n$ such that $\forall A \in \mathcal{A}_n, B \in \mathcal{B}_n, M_n^B$ is not A 's decider.

*6.17 Let

$$\begin{aligned} A &= \{\langle M, w \rangle \mid \text{TM } M \text{ on input } w \text{ halts with 0 on its tape}\} \\ B &= \{\langle M, w \rangle \mid \text{TM } M \text{ on input } w \text{ halts with 1 on its tape}\} \end{aligned}$$

If there is a C 's decider N , we can construct TM M which on input w first run N on $\langle M, w \rangle$ to know that M would not halt with $x \in \{0, 1\}$ on M 's tape, and then violates it.

6.18 Suppose $L(M) \neq L(N)$, we can enumerate x to find one such that $\langle M, x \rangle \in A_{TM} \oplus \langle N, x \rangle \in A_{TM}$ holds, where \oplus means exclusive or.

6.19 $|\{L(M^A) \mid M^A \text{ is an oracle TM}\}| \leq |\{M^A \mid M^A \text{ is an oracle TM}\}| \leq \aleph_0 < 2^{\aleph_0} = |\{L \mid L \text{ is a language}\}|$.

- 6.20 Let M be PCP 's recognizer. Then check whether $\langle M, \langle P \rangle \rangle \in A_{\text{TM}}$ to know if instance P has a match.
- 6.21 Since $K(x) \leq |x| + c$, we can check all possible minimal description $\langle M, w \rangle$ to see if M on input w halts with x on its tape by simulating M , where “possible” means that $|\langle M, w \rangle| < |x| + c$ and $\langle M, w \rangle \in A_{\text{TM}}$.
- 6.22 Trivial.
- 6.23 Reduce from Problem 6.24.
- 6.24 Reduce from Problem 6.25.
- 6.25 If not, there is an enumerator E which would print infinite many incompressible strings one by one: s_1, s_2, \dots . By using E we can construct TM M , which prints an incompressible string s_i , such that $|s_i| > |\langle M, 0 \rangle|$, on its tape. Then we have $K(s_i) \leq |\langle M, 0 \rangle| < |s_i|$. Absurd.
- *6.26 Suppose for the sake of contradiction that $K(xy) \leq K(x) + K(y) + c$ always holds. Define

$$f_n = \sum_{|x|=n} 2^{-K(x)}.$$

Then $K(xy) \leq K(x) + K(y) + c \implies f_{n+m} \geq 2^{-c} f_n f_m \implies f_{kn} \geq (2^{-c} f_n)^k$. On the other hand, Corollary 6.30 implies $f_n \leq n + 1$. Therefore,

$$f_n \leq 2^c \sqrt[k]{f_{kn}} \leq 2^c \sqrt[k]{kn + 1}.$$

Letting $k \rightarrow +\infty$ we obtain that $f_n \leq 2^c$ for all n . However, there is a TM M which on input $\langle p, q \rangle$ ($p, q \in \mathbb{N}$ and $q < 2^{2^p}$) halts with $r(p, q)$, a 2^p -bits binary representation of q , on its tape. So $K(r(p, q)) \leq 2 \log_2 p + \log_2 q + d$ with some constant d . Then, if $n = 2^p$ for some large p ,

$$f_n = \sum_{|x|=n} 2^{-K(x)} \geq \sum_{q < 2^n} 2^{-K(r(p, q))} \geq 2^{-2 \log_2 p - d} \sum_{q < 2^n} \frac{1}{q} \geq \frac{n}{2^d (\log_2 n)^2},$$

which apparently contradicts with $f_n \leq 2^c$.

6.27 Show that $\overline{HALT_{\text{TM}}} \leq_m S, \bar{S}$.

- 6.28
- a. $x = 0 \iff \forall y, x + y = y$
 - b. $x = 1 \iff \forall y, y = 0 \wedge x + y = 1$
 - c. $x = y \iff \forall z, z = 0 \wedge x + z = y$
 - d. $x < y \iff \exists z, \neg(z = 0) \wedge x + z = y$

7 Time Complexity

7.13 $a^b = (a^{\lfloor b/2 \rfloor})^2 \cdot a^{b \bmod 2}$, where $a, b \in \mathbb{N}$.

7.14 $q^t = (q^{\lfloor t/2 \rfloor})^2 \cdot q^{t \bmod 2}$, where $q \in S_k$ and $t \in \mathbb{N}$.

7.15 The hint in the textbook is sufficient.

7.16 Refer to the textbook.

7.17 Use dynamic programming. Denote $dp[i][j] = \mathbf{1}\{\langle \{x_1, \dots, x_i\}, j \rangle \in SUBSET-SUM\}$, where $\alpha \implies \mathbf{1}\{\alpha\} = 1$ and $\neg\alpha \implies \mathbf{1}\{\alpha\} = 0$.

7.18 First $A \in P = NP$. Then since there exist $x \in A$ and $y \notin A$, for an arbitrary language $B \in NP = P$, f defined as follows is polynomial time computable.

$$f(w) = \begin{cases} x, & w \in B \\ y, & w \notin B \end{cases}$$

*7.19 Let the certificate of $q \in \mathbb{P}$ consist of

- $g \in \mathbb{Z}_m^*$ such that $g^{m-1} = 1$ and $g^{(m-1)/q} \neq 1$ for all prime $q \mid m-1$,
- the standard factorization of $m-1 = \prod q_i^{r_i}$,
- certificates of $q_i \in \mathbb{P}$.

7.20 It follows from the result of Problem 7.18.

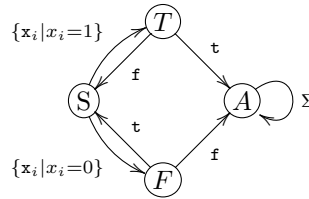
7.21 a. Modify the proof of $PATH \in P$.

b. It is in NP evidently. For the other side, reduce from $UHAMPATH$ by setting k to the amount of nodes in G minus 1.

7.22 It is in NP evidently. For the other side, reduce from SAT . In order to determine whether $\langle \phi \rangle \in SAT$, construct $\phi' = \phi \wedge (z \vee \bar{z})$.

7.23 Refer to the textbook.

*7.36 It is in NP evidently. For the other side, reduce from $3SAT$. If there is no limitation on Σ , let the following DFA correspond to an assignment to variables x_1, x_2, \dots, x_n in 3cnf-formula ϕ . You may need some extra states and transitions.



Once you solved the problem without regard to the limitation on Σ , based on your solution, consider how to build a reduction where $\Sigma = \{0, 1\}$.

7.37 Using the computation history as certificate we easily obtain $U \in NP$. And it is easy to show that $3SAT \leq_P U$, by designing an NTM M , which accepts $\langle \phi \rangle$ in polynomial time on at least one branch if and only if $\langle \phi \rangle \in 3SAT$.

*7.38 Let $\phi(x/t)$ denote the Boolean formula ϕ after replacing every existence of x in ϕ with t . Suppose there are n variables x_1, \dots, x_n in ϕ . $\langle \phi \rangle \in SAT \implies \langle \phi(x_1/0) \rangle \in SAT$ or $\langle \phi(x_1/1) \rangle \in SAT$, so we can directly assign 0 or 1 to x_1 . Recursively assigning x_2, \dots, x_n in this way we have done.

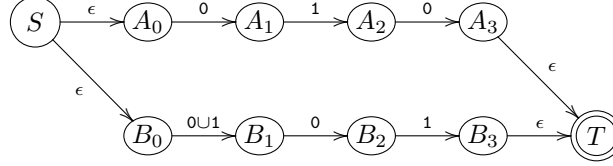
*7.39 Construct language $L = \{\langle n, x, y \rangle \mid n \text{ has a nontrivial factor in interval } [x, y]\}$, which is apparently in $NP = P$. Then refer to the solution to Problem 7.38.

*7.40 Refer to the textbook.

7.41 Trivial.

*7.42 For b), note that the complement graph of G induces an equivalence relation \sim on Q ($[q]$ is exactly the equivalence class under \sim including q), which has much to do with $\equiv_{L(M)}$ defined in Problem 1.51.

7.43 Here is a sample for $\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee \overline{x_3})$.



And $\langle \phi \rangle \notin SAT \iff$ the equivalent minimal NFA is a trivial one.

*7.44 It is a classical problem. See <https://en.wikipedia.org/wiki/2-satisfiability> or refer to Problem 7.51.

7.45 Trivial.

7.46 Since P is closed under complement, we only need to show that $\overline{MIN-FORMULA} \in NP = P$. It is easy to do because $SAT \in NP = P$, and then the certificate for $\langle \phi \rangle \in \overline{MIN-FORMULA}$ can be $\langle \phi' \rangle$ such that $|\phi'| < |\phi|$ and they are equivalent.

7.47 $Z = \{\langle G_1, k_1, G_2, k_2 \rangle \mid \langle G_1, k_1 \rangle \in CLIQUE\} - \{\langle G_1, k_1, G_2, k_2 \rangle \mid \langle G_2, k_2 \rangle \in CLIQUE\}$.

*7.48 Obviously $MAX-CLIQUE \in DP$. Suppose there is a graph $G = (V, E)$ with $V = \{v_1, v_2, \dots, v_n\}$. Denote $[n] = \{1, 2, \dots, n\}$. Let $G_+ = (V_+, E_+)$, where

$$\begin{cases} V_+ = \{(i, v_j) \mid i \in [n+1], v_j \in V\} \cup \{(i, \alpha) \mid i \in [n+1] - [k]\} \\ E_+ = \{\{(i, u), (j, v)\} \mid i \neq j, \{u, v\} \in E\} \cup \{\{(i, \alpha), (j, v)\} \mid i \neq j, v \in V \cup \{\alpha\}\} \end{cases}$$

Then $\langle G, k \rangle \in CLIQUE \iff \langle G_+, n+1 \rangle \in MAX-CLIQUE$. Let G_- consist of G_+ and a n -clique (i.e., complete graph K_n). Then $\langle G, k \rangle \notin CLIQUE \iff \langle G_-, n \rangle \in MAX-CLIQUE$. Try to build a reduction by taking advantage of G_+ and G_- .

*7.49 *Need a solution.*

*7.50 $\overline{EQ_{SF-REX}} = \{\langle R, S \rangle \mid \exists c, c \in L(R) \oplus c \in L(S)\}$. Determining whether $c \in L(R)$ can be achieved in (deterministic) polynomial time by constructing a corresponding NFA. Note that any $c \in L(R)$ for a star-free REX satisfies $|c| \leq \text{Poly}(|R|)$.

*7.51 Trivial.

*7.52 *Need a solution.*

*7.53 *Need a solution.*

7.54 *Need a solution.*

8 Space Complexity

- 8.8 Refer to Example 8.4.
- 8.9 Build an NTM which nondeterministically guesses a ladder s, s_2, \dots, s_k and verifies whether $s_k = t$, where k is obviously bounded in $2^{\mathcal{O}(|s|)}$. Then $LADDER_{\text{DFA}} \in \text{NPSpace} = \text{PSpace}$ follows.
- 8.10 It can be reduced to *FORMULA-GAME* directly.
- 8.11 If so, *SAT* is PSPACE-hard, thus it is PSPACE-complete.
- 8.12 It is because $\phi_{c_1, c_2, 1}$ in the proof of Theorem 8.9 can be written in conjunctive normal form, as we see in the proof of Theorem 7.37 (Cook–Levin theorem).
- 8.13 Reduce from *TQBF*. Clearly we can construct a $g(n) = n^{100} + 10^{100}$ space TM M deciding *TQBF*. Build mapping $f(\langle \phi \rangle) = \langle M', \langle \phi \rangle \$ 0^{g(|\phi|)} \rangle$, where LBA M' does almost the same as M does. On the other side, obviously $A_{\text{LBA}} \in \text{PSpace}$.
- *8.14 For any $\langle G, c, m, h \rangle$, here is a polynomial time algorithm. Let $C[i][j] = +1$ or -1 stand for that we can determine $\langle G, i, j, h \rangle \in \text{HAPPY-CAT}$ or $\langle G, i, j, h \rangle \in \text{HAPPY-MOUSE}$. Similarly let $M[i][j] = +1$ or -1 stand for that we can determine $\langle G, i, j, h \rangle \in \text{HAPPY-MOUSE}'$ or $\langle G, i, j, h \rangle \in \text{HAPPY-CAT}'$, where *HAPPY-CAT'* is defined like *HAPPY-CAT*, except that Mouse moves first. Now we set $M[i][i] = -1$ and $C[i][h] = -1$ for all $i \in G$. Then, repeatedly assign new value to M and C according to the following rules until we get nothing more from the rules.

$$\begin{cases} C[i][j] = -1, & \forall i' \in N(i), M[i'][j] = +1 \\ C[i][j] = +1, & \exists i' \in N(i), M[i'][j] = -1 \\ M[i][j] = -1, & \forall j' \in N(j), C[i][j'] = +1 \\ M[i][j] = +1, & \exists j' \in N(j), C[i][j'] = -1 \end{cases}$$

where $N(v)$ stands for the collection of all nodes adjacent to v in G . After this calculation, we claim that $\langle G, c, m, h \rangle \in \text{HAPPY-CAT} \iff C[c][m] = +1$. The proof of correctness of the algorithm is not evident but also not hard.

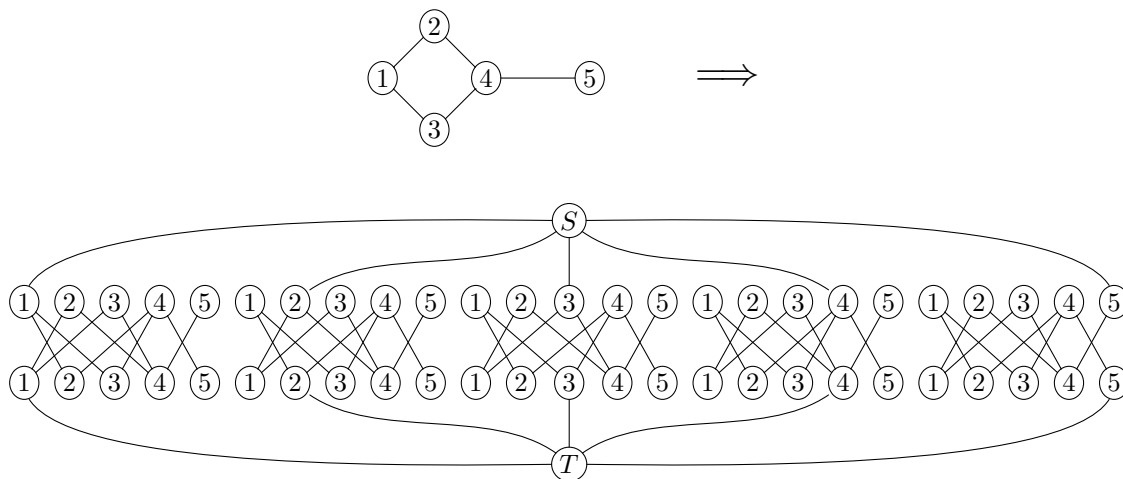
- 8.15 *Need a solution.*
- 8.16 *Need a solution.*
- 8.17 A left-to-right scan with memorizing the amount of non-matched (s is enough).
- *8.18 Let h be a homomorphism that maps brackets to parentheses, e.g., $h([] []) = () ()$. Denote the substring of w from the i -th character to the j -th character as $w_{[i,j]}$ and $w_i = w_{[i,i]}$. Then, $w \in B$ if and only if
- $h(w) \in A$, where A is defined in Problem 8.17.
 - w_i matches w_j whenever $h(w_{[i+1, j-1]}) \in A$, for all $1 \leq i \leq j \leq |w|$.
- *8.19 It is a classical problem. See <https://en.wikipedia.org/wiki/Nim>.
- 8.20 *Need a solution.*
- 8.21 *Need a solution.*
- 8.22 *Need a solution.*
- *8.23 Suppose $G = (V, E)$ has n nodes named v_0, v_1, \dots, v_{n-1} and define $v_k = v_{k \bmod n}$. Consider the sequence consisting of *directed* edges in G , by seeing an undirected edge as two directed ones: w_0, w_1, w_2, \dots , where
- $$w_{m+1} = (v_j, v_k), \text{ if } w_m = (v_i, v_j) \text{ and } \{v_j, v_{i+1}\}, \{v_j, v_{i+2}\}, \dots, \{v_j, v_{k-1}\} \notin E.$$

G contains no cycle if and only if for every $w_0 = (u, v)$, the first w_r in the sequence such that $w_r = (\cdot, u)$ is (v, u) . You may assume G is connected without loss of generality and then use mathematical induction on n to prove this. Note that the sequence w_0, w_1, w_2, \dots would traverse every edges in G , i.e., perform depth-first search, if G is a tree.

*8.24 See <https://math.stackexchange.com/a/79031/134950>.

8.25 Surely $BIPARTITE \in \text{coNL} = \text{NL}$, since we can guess a odd cycle, whose length is no more than n , to verify $\langle G \rangle \notin BIPARTITE$.

8.26 Here is a sample which shows how to do reduction.



8.27 Reduce from *PATH*. In order to determine whether a path from s to t exists in graph $G = (V, E)$, we add some edges (v, s) and (t, v) for all $v \in V$ to obtain a modified graph G' . Then a path exists if and only if G' is strongly connected. On the other side, it is obviously in NL.