第二次助教习题课

理论计算机科学基础

柴文健

中国科学院软件研究所

2025 年 6 月 20 日

1. 第 7 章：时间复杂性
2. 第 8 章：空间复杂性
3. 第 9 章：难解性
4. 第 10 章：复杂性理论高级专题

1. 第 7 章：时间复杂性
2. 第 8 章：空间复杂性
3. 第 9 章：难解性
4. 第 10 章：复杂性理论高级专题

第 7 章习题

- 7.1 下面各项是真是假？

$$(a)\ 2n = O(n) \quad (b)\ n^2 = O(n)$$
$$(e)\ 3^n = 2^{O(n)} \quad (f)\ 2^{2^n} = O(2^{2^n})$$

- (a) 正确。存在正整数 $c = 2$ 和 $n_0 = 1$，对所有 $n \geq n_0$，有 $2n \leq cn$；故 $2n = O(n)$。
- (b) 错误。对任意正整数 c 和 n_0，存在 $n = max\{n_0, c+1\} \geq n_0$，使得 $n^2 \geq (c+1)n > cn$；故 $n^2 \neq O(n)$。
- (e) 正确。存在正整数 $c = log_2 3$ 和 $n_0 = 1$，对所有 $n \geq n_0$，有 $log_2 3^n \leq cn$；故 $log_2 3^n = O(n)$，即 $3^n = 2^{O(n)}$。
- (f) 正确。存在正整数 $c = 1$ 和 $n_0 = 1$，对所有 $n \geq n_0$，有 $2^{2^n} \leq c2^{2^n}$；故 $2^{2^n} = O(2^{2^n})$。

第 7 章习题

- 7.2 下面各项是真是假？

$$(a)\ n = o(2n) \quad (b)\ 2n = o(n^2)$$

- (a) 错误。$lim_{n\to\infty}\frac{n}{2n} = \frac{1}{2} \neq 0$；故 $n \neq o(2n)$。或存在正整数 $c = \frac{1}{2}$，对任意 n_0，存在 $n = n_0 \geq n_0$，使得 $n \geq c(2n)$；故 $n \neq o(2n)$。
- (b) 正确。$lim_{n\to\infty}\frac{2n}{n^2} = 0$；故 $2n = o(n^2)$。或对任意实数 $c > 0$，存在 $n_0 = \lfloor\frac{2}{c}\rfloor + 1$，使得对所有 $n \geq n_0$，有 $2n = cn\frac{2}{c} < cn(\lfloor\frac{2}{c}\rfloor + 1) \leq cn^2$；故 $2n = o(n^2)$。

第 7 章习题

- 7.4 对于字符串 $w = baba$ 和下面的文法 CFG G，试填写定理 7.14 中识别上下文无关语言的多项式时间算法中所描述的表：

$$S \rightarrow RT$$
$$R \rightarrow TR \mid a$$
$$T \rightarrow TR \mid b$$

- 表格如下所示。由 S 在 $table(1, n)$ 中，故 $w = baba \in L(G)$。

		j			
		1	2	3	4
i	1	T	R, T	S	R, S, T
	2		R	S	S
	3			T	R, T
	4				R

第 7 章习题

- 7.6 证明 P 在并、连接和补运算下封闭。
- 给定 $L_1, L_2 \in$P，L_1, L_2 的多项式时间算法分别为 M_1, M_2。构造 $L_1 \cup L_2$ 的一个多项式时间算法 M，运行如下：
  $M=$ “对输入 w：
  1. 在 w 上运行 M_1，如果 M_1 接受则接受；
  2. 在 w 上运行 M_2，如果 M_2 接受则接受，否则拒绝。”
- 若 M 接受 w，则 $w \in L_1 \cup (\overline{L_1} \cap L_2) = L_1 \cup L_2$；若 M 拒绝 w，则 $w \in \overline{L_1} \cap \overline{L_2} = \overline{L_1 \cup L_2}$。故 M 接受 $w \iff w \in L_1 \cup L_2$。
- 由于步骤 1 和步骤 2 都在输入长度的多项式时间内运行完成，M 也在输入长度的多项式时间内完成。
- 即判定 $L_1 \cup L_2$ 有多项式时间算法，故 P 在并下封闭。

第 7 章习题

- 7.6 证明 P 在并、**连接**和补运算下封闭。
- 给定 $L_1, L_2 \in$P，L_1, L_2 的多项式时间算法分别为 M_1, M_2。构造 $L_1 \circ L_2$ 的一个多项式时间算法 M，运行如下：
  M= "对长度为 n 的输入 w：
  1. 令 $i = 0, 1, ..., n$ 依次递增，执行步骤 2；
  2. 记 w 的前 i 位为 w_1、w 的后 $n - i$ 位为 w_2，在 $\langle w_1, w_2 \rangle$ 上运行 M'；
  3. 如果 M' 对于某个 i 接受，则接受；否则 M' 对于所有的 $i = 0, 1, ..., n$ 都拒绝，则拒绝。"
- 其中子程序 M' 运行如下：
  M'= "对输入 $\langle w_1, w_2 \rangle$：
  1. 在 w_1 上运行 M_1，如果 M_1 拒绝则拒绝；
  2. 在 w_2 上运行 M_2，如果 M_2 拒绝则拒绝，否则接受。"

第 7 章习题

- 7.6 证明 P 在并、**连接**和补运算下封闭。
- （续）类似地，M' 接受 $\langle w_1, w_2 \rangle \iff w_1 \in L_1$ 且 $w_2 \in L_2$。而 M 的步骤 1 和步骤 2 遍历了 w 是两个子串连接而成的所有情况，因而 M 接受 $w \iff w \in L_1 \circ L_2$。
- 由于 M' 的步骤 1 和步骤 2 都在各自输入长度的多项式时间内运行完成，M' 也在输入长度的多项式时间内完成（若 $f(n)$ 和 $g(n)$ 是多项式，则 $\forall 0 \leq i \leq n, f(i) + g(n-i)$ 也是多项式）。M 至多运行 M' n 次，进而 M 也在输入长度的多项式时间内完成。
- 即判定 $L_1 \circ L_2$ 有多项式时间算法，故 P 在连接下封闭。

第 7 章习题

- 7.6 证明 P 在并、连接和**补**运算下封闭。
- 给定 $L \in P$，L 的一个多项式时间算法为 M。构造 $\overline{L}$ 的一个多项式时间算法 M'，运行如下：
  $M'=$ “对输入 w：
  1. 在 w 上运行 M，如果 M 接受则拒绝，如果 M 拒绝则接受。”
- M' 接受 $w \iff M$ 拒绝 $w \iff w \in \overline{L}$。
- 由于步骤 1 在输入长度的多项式时间内运行完成，M' 也在输入长度的多项式时间内完成。
- 即判定 $\overline{L}$ 有多项式时间算法，故 P 在补下封闭。

第 7 章习题

- 7.7 证明 NP 在并和连接运算下封闭。
- 方法 1：给定 $L_1, L_2 \in$NP，L_1, L_2 的多项式时间验证机分别为 V_1, V_2。构造 $L_1 \cup L_2$ 的多项式时间验证机 V，运行如下：
  $V=$ “对输入 $\langle w, c\rangle$，其中 c 是证书：
  1. 在 $\langle w, c\rangle$ 上运行 V_1，如果 V_1 接受则接受；
  2. 在 $\langle w, c\rangle$ 上运行 V_2，如果 V_2 接受则接受，否则拒绝。”
- 若 V 接受 $\langle w, c\rangle$，则可以通过 c 验证 $w \in L_1 \cup (\overline{L_1} \cap L_2) = L_1 \cup L_2$；反之亦然。因而 V 是 $L_1 \cup L_2$ 的验证机。
- 由于步骤 1 和步骤 2 都在 w 长度的多项式时间内运行完成，V 也在 w 长度的多项式时间内完成。
- 即验证 $L_1 \cup L_2$ 有多项式时间算法，故 NP 在并下封闭。

第 7 章习题

- 7.7 证明 NP 在并和连接运算下封闭。
- 方法 2：给定 $L_1, L_2 \in$NP，L_1, L_2 分别被多项式时间 NTM N_1, N_2 判定。构造多项式时间判定 $L_1 \cup L_2$ 的 NTM N，运行如下：
  $N=$ "对输入 w：
  1. 非确定地选择 $N_i, i = 1, 2$;
  2. 在 w 上运行 N_i;
  3. 如果 N_i 接受，则接受；否则拒绝。"
- 若 N 接受 w，则存在某个计算分支被 N 接受，即 $w \in L_1 \cup L_2$；若 N 拒绝 w，则所有计算分支都被 N 拒绝，即 $w \in \overline{L_1} \cap \overline{L_2} = \overline{L_1 \cup L_2}$。因而 N 接受 $w \iff w \in L_1 \cup L_2$。
- 由于步骤 2 在 w 长度的多项式时间内运行完成、步骤 1 和步骤 3 也在常数时间内完成选择或判断，N 在 w 长度的多项式时间内完成。
- 即判定 $L_1 \cup L_2$ 有非确定的多项式时间算法，故 NP 在并下封闭。

第 7 章习题

- 7.7 证明 NP 在并和**连接**运算下封闭。
- 给定 $L_1, L_2 \in$NP，L_1, L_2 分别被多项式时间 NTM N_1, N_2 判定。构造多项式时间判定 $L_1 \circ L_2$ 的 NTM N，运行如下：
  N= "对长度为 n 的输入 w：
  1. 非确定地选择 $i = 0, 1, ..., n$；
  2. 记 w 的前 i 位为 w_1、w 的后 $n-i$ 位为 w_2，在 $\langle w_1, w_2 \rangle$ 上运行 N'；
  3. 如果 N' 接受，则接受；否则拒绝。"
- 其中子程序 N' 运行如下：
  N'= "对输入 $\langle w_1, w_2 \rangle$：
  1. 在 w_1 上运行 N_1，如果 N_1 拒绝则拒绝；
  2. 在 w_2 上运行 N_2，如果 M_2 拒绝则拒绝，否则接受。"

第 7 章习题

- 7.7 证明 NP 在并和**连接**运算下封闭。
- （续）类似地，N' 接受 $\langle w_1, w_2\rangle \iff w_1 \in L_1$ 且 $w_2 \in L_2$。若 N 接受 w，则存在某个计算分支被 N 接受，即 $w \in L_1 \circ L_2$；若 N 拒绝 w，则所有计算分支都被 N 拒绝，又由 N 的步骤 1 和步骤 2 选择的是将 w 拆成两个子串的所有情况，故 $w \notin L_1 \circ L_2$。因而 N 接受 $w \iff w \in L_1 \circ L_2$。
- 由于 N' 的步骤 1 和步骤 2 都在各自输入长度的多项式时间内运行完成，N' 也在输入长度的多项式时间内完成。即 N 的步骤 2 在 w 长度的多项式时间内运行完成，又 N 的步骤 1 和步骤 3 也在常数时间内完成选择或判断，故 N 在 w 长度的多项式时间内完成。
- 即判定 $L_1 \circ L_2$ 有非确定的多项式时间算法，故 NP 在连接下封闭。

第 7 章习题

- 7.8 令 $CONNECTED = \{\langle G\rangle | G$ 是连通的无向图$\}$。分析 3.3.2 节给出的算法，证明此语言属于 P。
- 记 $G = \langle V, E\rangle$，$|V| = n$。
- 步骤 1 在常数时间内完成选择。
- 步骤 2 和步骤 3 的每轮迭代至少增加一个标记点（最后一轮发现无法再增加标记点便停止），迭代轮数为 $O(n)$；每轮迭代对 G 中每个顶点（总数为 n）进行判断，判断“其是否与另一个已被标记的顶点通过边相连”可在 n 的多项式时间内完成（具体情况与存储图的数据结构有关）。因而步骤 2 和步骤 3 在 n 的多项式时间内完成。
- 步骤 4 扫描 G 的所有顶点，至多扫描 n 次即决定接受与否。
- 综上，上述算法在 n（即输入长度）的多项式时间内完成。即使用 3.3.2 节给出的多项式算法可以判定 $CONNECTED$，故 $CONNECTED \in$P。

第 7 章习题

- 7.9 无向图中的**三角形**(triangle) 是一个 3-团。证明 $TRIANGLE \in P$，其中 $TRIANGLE = \{\langle G \rangle | G$ 包含一个三角形$\}$。
- 如下给出判定 $TRIANGLE$ 的图灵机 M_{TRI}：
  M_{TRI}= "对输入 $\langle G \rangle$，其中 $G = \langle V, E \rangle$ 是无向图：
  1. 按字典序遍历 $u, v, w \in V$，其中按字典序有 $u < v < w$；
  2. 如果 $(u, v), (u, w), (v, w)$ 都在 E 中，则接受；
  3. 遍历完所有满足条件的 u, v, w 仍不接受，则拒绝。"
- 若 M_{TRI} 接受 $\langle G \rangle$，则无向图 G 中包含以接受时的 u, v, w 为顶点的三角形；若无向图 G 中包含一个三角形（不妨设其字典序最小）、其顶点为 s_1, s_2, s_3（按字典序 $s_1 < s_2 < s_3$），则 M_{TRI} 的步骤 1 和步骤 2 在 $u = s_1, v = s_2, w = s_3$ 时接受 $\langle G \rangle$。即 M_{TRI} 接受 $\langle G \rangle \iff G \in TRIANGLE$。
- 步骤 1 和步骤 2 至多遍历 $|V|^3$ 次，判断 3 条边是否在 E 中可在 $|E|$ 的多项式时间内完成。步骤 3 可在常数时间内完成判断。故上述图灵机在输入长度的多项式时间内接受/拒绝。
- 即判定 $TRIANGLE$ 有多项式时间算法，故 $TRIANGLE \in$P。

第 7 章习题

- 7.10 证明 ALL_{DFA} 属于 P。
- 方法 1：如下给出判定 ALL_{DFA} 的图灵机 M_{A_D}：
  M_{A_D} = "对于输入 $\langle A\rangle$，其中 A 是一个 DFA：
  1. 用练习 1.14(a) 中交换接受状态与非接受状态的方法构造满足 $L(B)=\overline{L(A)}$ 的 DFA B；
  2. 在输入 $\langle B\rangle$ 上运行定理 4.4 中判定 E_{DFA} 的图灵机 T；
  3. 如果 T 接受，则接受；如果 T 拒绝，则拒绝。"
- M_{A_D} 接受 $\langle A\rangle \iff T$ 接受 $\langle B\rangle \iff L(B)=\emptyset \iff L(A)=\Sigma^*$。
- M_{A_D} 的步骤 1 可在 $|Q_A|$ 的多项式时间内交换接受状态和非接受状态，因而构造可在输入长度的多项式时间内完成、且 $\langle B\rangle$ 的长度不超过输入长度的多项式。T 的步骤 1 可在常数时间内完成标记；类似 7.8 中的分析，T 的步骤 2 和步骤 3 可在 $|Q_B|$ 的多项式时间内结束标记过程；步骤 4 扫描所有接受状态，至多扫描 $|Q_B|$ 次即决定接受与否。因此 M_{A_D} 的步骤 2 在 $|Q_A|$ 的多项式时间内运行完成；又 M_{A_D} 的步骤 3 可在常数时间内完成判断，故上述图灵机在输入长度的多项式时间内接受/拒绝。
- 即判定 ALL_{DFA} 有多项式时间算法，故 $ALL_{DFA}\in$P。

第 7 章习题

- 7.10 证明 ALL_{DFA} 属于 P。
- 方法 2：如下给出判定 ALL_{DFA} 的图灵机 M_{A_D}：
  M_{A_D} = "对于输入 $\langle A\rangle$，其中 A 是一个 DFA：
  1. 标记 DFA A 的起始状态；
  2. 重复第 3 步，直到所有未被标记的状态都不能被标记；
  3. 对于一个状态，如果有一个到达它的转移是从某个被标记的状态出发的，则将其标记；
  4. 如果所有被标记的状态均是接受状态，则接受；否则拒绝。"
- M_{A_D} 拒绝 $\langle A\rangle \iff \exists q \in Q_A, q$ 被标记但 $q \notin F_A \overset{\delta^*(q_0,w)=q}{\iff}$ $\exists w \in \Sigma^*, w \notin L(A) \iff L(A) \neq \Sigma^*$。
- M_{A_D} 的步骤 1 可在常数时间内完成标记；类似方法 1 中的分析，M_{A_D} 的步骤 2 和步骤 3 可在 $|Q_A|$ 的多项式时间内结束标记过程；步骤 4 扫描所有被标记状态，可在 $|Q_A|$ 的多项式时间内决定接受与否。故上述图灵机在输入长度的多项式时间内接受/拒绝。
- 即判定 ALL_{DFA} 有多项式时间算法，故 $ALL_{DFA} \in$P。

第 7 章习题

- 7.11(a) 证明 $EQ_{DFA} \in P$。
- 定理 4.5 中的图灵机 F 即可判定 EQ_{DFA}。
- F 接受 $\langle A, B\rangle \iff T$ 接受 $\langle C\rangle \iff L(C) = \emptyset \iff L(A) \cap \overline{L(B)} = \emptyset$ 且 $\overline{L(A)} \cap L(B) = \emptyset \iff L(A) \subseteq L(B)$ 且 $L(B) \subseteq L(A) \iff L(A) = L(B)$。
- 构造识别语言补集的 DFA 可在 $|Q|$ 的多项式时间内完成（7.10 的方法 1）；可借助状态集的笛卡尔积构造识别语言交集/并集的 DFA，同样可在 $|Q|$ 的多项式时间内完成。因此 F 的步骤 1 可在输入长度的多项式时间内完成构造、且 $\langle C\rangle$ 的长度不超过输入长度的多项式。F 的步骤 2 在 $|Q_C|$ 的多项式时间内运行完成（7.10 的方法 1）。又 F 的步骤 3 可在常数时间内完成判断，故图灵机 F 在输入长度的多项式时间内接受/拒绝。
- 即判定 EQ_{DFA} 有多项式时间算法，故 $EQ_{DFA} \in$P。

第 7 章习题

- 7.11(b) 对语言 A，如果 $A = A^*$，则称 A 是**星闭的**。给出测试一个 DFA 是否识别一个星闭的语言的多项式时间算法。
- 给定一个 DFA $M = (Q, \Sigma, \delta, q_0, F)$，先检查 $q_0 \in F$，否则 $\varepsilon \notin L(M)$，$L(M)$ 必然不是星闭的。
- 方法 1：检查 $L(M)L(M) \subseteq L(M)$。对每一个 $q \in F, q \neq q_0$，构造 DFA $M_q = (Q, \Sigma, \delta, q, F)$；则 $L(M)L(M) \subseteq L(M) \iff \forall q \in F, q \neq q_0, L(M_q) \subseteq L(M)$。
- 检查 $L(M_q) \subseteq L(M)$：构造 $M'_q = (Q', \Sigma, \delta', q_0^q, F')$，其中 $Q' = Q \times Q, \delta'((q_1, q_2), a) = (\delta(q_1, a), \delta(q_2, a)), F' = F \times (Q - F)$，特别地，$q_0^q = (q, q_0)$。则 $L(M'_q) = \emptyset$
  $\iff \forall w \in \Sigma^*, \delta'((q, q_0), w) \notin F \times (Q - F)$
  $\iff \forall w \in \Sigma^*, \delta(q, w) \in F \implies \delta(q_0, w) \in F$
  $\iff \forall w \in \Sigma^*, w \in L(M_q) \implies w \in L(M)$
  $\iff L(M_q) \subseteq L(M)$。

第 7 章习题

- 7.11(b) 对语言 A，如果 $A = A^*$，则称 A 是**星闭的**。给出测试一个 DFA 是否识别一个星闭的语言的多项式时间算法。
- （续）即如下给出判定 $L(M)$ 是否星闭的图灵机 T_{SC}：
  T_{SC} = "对于输入 $\langle M\rangle$，M 是一个 DFA：
  1. $M = (Q, \Sigma, \delta, q_0, F)$，如果 $q_0 \notin F$ 则拒绝；
  2. 对所有 $q \neq q_0 \in F$，构造如上的 M'_q；
  3. 在输入 $\langle M'_q\rangle$ 上运行定理 4.4 中判定 E_{DFA} 的图灵机 T，如果 T 拒绝则拒绝；
  4. 如果对所有 $q \neq q_0 \in F$，T 均接受 $\langle M'_q\rangle$，则接受。
- T_{SC} 接受 $\langle M\rangle \iff \forall q \neq q_0 \in F, L(M_q) \subseteq L(M) \iff L(M)L(M) \subseteq L(M) \overset{\varepsilon\in L(M)}{\iff} L(M)L(M) = L(M) \overset{\varepsilon\in L(M)}{\iff} L(M) = L(M)^*$。
- 步骤 1 可在 $|F|$ 的时间内完成。步骤 2 可在输入长度的多项式时间内完成构造、且 $\langle M'_q\rangle$ 的长度不超过输入长度的多项式。步骤 3 可在 $|\langle M'_q\rangle|$ 的多项式时间内完成运行。步骤 2 和步骤 3 至多构造和运行 $|F|$ 次。步骤 4 可在常数时间内完成判断，故图灵机 T_{SC} 在输入长度的多项式时间内接受/拒绝。
- 即判定 $L(M)$ 是否星闭有多项式时间算法，故该语言属于 P。

第 7 章习题

- 7.11(b) 对语言 A，如果 $A = A^*$，则称 A 是**星闭的**。给出测试一个 DFA 是否识别一个星闭的语言的多项式时间算法。
- 给定一个 DFA $M = (Q, \Sigma, \delta, q_0, F)$，先检查 $q_0 \in F$，否则 $\varepsilon \notin L(M)$，$L(M)$ 必然不是星闭的。
- 方法 2：构造 NFA $M^* = (Q, \Sigma_\varepsilon, \delta^*, q_0, F)$ 使得 $L(M^*) = L(M)^*$；其中 δ^* 在保留 δ 的基础上，对每个 $q \neq q_0 \in F$ 增加 $\delta^*(q, \varepsilon) = q_0$。检查 $L(M^*) \subseteq L(M)$ 即可。
- 检查 $L(M^*) \subseteq L(M)$：构造 NFA $M^{*\prime} = (Q', \Sigma_\varepsilon, \delta', q_0', F')$，其中 $Q' = Q \times Q, q_0' = (q_0, q_0), F' = F \times (Q - F), \delta'((q_1, q_2), a) = (\delta^*(q_1, a), \delta(q_2, a))$，特别地，$\forall q_1 \neq q_0 \in F, \forall q_2 \in Q$，$\delta'((q_1, q_2), \varepsilon) = (q_0, q_2)$。则 $L(M^{*\prime}) = \emptyset$
  $\iff \forall w \in \Sigma^*, \delta'((q_0, q_0), w) \notin F \times (Q - F)$
  $\iff \forall w \in \Sigma^*, \delta^*(q_0, w) \in F \implies \delta(q_0, w) \in F$
  $\iff \forall w \in \Sigma^*, w \in L(M^*) \implies w \in L(M)$
  $\iff L(M^*) \subseteq L(M)$。

第 7 章习题

- 7.11(b) 对语言 A，如果 $A = A^*$，则称 A 是**星闭的**。给出测试一个 DFA 是否识别一个星闭的语言的多项式时间算法。
- （续）即如下给出判定 $L(M)$ 是否星闭的图灵机 T_{SC}：
  T_{SC} = "对于输入 $\langle M\rangle$，M 是一个 DFA：
  1. $M = (Q, \Sigma, \delta, q_0, F)$，如果 $q_0 \notin F$ 则拒绝；
  2. 构造如上的 NFA $M^{*\prime}$；
  3. 在输入 $\langle M^{*\prime}\rangle$ 上运行定理 4.4 中判定 E_{DFA} 的图灵机 T（算法也可判定 NFA 是否接受空语言）；
  4. 如果 T 接受，则接受；如果 T 拒绝，则拒绝。
- T_{SC} 接受 $\langle M\rangle \iff L(M^{*\prime}) = \emptyset \iff L(M)^* = L(M^*) \subseteq L(M) \iff L(M) = L(M)^*$。
- 步骤 1 可在 $|F|$ 的时间内完成。步骤 2 可在输入长度的多项式时间内完成构造、且 $\langle M^{*\prime}\rangle$ 的长度不超过输入长度的多项式。步骤 3 可在 $|\langle M^{*\prime}\rangle|$ 的多项式时间内完成运行。步骤 4 可在常数时间内完成判断，故图灵机 T_{SC} 在输入长度的多项式时间内接受/拒绝。
- 即判定 $L(M)$ 是否星闭有多项式时间算法，故该语言属于 P。

第 7 章习题

- 7.12 若图 G 的结点重新排序后，G 可以变得与 H 完全相同，则称 G 与 H 是**同构的**。令 $ISO = \{\langle G, H\rangle | G$ 和 H 是同构的图$\}$。证明 $ISO \in$NP。
- 方法 1：考虑构造 ISO 的多项式时间验证机 V_{ISO}（证书为 V_G 到 V_H 的映射 π），运行如下：
  V_{ISO} = "对于输入 $\langle G, H, \pi\rangle$，其中 G, H 都是图：
  1. 如果 π 不是双射（单射且满射），则拒绝；
  2. 遍历所有 $v_1, v_2 \in V_G$；
  3. 如果 $(v_1, v_2) \in E_G$ 但 $(\pi(v_1), \pi(v_2)) \notin E_H$，或 $(v_1, v_2) \notin E_G$ 但 $(\pi(v_1), \pi(v_2)) \in E_H$，则拒绝；
  4. 遍历完所有 $v_1, v_2 \in V_G$ 后仍未拒绝，则接受。"
- 步骤 1 中单射（$\forall v_1 \in V_G, \forall v_2 \in V_G$ 且 $v_2 \neq v_1, \pi(v_1) \neq \pi(v_2)$）和满射（$\forall v' \in V_H, \exists v \in V_G, \pi(v) = v'$）的判断可在 $|V_G|$ 和 $|V_H|$ 的多项式时间内完成。步骤 2 和步骤 3 至多遍历 $|V_G|^2$ 次，边是否在 E_G/E_H 中的判断可在 $|E_G|/|E_H|$ 的多项式时间内完成。步骤 4 在常数时间内完成判断，故上述图灵机在输入长度的多项式时间内接受/拒绝。
- 即验证 ISO 有多项式时间算法，故 $ISO \in$NP。

第 7 章习题

- 7.12 若图 G 的结点重新排序后，G 可以变得与 H 完全相同，则称 G 与 H 是**同构的**。令 $ISO = \{\langle G, H\rangle | G$ 和 H 是同构的图$\}$。证明 $ISO \in$NP。
- 方法 2：考虑构造多项式时间判定 ISO 的 NTM N_{ISO}，运行如下：
  $N_{ISO} =$ "对于输入 $\langle G, H\rangle$，其中 G, H 都是图：
  1. 如果 $|V_G| \neq |V_H|$ 或 $|E_G| \neq |E_H|$，则拒绝；
  2. 非确定地选择一个从 V_G 到 V_H 的双射 π；
  3. 在 $\langle G, H, \pi\rangle$ 上运行 N'；
  4. 如果 N' 接受，则接受；否则拒绝。"
- 其中子程序 N' 运行如下：
  $N' =$ "对于输入 $\langle G, H, \pi\rangle$，其中 G, H 是图，$\pi : V_G \to V_H$ 是双射：
  1. 遍历所有 $v_1, v_2 \in V_G$；
  2. 如果 $(v_1, v_2) \in E_G$ 但 $(\pi(v_1), \pi(v_2)) \notin E_H$，或 $(v_1, v_2) \notin E_G$ 但 $(\pi(v_1), \pi(v_2)) \in E_H$，则拒绝；
  3. 遍历完所有 $v_1, v_2 \in V_G$ 后仍未拒绝，则接受。"

第 7 章习题

- 7.12 若图 G 的结点重新排序后，G 可以变得与 H 完全相同，则称 G 与 H 是**同构的**。令 $ISO = \{\langle G, H\rangle | G$ 和 H 是同构的图$\}$。证明 $ISO \in$NP。
- （续）若 N' 接受 $\langle G, H, \pi\rangle$，则说明 G 和 H 在映射 π 下同构；反之亦然。若 N_{ISO} 接受 $\langle G, H\rangle$，则存在某个计算分支被 N_{ISO} 接受，即 $\langle G, H\rangle \in ISO$。若 N_{ISO} 拒绝 $\langle G, H\rangle$，则 G 和 H 不可能同构（结点数或边数不相等）；或是所有计算分支都被 N_{ISO} 拒绝，又由 N_{ISO} 的步骤 2 和步骤 3 选择的是结点集间的所有双射，故 $\langle G, H\rangle \notin ISO$。因而 N_{ISO} 接受 $\langle G, H\rangle \iff \langle G, H\rangle \in ISO$。
- 由方法 1 的分析可知 N' 在 $|V_G|$、$|E_G|$ 和 $|E_H|$ 的多项式时间内完成，即 N_{ISO} 的步骤 3 在输入长度的多项式时间内完成运行。N_{ISO} 的步骤 1 可在 $|V_G|$ 和 $|V_H|$、$|E_G|$ 和 $|E_H|$ 的多项式时间内完成判断。又 N_{ISO} 的步骤 2 可在 $|V_G| = |V_H|$ 的时间内完成选择、步骤 4 可在常数时间内完成判断，故 N_{ISO} 在输入长度的多项式时间内接受/拒绝。
- 即判定 ISO 有非确定的多项式时间算法，故 $ISO \in$NP。

第 7 章习题

- 7.16 在有向图中，一个结点的**入度**为所有射入边的总数，**出度**为所有射出边的总数。证明如下问题是 NP 完全的。给定一个无向图 G 和一个 G 结点的子集 C，是否可以通过给 G 的每条边赋予方向，将 G 转换为一个有向图并且满足属于 C 的结点的入度或出度为 0，不属于 C 的结点的入度至少为 1？
- 记 $TD = \{\langle G, C\rangle |$ 能通过赋予边的方向使无向图 G 满足要求$\}$。先证明 $TD \in$NP：多项式时间判定 TD 的 NTM N_{TD} 运行如下：
  $N_{TD} =$ "对于输入 $\langle G, C\rangle$，其中 G 是无向图、$C \subseteq V_G$：
  1. 非确定地为 G 中每条边 $e \in E_G$ 选定一个方向；
  2. 检查每个 $v \in C$，如果 v 的入度和出度均不为 0，则拒绝。
  3. 检查每个 $v \in V_G \backslash C$，如果 v 的入度为 0，则拒绝。
  4. 如果完成步骤 2 和步骤 3 的检查仍未拒绝，则接受。
- N_{TD} 接受 $\langle G, C\rangle \iff \exists$ 一种赋方向方式使无向图 G 满足要求。
- 步骤 1 可在 $|E_G|$ 的时间内完成选择。步骤 2 和步骤 3 可在 $|V_G|$ 和 $|E_G|$ 的多项式时间内完成检查和判断。步骤 4 可在常数时间内完成判断。故 N_{TD} 在输入长度的多项式时间内接受/拒绝。
- 即判定 TD 有非确定的多项式时间算法，故 $TD \in$NP。

第 7 章习题

- 7.16 在有向图中，一个结点的**入度**为所有射入边的总数，**出度**为所有射出边的总数。证明如下问题是 NP 完全的。给定一个无向图 G 和一个 G 结点的子集 C，是否可以通过给 G 的每条边赋予方向，将 G 转换为一个有向图并且满足属于 C 的结点的入度或出度为 0，不属于 C 的结点的入度至少为 1？
- （续）再证明 NP 中的每个 A 都多项式时间可归约到 TD：考虑如下将 $3SAT$ 多项式时间归约到 TD。给定布尔公式 ϕ，将其转换成一个无向图 G 和其结点集的子集 C：
- 记 $x_1, ..., x_m$ 是 ϕ 中的变量，$c_1, ..., c_l$ 是 ϕ 的子句。每个变量 x_i 对应两个结点 x_i 和 $\neg x_i$，x_i 和 $\neg x_i$ 有边相连。每个字句 c_j 对应一个结点 c_j，如果 $x_i/\neg x_i \in c_j$，则 $x_i/\neg x_i$ 与 c_j 有边相连。令 $C = \{x_i, \neg x_i | 1 \leq i \leq m\}$。
- 如果 ϕ 有解，若 x_i 为真则 x_i 指向 $\neg x_i$、若 x_i 为假则 $\neg x_i$ 指向 $\neg x_i$。对于 $x_i \in c_j$，若 x_i 为真则 x_i 指向 c_j、若 x_i 为假则 c_j 指向 x_i；对于 $\neg x_i \in c_j$，若 x_i 为真则 c_j 指向 $\neg x_i$、若 x_i 为假则 $\neg x_i$ 指向 c_j。这样，若 x_i 为真，结点 x_i 的入度为 0，结点 $\neg x_i$ 的出度为 0；若 x_i 为假，结点 x_i 的出度为 0，结点 $\neg x_i$ 的入度为 0。由于 c_j 被满足，即 $\exists i$ 使得 $x_i \in c_j$ 且 x_i 为真或 $\neg x_i \in c_j$ 且 x_i 为假，因而 c_j 的入度至少为 1。即这样的赋方向方式使得无向图 G 满足要求，$\langle G, C \rangle \in TD$。

第 7 章习题

- 7.16 在有向图中，一个结点的**入度**为所有射入边的总数，**出度**为所有射出边的总数。证明如下问题是 NP 完全的。给定一个无向图 G 和一个 G 结点的子集 C，是否可以通过给 G 的每条边赋予方向，将 G 转换为一个有向图并且满足属于 C 的结点的入度或出度为 0，不属于 C 的结点的入度至少为 1？
- （续）如果 $\langle G, C\rangle \in TD$、在当前赋方向方式下 G 满足要求。由于结点 $x_i, \neg x_i \in C$ 且 x_i 和 $\neg x_i$ 有边相连，若 x_i 指向 $\neg x_i$（此时结点 x_i 的入度为 0、$\neg x_i$ 的出度为 0）则 x_i 赋值为真，若 $\neg x_i$ 指向 x_i（此时结点 x_i 的出度为 0、$\neg x_i$ 的入度为 0）则 x_i 赋值为假。由于结点 $c_j \in V_G \backslash C$、入度至少为 1，$\exists i$ 使得 $x_i \in c_j$ 且结点 x_i 指向 c_j 或 $\neg x_i \in c_j$ 且结点 $\neg x_i$ 指向 c_j，对应地必有 x_i 为真或 x_i 为假，从而子句 c_j 被满足。因而该赋值方式使得 ϕ 被满足，$\phi \in 3SAT$。
- 综上，$\phi \in 3SAT \iff \langle G, C\rangle \in TD$。又由 NP 中的每个 A 都多项式时间可归约到 $3SAT$，且上述转换能在 $|\phi|$ 的多项式时间内完成，故 NP 中的所有语言都多项式时间可归约到 TD。
- 综上，TD 是 NP 完全的。

第 7 章习题

- 7.29 令 $SET-SPLITTING=\{\langle S,C\rangle|S$ 是一个有穷集，$C=\{C_1,\ldots,C_k\}$ 是由 S 的某些子集组成的集合，$k>0$，使得 S 的元素可以被染为红色或蓝色，而且对所有 C_i，C_i 中的元素不会被染成同一种颜色 $\}$。证明 $SET-SPLITTING$ 是 NP 完全的。
- 先证明 $SET-SPLITTING \in$NP：多项式时间判定 $SET-SPLITTING$ 的 NTM N_{S-S} 运行如下：
  $N_{S-S}=$ "对于输入 $\langle S,C\rangle$，其中 S 是有穷集，$C\neq\emptyset\subseteq P(S)$：
  1. 非确定地为 S 中的每个元素染色（红色或蓝色）；
  2. 检查每个 $C_i\in C$，如果 C_i 中的元素被染成同一种颜色，则拒绝；
  3. 如果检查完所有的 C_i 后仍未拒绝，则接受。
- N_{S-S} 接受 $\langle S,C\rangle \iff \exists$ 一种染色方式使得所有的 C_i 中元素不会被染成同一种颜色 $\iff \langle S,C\rangle\in SET-SPLITTING$。
- 步骤 1 可在 $|S|$ 的时间内完成选择。步骤 2 可在 $|S|$ 和 k 的多项式时间内完成检查和判断。步骤 3 可在常数时间内完成判断。故 N_{S-S} 在输入长度的多项式时间内接受/拒绝。
- 即判定 $SET-SPLITTING$ 有非确定的多项式时间算法，故 $SET-SPLITTING \in$NP。

第 7 章习题

- 7.29 令 $SET-SPLITTING=\{\langle S,C\rangle | S$ 是一个有穷集，$C=\{C_1,...,C_k\}$ 是由 S 的某些子集组成的集合，$k>0$，使得 S 的元素可以被染为红色或蓝色，而且对所有 C_i，C_i 中的元素不会被染成同一种颜色 $\}$。证明 $SET-SPLITTING$ 是 NP 完全的。
- (续) 再证明 NP 中的每个 A 都多项式时间可归约到 $SET-SPLITTING$：考虑如下将 3SAT 多项式时间归约到 $SET-SPLITTING$。给定布尔公式 ϕ，将其转换成一个有穷集 S 和 $C\neq\emptyset\subseteq P(S)$：
- 记 $x_1,...,x_m$ 是 ϕ 中的变量，$d_1,...,d_l$ 是 ϕ 的子句。令 $S=\{x_1,\neg x_1,...,x_m,\neg x_m,y\}$，其中 y 是表示“假”的元素。$C=C_V\cup C_D$ 由两部分组成：$C_V=\{C_1,...,C_m\}$ 限制真值的选取，其中 $C_i(1\leq i\leq m)=\{x_i,\neg x_i\}$；$C_D=\{C_{m+1},...,C_{m+l}\}$ 表示子句，其中 $C_{m+i}(1\leq i\leq l)=\{s_1,s_2,s_3,y\}$，$s_j(1\leq j\leq 3)$ 为 d_i 中元素 x_{i_j} 或 $\neg x_{i_j}$。
- 如果 ϕ 有解，先将 y 染成蓝色，若 x_i 为真则将 x_i 染成红色、将 $\neg x_i$ 染成蓝色；若 x_i 为假则将 x_i 染成蓝色、将 $\neg x_i$ 染成红色。x_i 和 $\neg x_i$ 必有一红一蓝，所以 $C_i(1\leq i\leq m)$ 中的元素不会被染成同色。由于每个子句 $d_i(1\leq i\leq l)$ 能被满足，C_{m+i} 中定有 $s_j(1\leq j\leq 3)$ 被染成红色；又 y 被染成蓝色，故 C_{m+i} 中的元素不会被染成同色。即这样的染色方式使得所有的 C_i 中元素不会被染成同一种颜色，$\langle S,C\rangle\in SET-SPLITTING$。

第 7 章习题

- 7.29 令 $SET-SPLITTING=\{\langle S,C\rangle|S$ 是一个有穷集，$C=\{C_1,...,C_k\}$ 是由 S 的某些子集组成的集合，$k>0$，使得 S 的元素可以被染为红色或蓝色，而且对所有 C_i，C_i 中的元素不会被染成同一种颜色 $\}$。证明 $SET-SPLITTING$ 是 NP 完全的。
- （续）如果 $\langle S,C\rangle\in SET-SPLITTING$、在当前染色方式下所有的 C_i 中元素不会被染成同一种颜色。不防设 y 被染成蓝色；否则将每个元素的颜色对换（红换蓝、蓝换红），染色方式仍满足要求。若 x_i 被染成红色，则 x_i 赋值为真；若 x_i 被染成蓝色，则 x_i 赋值为假。由于 $C_i(1\le i\le m)$ 中的元素不能同色，即 x_i 和 $\neg x_i$ 必有一红一蓝；进而在该赋值下，染成红色的真值为真。又 $C_{m+i}(1\le i\le l)$ 中的元素不能同色，且 y 为蓝色，故 C_{m+i} 中一定存在 $s_j(1\le j\le 3)$ 被染成红色；进而在该赋值下，每个子句 d_i 中必有“真”出现、子句 d_i 被满足。因而该赋值方式使得 ϕ 被满足，$\phi\in 3SAT$。
- 综上，$\phi\in 3SAT \iff \langle S,C\rangle\in SET-SPLITTING$。又由 NP 中的每个 A 都多项式时间可归约到 $3SAT$，且上述转换能在 $|\phi|$ 的多项式时间内完成，故 NP 中的所有语言都多项式时间可归约到 $SET-SPLITTING$。
- 综上，$SET-SPLITTING$ 是 NP 完全的。

第 7 章习题

- 7.34 令 $U = \{\langle M, x, \#^t\rangle |$ 非确定型图灵机 M 在至少一个分支上在 t 步内接受输入 $x\}$。注意并不要求 M 在所有分支上停机。证明 U 是 NP 完全的。
- 先证明 $U \in$NP：多项式时间判定 U 的 NTM N_U 运行如下：
  $N_U =$ "对于输入 $\langle M, x, \#^t\rangle$，其中 M 是 NTM，x 是字符串：
  1. 在 x 上运行 M，运行 t 步；
  2. 如果 M 接受，则接受；如果 M 拒绝，则拒绝。
- N_U 接受 $\langle M, x, \#^t\rangle \iff \langle M, x, \#^t\rangle \in U$，且 N_U 在输入长度的多项式时间内接受/拒绝，故 $U \in$NP。
- 再证明 NP 中的每个 A 都多项式时间可归约到 U（方法 1）：考虑如下将 $3SAT$ 多项式时间归约到 U。由于 $3SAT \in$NP，记 N_{3S} 为 cn^k 步内判定 $3SAT$ 的 NTM，其中 n 为输入长度。将布尔公式 ϕ 映射到 $\langle N_{3S}, \phi, \#^{c|\phi|^k}\rangle$，则 $\phi \in 3SAT \iff \langle N_{3S}, \phi, \#^{c|\phi|^k}\rangle \in U$。又由 NP 中的每个 A 都多项式时间可归约到 $3SAT$，且上述映射能在 $|\phi|$ 的多项式时间内完成，故结论成立。
- 综上，U 是 NP 完全的。

第 7 章习题

- 7.34 令 $U = \{\langle M, x, \#^t \rangle |$ 非确定型图灵机 M 在至少一个分支上在 t 步内接受输入 $x\}$。注意并不要求 M 在所有分支上停机。证明 U 是 NP 完全的。
- 先证明 $U \in$NP：多项式时间判定 U 的 NTM N_U 运行如下：
  $N_U =$ "对于输入 $\langle M, x, \#^t \rangle$，其中 M 是 NTM，x 是字符串：
  1. 在 x 上运行 M，运行 t 步；
  2. 如果 M 接受，则接受；如果 M 拒绝，则拒绝。
- N_U 接受 $\langle M, x, \#^t \rangle \iff \langle M, x, \#^t \rangle \in U$，且 N_U 在输入长度的多项式时间内接受/拒绝，故 $U \in$NP。
- 再证明 NP 中的每个 A 都多项式时间可归约到 U（方法 2）：任取 $A \in$NP，N_A 为 cn^k 步内判定 A 的 NTM，其中 n 为输入长度。将输入 w 映到 $\langle N_A, w, \#^{c|w|^k} \rangle$，则 $w \in A \iff \langle N_A, w, \#^{c|w|^k} \rangle \in U$。且上述映射能在 $|w|$ 的多项式时间内完成，故结论成立。
- 综上，U 是 NP 完全的。

第 7 章习题

- 7.35 证明：若 P=NP，则任给布尔公式 ϕ，若 ϕ 是可满足的，则存在多项式时间算法给出一个 ϕ 的满足赋值。(注意：要求提供的算法计算一个函数而非若干函数，除非是 NP 包含的语言。P=NP 的假定说明 SAT 在 P 中，所以其可满足性的验证是多项式时间可解的。但是这个假设没有说明这项验证如何进行，而且验证不能给出满足赋值。必须证明它们一定能够被找到。提示：重复地使用可满足性验证器一位一位地找出满足赋值。)
- 若 $P = NP$，则 $SAT \in$P，进而有多项式的确定性算法 A_{SAT} 告知某个布尔公式 ϕ 可否被满足。
- 考虑如下的算法 A，来找出一个 ϕ 的满足赋值：
  $A =$ "对于输入 $\langle\phi\rangle$，其中 $\phi \in SAT$ 是可满足的布尔公式：
  1. 解码 ϕ，其中的变量为 $x_1, ..., x_n$；令 $\phi_0 = \phi$。
  2. 对于 $i = 1, ..., n-1$，依次执行第 3 步。
  3. 假设 x_i 为真，化简 ϕ_{i-1} 得到 ϕ'。询问 A_{SAT}，若 ϕ' 可满足，则将 x_i 赋值为真、令 $\phi_i = \phi'$；否则将 x_i 赋值为假，化简 ϕ_{i-1} 得到 ϕ_i。
  4. 假设 x_n 为真，若 ϕ_{n-1} 的真值为真，则将 x_n 赋值为真；否则将 x_n 赋值为假。
  5. 经过步骤 2、3 和 4 得到一组 ϕ 的满足赋值。

第 7 章习题

- 7.35 证明：若 P=NP，则任给布尔公式 ϕ，若 ϕ 是可满足的，则存在多项式时间算法给出一个 ϕ 的满足赋值。
- （续）先说明在 ϕ_{i-1} 可满足、变量为 $x_i, ..., x_n$ 的前提下，执行第 3 步后得到的 ϕ_i 也可满足、变量为 $x_{i+1}, ..., x_n$，且 $\phi_i(x_{i+1}, ..., x_n) = \phi_{i-1}(\overline{x_i}, x_{i+1}, ..., x_n)$，其中 $\overline{x_i}$ 为第 3 步给 x_i 的赋值。如果 $\phi'(x_{i+1}, ..., x_n) = \phi_{i-1}(1, x_{i+1}, ..., x_n)$ 可满足，由 $\overline{x_i} = 1$ 且 $\phi_i = \phi'$ 可知上述结论成立。否则 $\phi_i(x_{i+1}, ..., x_n) = \phi_{i-1}(0, x_{i+1}, ..., x_n)$ 可满足（不然与 ϕ_{i-1} 可满足矛盾），此时 $\overline{x_i} = 0$、上述结论也成立。
- $\phi_0 = \phi$ 可满足、变量为 $x_1, ..., x_n$，由归纳法可知 ϕ_{n-1} 可满足、变量为 x_n，且 $\phi_{n-1}(x_n) = \phi_{n-2}(\overline{x_{n-1}}, x_n) = ... = \phi(\overline{x_1}, ..., \overline{x_{n-1}}, x_n)$。如果 $\phi_{n-1}(1) = 1$，则第 4 步给 x_n 的赋值 $\overline{x_n} = 1$；否则 $\phi_{n-1}(0) = 1$（不然与 ϕ_{n-1} 可满足矛盾），此时 $\overline{x_n} = 0$。即执行第 4 步后 $\phi(\overline{x_1}, ..., \overline{x_n}) = \phi_{n-1}(\overline{x_n}) = 1$。即算法 A 给出了一组 ϕ 的满足赋值 $\overline{x_1}, \overline{x_2}, ..., \overline{x_n}$。
- 步骤 3 中化简布尔公式、利用 A_{SAT} 判断 ϕ' 是否可满足均可在多项式时间内完成；步骤 3 执行 $n-1$ 次。步骤 4 中得到 x_n 的赋值也可在多项式时间内完成。进而算法 A 能在 $|\phi|$ 的多项式时间内得到结果。
- 综上，若 P=NP，多项式时间算法 A 能给出一组 $\phi \in SAT$ 的满足赋值。

第 7 章习题

- 7.39 有一个盒子和一些卡片，如下图所示。盒子里有栓塞，卡片上有凹口，所以每张卡片可以以两种方式放入盒子中。每张卡片上有两排孔，有些孔没有打穿。把所有卡片放进盒子里，使得盒子的底被完全覆盖（即，每个孔的位置都被至少一张在该位置上无孔的卡片堵住），则谜题就算破解了。令 $PUZZLE = \{\langle c_1, ..., c_k\rangle |$ 每个 c_i 代表一张卡片，并且这个卡片集有解$\}$。证明 $PUZZLE$ 是 NP 完全的。
- 先证明 $PUZZLE \in$NP：多项式时间判定 $PUZZLE$ 的 NTM N_P 运行如下：

  $N_P =$ "对于输入 $\langle c_1, ..., c_k\rangle$，其中每个 c_i 是一张卡片：
  1. 非确定地为每张卡片选择一种放置方式放入盒子中；
  2. 检查每个孔的位置，判断盒子的底部是否被完全覆盖；若是则接受，否则拒绝。
- N_P 接受 $\langle c_1, ..., c_k\rangle \iff \exists$ 一种放置方式使得盒子底部被完全覆盖 $\iff \langle c_1, ..., c_k\rangle \in PUZZLE$。
- 步骤 1 可在 k 的时间内完成选择。步骤 2 可在孔的个数和 k 的多项式时间内完成检查和判断。故 N_P 在输入长度的多项式时间内接受/拒绝。
- 即判定 $PUZZLE$ 有非确定的多项式时间算法，故 $PUZZLE \in$NP。

第 7 章习题

- 7.39 有一个盒子和一些卡片，如下图所示。盒子里有栓塞，卡片上有凹口，所以每张卡片可以以两种方式放入盒子中。每张卡片上有两排孔，有些孔没有打穿。把所有卡片放进盒子里，使得盒子的底被完全覆盖（即，每个孔的位置都被至少一张在该位置上无孔的卡片堵住），则谜题就算破解了。令 $PUZZLE = \{\langle c_1, ..., c_k \rangle |$ 每个 c_i 代表一张卡片，并且这个卡片集有解$\}$。证明 $PUZZLE$ 是 NP 完全的。
- （续）再证明 NP 中的每个 A 都多项式时间可归约到 $PUZZLE$：考虑如下将 $3SAT$ 多项式时间归约到 $PUZZLE$。给定布尔公式 ϕ，将其转换成一系列卡片 $\{cd_1, ...cd_{m+1}\}$：
- 记 $x_1, ..., x_m$ 是 ϕ 中的变量，$c_1, ..., c_l$ 是 ϕ 的子句。每个变量 x_i 对应一张卡片 cd_i：对于左排，如果 $x_i \notin c_j$ 则在从上往下的第 j 个孔位打孔（$x_i \in c_j$ 则不打孔）；对于右排，如果 $\neg x_i \notin c_j$ 则在从上往下的第 j 个孔位打孔（$\neg x_i \in c_j$ 则不打孔）。再拿一张卡片 cd_{m+1}，左排的 l 个孔位全都打孔，右排的 l 个孔位全都不打孔。
- 如果 ϕ 有解，将 x_i 为真的卡片 cd_i 正常放入盒子、x_i 为假的卡片 cd_i 翻转后放入盒子（cd_{m+1} 正常放入）。由于右排的 l 个孔位全都被 cd_{m+1} 覆盖，只需考虑左排的 l 个孔位：对于第 j 个孔位，由于 c_j 被满足，$\exists i$ 使得 $x_i \in c_j$ 且 x_i 为真或 $\neg x_i \in c_j$ 且 x_i 为假，故该孔位被 cd_i 覆盖。即这样的放置方式使得盒子的底部被完全覆盖，$\langle cd_1, ...cd_{m+1} \rangle \in PUZZLE$。

第 7 章习题

- 7.39 有一个盒子和一些卡片，如下图所示。盒子里有栓塞，卡片上有凹口，所以每张卡片可以以两种方式放入盒子中。每张卡片上有两排孔，有些孔没有打穿。把所有卡片放进盒子里，使得盒子的底被完全覆盖（即，每个孔的位置都被至少一张在该位置上无孔的卡片堵住），则谜题就算破解了。令 $PUZZLE = \{\langle c_1, \ldots, c_k \rangle |$ 每个 c_i 代表一张卡片，并且这个卡片集有解$\}$。证明 $PUZZLE$ 是 NP 完全的。
- （续）如果 $\langle cd_1, \ldots cd_{m+1} \rangle \in PUZZLE$、在当前放置方式下盒子底被完全覆盖，不妨设 cd_{m+1} 是正常放入盒子（如果不是，可以把所有卡片都翻转，仍是该卡片集的一个解）。与构造卡片时相比，正常放入盒子的卡片 cd_i，x_i 赋值为真；翻转后放入盒子的卡片 cd_i，x_i 赋值为假。对于每个子句 c_j，由于左排的第 j 个孔位需要被覆盖，且仅有 $x_i \in c_j$ 或 $\neg x_i \in c_j$ 的卡片能够覆盖，即 $\exists i$ 使得 $x_i \in c_j$ 且 x_i 为真或 $\neg x_i \in c_j$ 且 x_i 为假，从而 c_j 被满足。因而该赋值方式使得 ϕ 被满足，$\phi \in 3SAT$。
- 综上，$\phi \in 3SAT \iff \langle cd_1, \ldots cd_{m+1} \rangle \in PUZZLE$。又由 NP 中的每个 A 都多项式时间可归约到 $3SAT$，且上述转换能在 $|\phi|$ 的多项式时间内完成，故 NP 中的所有语言都多项式时间可归约到 $PUZZLE$。
- 综上，$PUZZLE$ 是 NP 完全的。

第 7 章习题

- 7.45 证明若 P=NP，则除了语言 $A=\emptyset$ 和 $A=\Sigma^*$ 以外，所有语言 $A\in$P 都是 NP 完全的。
- 由于 P=NP 且 $A\in$P，故 $A\in$NP。
- 对于 NP 的每个 L，由于 P=NP，$L\in$P，故存在图灵机 M_L 可在多项式时间内判定 L。由于 $A\neq\emptyset$ 且 $A\neq\Sigma^*$，取 $w_{in}\in A$ 和 $w_{out}\notin A$。构造多项式时间可计算函数 $f:\Sigma^*\to\Sigma^*$ 如下：在输入 w 上运行 M_L；若 $w\in L$ 则 $f(w)=w_{in}$，若 $w\notin L$ 则 $f(w)=w_{out}$。
- 即 $w\in L \iff f(w)\in A$，且 f 能在 $|w|$ 的多项式时间内完成，故 NP 中的所有语言都多项式时间可归约到 A。
- 综上，A 是 NP 完全的。

1 第 7 章：时间复杂性

2 第 8 章：空间复杂性

3 第 9 章：难解性

4 第 10 章：复杂性理论高级专题

第 8 章习题

- 8.1 证明对于任意函数 $f:\mathbb{N}\to\mathbb{R}^+$，其中 $f(n)\geq n$，不论用单带图灵机模型还是用双带只读输入图灵机模型，所定义的空间复杂性 $\mathrm{SPACE}(f(n))$ 总是相同的。
- 对于 A 属于单带图灵机模型定义的 $\mathrm{SPACE}(f(n))$，我们使用双带只读输入图灵机模型进行如下模拟：将只读输入带上的输入 w 复制到读写工作带上，在读写工作带上模拟单带图灵机的操作即可。由于判定 A 在单带图灵机上仅需使用 $O(f(n))$ 的空间，在上述模拟的双带只读输入图灵机上也仅需使用 $O(f(n))$ 的空间。
- 对于 A 属于双带只读输入图灵机模型定义的 $\mathrm{SPACE}(f(n))$，我们使用单带图灵机模型进行如下模拟：将单带图灵机分为只读区和读写区，带头在只读区和读写区来回移动、模拟双带只读输入图灵机在只读输入带和读写工作带上的操作。由于判定 A 在双带只读输入图灵机上仅需使用 $O(f(n))$ 的空间，在上述模拟的单带图灵机上也仅需使用 $n+O(f(n))=O(f(n))$ 的空间。
- 综上，单带和双带只读输入图灵机模型所定义的空间复杂性 $\mathrm{SPACE}(f(n))$ 相同。

第 8 章习题

- 8.4 证明 PSPACE 在并、补和星号运算下封闭。
- 给定 $L_1, L_2 \in$PSPACE，M_1, M_2 分别是使用多项式空间判定 L_1, L_2 的确定型图灵机。构造使用多项式空间判定 $L_1 \cup L_2$ 的确定型图灵机 M，运行如下：
  $M =$ "对输入 w：
  1. 将输入 w 在原先的输入右侧复制一份；
  2. 在第 2 个 w 及其右侧的区域内运行 M_1（在 w 上），如果 M_1 接受则接受；
  3. 清除第 1 个 2 右侧的所有内容，在 w 上运行 M_2，如果 M_2 接受则接受，否则拒绝。
- 若 M 接受 w，则 $w \in L_1 \cup (\overline{L_1} \cap L_2) = L_1 \cup L_2$；若 M 拒绝 w，则 $w \in \overline{L_1} \cap \overline{L_2} = \overline{L_1 \cup L_2}$。故 M 接受 $w \iff w \in L_1 \cup L_2$。
- 由于步骤 2 和步骤 3 都仅需输入长度的多项式空间，步骤 1 也仅需输入长度的空间，M 仅需输入长度的多项式空间。
- 即判定 $L_1 \cup L_2$ 有多项式空间算法，故 PSPACE 在并下封闭。

第 8 章习题

- 8.4 证明 PSPACE 在并、补和星号运算下封闭。
- 给定 $L \in$PSPACE，M 是使用多项式空间判定 L 的确定型图灵机。构造使用多项式空间判定 $\overline{L}$ 的确定型图灵机 M'，运行如下：
  $M' =$ "对输入 w：
  1. 在 w 上运行 M，如果 M 接受则拒绝，如果 M 拒绝则接受。
- 若 M' 接受 w，则 M 拒绝 w，$w \in \overline{L}$；若 M' 拒绝 w，则 M 接受 w，$w \in L$。故 M 接受 $w \iff w \in \overline{L}$。
- 由于步骤 1 仅需输入长度的多项式空间，M' 也仅需输入长度的多项式空间。
- 即判定 $\overline{L}$ 有多项式空间算法，故 PSPACE 在补下封闭。

第 8 章习题

- 8.4 证明 PSPACE 在并、补和**星号**运算下封闭。
- 给定 $L \in$PSPACE，M 是使用多项式空间判定 L 的确定型图灵机。构造使用多项式空间判定 L^* 的确定型图灵机 M'：
  M' = "对长度为 n 的输入 w：
  1. 如果 w 为 ε，则接受；
  2. 依次选取 $k = 1, 2, ..., n$;
  3. 将 w 分成 k 段 $w_1, ...w_k$，每段长度至少为 1；
  4. 依次在 w_i 上运行 M，如果 M 拒绝 w_i 则返回步骤 3；
  5. 如果对于所有 $i = 1, ..., k$，M 接受 w_i，则接受；
  6. 如果穷尽步骤 2 和步骤 3 的所有情况仍未接受，则拒绝。
- 若 M' 接受 w，则存在 $w = w_1...w_k$，对于所有 $i = 1, ..., k$，M 接受 w_i，$w \in L^*$；若 M' 拒绝 w，则 w 的所有拆分情况都无法验证 $w \in L^*$，$w \notin L^*$。故 M 接受 $w \iff w \in L^*$。
- 由于步骤 4 运行 M 仅需输入长度的多项式空间，步骤 2、步骤 3 和步骤 4 的 3 层循环可重复利用空间，M' 也仅需输入长度的多项式空间。
- 即判定 L^* 有多项式空间算法，故 PSPACE 在星号下封闭。

第 8 章习题

- 8.6 证明 PSPACE 难的语言也是 NP 难的。
- 设 L 是 PSPACE 难的，则 PSPACE 中的每一个语言 A 多项式时间可归约到 L。
- 任取 NP 中的语言 B，存在非确定型图灵机 N_B 在多项式时间内判定 B；由于时间开销不可能超过空间开销，N_B 需要的空间不超过多项式，故 $B \in$NPSPACE=PSPACE。
- 综上，NP 中的每一个语言 B 多项式时间可归约到 L，故 L 也是 NP 难的。

第 8 章习题

- 8.7 证明 NL 在并、连接和星号运算下封闭。
- 给定 $L_1, L_2 \in$NL，L_1, L_2 分别被 NTM N_1, N_2 在对数空间内判定。构造对数空间内判定 $L_1 \cup L_2$ 的 NTM N，运行如下：
  N= "对输入 w：
  1. 非确定地选择 $N_i, i = 1, 2$；
  2. 在 w 上运行 N_i；
  3. 如果 N_i 接受，则接受；否则拒绝。"
- 若 N 接受 w，则存在某个计算分支被 N 接受，即 $w \in L_1 \cup L_2$；若 N 拒绝 w，则所有计算分支都被 N 拒绝，即 $w \in \overline{L_1} \cap \overline{L_2} = \overline{L_1 \cup L_2}$。因而 N 接受 $w \iff w \in L_1 \cup L_2$。
- 步骤 2 占用的额外空间不超过 $O(log|w|)$，故 N 在 $|w|$ 的对数空间内完成判定。
- 即 $L_1 \cup L_2$ 能被非确定型图灵机在对数空间内判定，故 NL 在并下封闭。

第 8 章习题

- 8.7 证明 NL 在并、**连接**和星号运算下封闭。
- 给定 $L_1, L_2 \in$NL，L_1, L_2 分别被 NTM N_1, N_2 在对数空间内判定。构造对数空间内判定 $L_1 \circ L_2$ 的 NTM N，运行如下：
  $N=$ "对长度为 n 的输入 w：
  1. 非确定地选择 $i = 0, 1, ..., n$；
  2. 记 w 的前 i 位为 w_1、后 $n-i$ 位为 w_2，在 $\langle w_1, w_2\rangle$ 上运行 N'；
  3. 如果 N' 接受，则接受；否则拒绝。"
- 其中子程序 N' 运行如下：
  $N'=$ "对输入 $\langle w_1, w_2\rangle$：
  1. 在 w_1 上运行 N_1，如果 N_1 拒绝则拒绝；
  2. 在 w_2 上运行 N_2，如果 M_2 拒绝则拒绝，否则接受。"
- 易知 N' 接受 $\langle w_1, w_2\rangle \iff w_1 \in L_1$ 且 $w_2 \in L_2$。若 N 接受 w，则存在某个计算分支被 N 接受，即 $w \in L_1 \circ L_2$；若 N 拒绝 w，则所有计算分支都被 N 拒绝，又由 N 的步骤 1 和步骤 2 选择的是将 w 拆成两个子串的所有情况，故 $w \notin L_1 \circ L_2$。因而 N 接受 $w \iff w \in L_1 \circ L_2$。
- 由于 N' 的步骤 1 和步骤 2 都在各自输入长度的对数空间内运行完成，N' 也在输入长度的对数空间内完成判定。即 N 的步骤 2 在 $|w|$ 的对数空间内运行完成，故 N 在 $|w|$ 的对数空间内完成判定。
- 即 $L_1 \circ L_2$ 能被非确定型图灵机在对数空间内判定，故 NL 在连接下封闭。

第 8 章习题

- 8.7 证明 NL 在并、连接和**星号**运算下封闭。
- 给定 $L \in$NL，L 被 NTM N 在对数空间内判定。构造对数空间内判定 L^* 的 NTM N^*，运行如下：
  N= “对于输入 w：
  1. 初始化两个指针 p_1 和 p_2，均指向第一个输入字符前面。
  2. 若 p_2 之后无输入字符，则接受。
  3. 向前移动 p_2，非确定地选择一个位置（p_1 和 p_2 之间至少有一个字符，p_2 至多指向最后一个输入字符后面）。
  4. 在 p_1 和 p_2 之间的子串上模拟 N。
  5. 如果 N 接受，则令 $p_1 = p_2$，返回步骤 2；如果 N 拒绝，则拒绝。”

第 8 章习题

- 8.7 证明 NL 在并、连接和**星号**运算下封闭。
- （续）已知 N 接受 $w' \iff w' \in L$。若 N^* 接受 w，则存在某个计算分支被 N^* 接受。要么 $w = \varepsilon$；要么存在 w 的某种划分方式 $w = w_1 \circ w_2 \circ \ldots \circ w_k$(其中 $|w_i| > 0$)，使得 $\forall 1 \leq i \leq k$，N 接受 w_i 即 $w_i \in L$。故 $w \in L^*$。反之若 $w \in L^*$，则 $w = \varepsilon$ 或存在 w 的某种划分方式 $w = w_1 \circ w_2 \circ \ldots \circ w_k$(其中 $|w_i| > 0$)，使得 $\forall 1 \leq i \leq k, w_i \in L$。$w = \varepsilon$ 时 N^* 会在初始化后的第 2 步立即接受 w。$w \neq \varepsilon$ 时，由于 p_2 的非确定性前移会穷尽 w 的所有划分方式，故一定存在某个计算分支，其划分方式就是 $w = w_1 \circ w_2 \circ \ldots \circ w_k$，那么 N^* 在该计算分支接受 w，即 N^* 接受 w。综上，N^* 接受 $w \iff w \in L^*$。
- 设 $|w| = n$，指针 p_1 和 p_2 的位置可用 $0, 1, \ldots, n$ 表示，故维护两个指针所需的空间不超过 $|w|$ 的对数。每次模拟 N 所需的空间不超过 $O(log|w'|) \leq O(log|w|)$（其中 w' 为当时 p_1 和 p_2 之间的子串）。故 N^* 在 $|w|$ 的对数空间内完成判定。
- 即 L^* 能被非确定型图灵机在对数空间内判定，故 NL 在星号下封闭。

第 8 章习题

- 8.11(a) 令 $ADD = \{\langle x, y, z\rangle | x, y, z > 0$ 且为二进制整数，$x + y = z\}$，证明 $ADD \in L$。
- 如下给出判定 ADD 的图灵机 M_{ADD}：
  M_{ADD}= "对于输入 $\langle x, y, z\rangle$，其中 $x, y, z > 0$ 且为二进制整数：
  1. 不妨设 $|x| \leq |y|$，否则对换对 x 和 y 的操作即可。初始化 3 个指针 p_x、p_y 和 p_z，分别指向 x、y 和 z 的最低位；初始化进位比特 $c = 0$。
  2. 记 p_x、p_y 和 p_z 指向的比特分别为 b_x、b_y 和 b_z。计算 $2 \times c' + b_r = b_x + b_y + c$($c'$ 和 b_r 为比特)。如果 $b_r \neq b_z$，则拒绝；否则令 $c = c'$。
  3. 如果 p_x 已经指向 x 的最高位，则执行第 5 步；否则，如果 p_z 已经指向 z 的最高位则拒绝，没有则将 p_x、p_y 和 p_z 分别指向 x、y 和 z 的更高一位，执行第 2 步。
  4. 记 p_y 和 p_z 指向的比特分别为 b_y 和 b_z。计算 $2 * c' + b_r = b_y + c$ (c' 和 b_r 为比特)。如果 $b_r \neq b_z$，则拒绝；否则令 $c = c'$。
  5. 如果 p_y 已经指向 y 的最高位，则执行第 6 步；否则，如果 p_z 已经指向 z 的最高位则拒绝，没有则将 p_y 和 p_z 分别指向 y 和 z 的更高一位，执行第 4 步。
  6. 如果 $c = 0$，若 p_z 已经指向 z 的最高位则接受；否则拒绝。如果 $c = 1$，若 p_z 指向的更高一位是 z 的最高位则接受，其他情况拒绝。"

第 8 章习题

- 8.11(a) 令 $ADD = \{\langle x, y, z\rangle | x, y, z > 0$ 且为二进制整数，$x + y = z\}$，证明 $ADD \in L$。
- （续）如果 M_{ADD} 接受 $\langle x, y, z\rangle$，则步骤 2、3、4 和 5 说明 x 和 y 按位相加得到的结果与 z 对应位置的比特一致，步骤 6 说明最高位的进位情况也与 z 相一致；故 $x + y = z$，即 $\langle x, y, z\rangle \in ADD$。反之如果 M_{ADD} 拒绝 $\langle x, y, z\rangle$，步骤 2 和 4 中拒绝说明按位相加时某一位不一致；步骤 3 和 5 中拒绝说明 x 或 y 还有更高位但 z 已经没有更高位了，即 $z < x + y$；步骤 6 中，$c = 0$ 时拒绝说明最终没有进位但 z 还有更高位，即 $z > x + y$，$c = 1$ 时拒绝说明最终进了一位但要么 z 没有更高位了、要么 z 高处超过一位，即 $z < x + y$ 或 $z > x + y$。对于上述所有拒绝的情况，$x + y \neq z$，故 $\langle x, y, z\rangle \notin ADD$。综上，$M_{ADD}$ 接受 $\langle x, y, z\rangle \iff \langle x, y, z\rangle \in ADD$。
- 维护 3 个指针 p_x、p_y 和 p_z 所需的空间不超过 $|x|$、$|y|$ 和 $|z|$ 的对数。维护进位比特 c、计算和存储中间结果 b_r 和 c 所需的空间为常数。故 M_{ADD} 在 $|w| = |\langle x, y, z\rangle|$ 的对数空间内完成判定。
- 即 ADD 能被确定型图灵机在对数空间内判定，故 $ADD \in$L。

第 8 章习题

- 8.16 在有向图中，如果每一对结点间都有双向的有向路径连接，则它称为**强连通的**。令 $STRONG-CONNECTED = \{\langle G\rangle | G$ 是强连通图$\}$，证明 $STRONG-CONNECTED$ 是 NL 完全的。
- 先证明 $STRONG-CONNECTED \in$NL：使用对数空间判定 $STRONG-CONNECTED$ 的非确定型图灵机 N_{S-C} 运行如下：
  $N_{S-C} =$ "对于输入 $\langle G\rangle$，G 是有向图：
  1. 依次选取 $s, t \in V_G$，其中 $s \neq t$；
  2. 在 $\langle G, s, t\rangle$ 上运行使用对数空间判定 $PATH$ 的非确定型图灵机 N_{PATH}，如果 N_{PATH} 拒绝则拒绝；
  3. 如果穷尽步骤 1 的所有情况仍未拒绝，则接受。
- 如果 N_{S-C} 接受 $\langle G\rangle$，则由 N_{PATH} 的结果可知每一对结点间都有双向的有向路径连接，故 $\langle G\rangle \in STRONG-CONNECTED$。如果 N_{S-C} 拒绝 $\langle G\rangle$，则存在结点 s 和 t，G 中不存在从 s 到 t 的有向路径，故 $\langle G\rangle \notin STRONG-CONNECTED$。
- 由于步骤 2 仅需输入长度的对数空间，且步骤 1 的循环可重复利用空间，N_{S-C} 也仅需输入长度的对数空间。
- 即判定 $STRONG-CONNECTED$ 有非确定的对数空间算法，故 $STRONG-CONNECTED \in$NL。

第 8 章习题

- 8.16 在有向图中，如果每一对结点间都有双向的有向路径连接，则它称为**强连通的**。令 $STRONG-CONNECTED = \{\langle G\rangle | G$ 是强连通图$\}$，证明 $STRONG-CONNECTED$ 是 NL 完全的。
- （续）再证明 NL 中的每个 A 都对数空间可归约到 $STRONG-CONNECTED$：考虑如下将 $PATH$ 对数空间归约到 $STRONG-CONNECTED$。给定有向图 G 和结点 s, t，构造新的有向图 G'：
- $V_{G'} = V_G$，且 G' 保留 G 中的边。对于 G' 中不是 s 且不是 t 的结点 v，添加从 v 到 s 的边和从 t 到 v 的边（如果不存在这样的 v，则需添加从 t 到 s 的边）。
- 如果 $\langle G, s, t\rangle \in PATH$，则 G 中存在从 s 到 t 的有向路径。对于任意结点 $v_1 \neq t, v_2 \neq s$，G' 中存在从 v_1 依次经过 s 和 t 到 v_2 的有向路径；且 G' 中存在从 t 到 $v \neq s$、从 $v \neq t$ 到 s 以及从 t 到 s 的有向路径。故 G' 中每一对结点间都有双向的有向路径连接，即 $\langle G'\rangle \in STRONG-CONNECTED$。

第 8 章习题

- 8.16 在有向图中，如果每一对结点间都有双向的有向路径连接，则它称为**强连通的**。令 $STRONG-CONNECTED = \{\langle G \rangle | G$ 是强连通图$\}$，证明 $STRONG-CONNECTED$ 是 NL 完全的。
- （续）如果 $\langle G, s, t \rangle \notin PATH$，则 G 中不存在从 s 到 t 的有向路径。由于 G' 中添加的边要么从 t 射出、要么射入 s，故 G' 中仍不存在从 s 到 t 的有向路径，即 $\langle G' \rangle \notin STRONG-CONNECTED$。
- 综上，$\langle G, s, t \rangle \in PATH \iff \langle G' \rangle \in STRONG-CONNECTED$。又由 NL 中的每个 A 都对数空间可归约到 $PATH$，且上述构造能在 $|\langle G, s, t \rangle|$ 的对数空间内完成（使用一个指针即可输出节点集和原边集，使用一个记录器即可输出新增的边集），故 NL 中的所有语言都对数空间可归约到 $STRONG-CONNECTED$。
- 综上，$STRONG-CONNECTED$ 是 NL 完全的。

第 8 章习题

- 8.18 证明 A_{NFA} 是 NL 完全的。
- 先证明 A_{NFA} ∈NL：使用对数空间判定 A_{NFA} 的非确定型图灵机 $N_{A_{NFA}}$ 运行如下：
  $N_{A_{NFA}}$ = "对于输入 $\langle B, w\rangle$，B 是 NFA，w 是字符串：
  1. 记 $B = (Q, \Sigma, \delta, q_0, F)$，$|Q| = k$；$|w| = n, w = w_1 w_2 \dots w_n$，其中 $w_i \in \Sigma$。初始化当前状态 q 为 q_0，指针 i 为 0，计数器 c 为 0。
  2. 如果 $i = n$ 且 $q \in F$ 则接受；如果 $c \geq (n+1)k$ 则拒绝。
  3. 若 $i < n$，则非确定地选择 $q \in \delta(q, \epsilon)$ 或 $q \in \delta(q, w_{i+1})$；如果 q 选自后者，则令 i 为 $i+1$。若 $i = n$，则非确定地选择 $q \in \delta(q, \epsilon)$。令 c 为 $c+1$；返回第 2 步。
- $N_{A_{NFA}}$ 的每个计算分支在模拟输入 w 时 NFA B 的状态迁移过程。如果 $N_{A_{NFA}}$ 接受 $\langle B, w\rangle$，则输入 w 时 NFA B 经过某个计算分支的状态迁移可由 q_0 到达接受状态，即 B 接受 w、$\langle B, w\rangle \in A_{NFA}$。如果 $\langle B, w\rangle \in A_{NFA}$、$B$ 接受 w，则存在状态迁移过程从 q_0 到达某个接受状态，能被某个计算分支捕获且迁移 $\leq (n+1)(k-1)+n$ 次（最多 $n+1$ 个连续的 ε 迁移，每个最多有 $k-1$ 次；另有 n 次非 ε 迁移），故 $N_{A_{NFA}}$ 接受 $\langle B, w\rangle$。
- 维护当前状态 q 和指针 i 分别需要 $log(k)$ 和 $log(n)$ 的空间。维护计数器 c 需要 $log((n+1)k) = O(log(n)) + O(log(k))$ 的空间。故 $N_{A_{NFA}}$ 在输入的对数空间内完成判定。

第 8 章习题

- 8.18 证明 A_{NFA} 是 NL 完全的。
- （续）即判定 A_{NFA} 有非确定的对数空间算法，故 A_{NFA} $\in$NL。
- 再证明 NL 中的每个 A 都对数空间可归约到 A_{NFA}：考虑如下将 $PATH$ 对数空间归约到 A_{NFA}。给定有向图 $G = \langle V_G, E_G \rangle$ 和结点 s, t，构造如下的 NFA $B = (Q, \Sigma, \delta, q_0, F)$ 和字符串 w：
- $Q = V_G, \Sigma = \emptyset, q_0 = s, F = \{t\}, \delta(u, \varepsilon) = \{v | (u, v) \in E_G\}$； $w = \varepsilon$。
- 如果 $\langle G, s, t \rangle \in PATH$，则 G 中存在从 s 到 t 的有向路径。对应地，在 NFA B 中存在从 s 到 t 的一系列 ε 迁移，故 B 接受 $w = \varepsilon$、$\langle B, w \rangle \in A_{NFA}$。反之如果 $\langle B, w \rangle \in A_{NFA}$、$B$ 接受 $w = \varepsilon$，则在 NFA B 中存在从 s 到 t 的一系列 ε 迁移（为什么？）。对应地，在 G 中存在从 s 到 t 的有向路径，$\langle G, s, t \rangle \in PATH$。
- 综上，$\langle G, s, t \rangle \in PATH \iff \langle B, w \rangle \in A_{NFA}$。又由 NL 中的每个 A 都对数空间可归约到 $PATH$，且上述构造能在 $|\langle G, s, t \rangle|$ 的对数空间内完成（使用一个指针即可输出 NFA B 的五元组和字符串 w），故 NL 中的所有语言都对数空间可归约到 A_{NFA}。
- 综上，A_{NFA} 是 NL 完全的。

第 8 章习题

- 8.31 考虑问题 7.39 中描述的语言 *PUZZLE* 的双人版。每名选手开始时都有一叠排好序的谜卡。他们轮流地按序把卡片放进盒子，并有权选择哪一面朝上。如果在最终的盒子中所有孔的位置都被堵住了，则选手 I 赢。如果还有孔的位置没被堵住，则选手 II 赢。证明对于给定的卡片的起始格局，判定哪位选手有必胜策略的问题是 PSPACE 完全的。
- 记 $GG_P = \{\langle c_1, \ldots, c_n\rangle |$ 在该双人版游戏中，选手 I 有必胜策略$\}$。先证明 $GG_P \in$PSPACE，构造使用多项式空间判定 GG_P 的图灵机 M_P：
  $M_P =$ "对输入 $\langle c_1, \ldots, c_n\rangle$，$c_i$ 是 *PUZZLE* 中的卡片：
  1. 如果所有孔的位置都被堵住则接受，否则若所有卡片的放置方式都已确定则拒绝；
  2. 如果轮到选手 I 放卡片 c_i，在剩余卡片上递归调用 M_P，先将 c_i 正常放入，然后将 c_i 翻转放入。只要有一个结果是接受则接受，否则拒绝；
  3. 如果轮到选手 II 放卡片 c_i，在剩余卡片上递归调用 M_P，先将 c_i 正常放入，然后将 c_i 翻转放入。若两个结果都是接受则接受，否则拒绝。
- M_P 接受 $\langle c_1, \ldots, c_n\rangle \iff$ 无论选手 II 如何个操作，选手 I 总能找到胜利的方式 $\iff$ 选手 I 有必胜策略。
- M_P 递归的深度最多等于卡片的个数。在每一层只需储存一张卡片的放置方式，所以空间总消耗是 $O(n)$。因此 M_P 在线性空间内运行。
- 即判定 GG_P 有确定的多项式空间算法，故 $GG_P \in$PSPACE。

第 8 章习题

- 8.31 考虑问题 7.39 中描述的语言 *PUZZLE* 的双人版。每名选手开始时都有一叠排好序的谜卡。他们轮流地按序把卡片放进盒子，并有权选择哪一面朝上。如果在最终的盒子中所有孔的位置都被堵住了，则选手 I 赢。如果还有孔的位置没被堵住，则选手 II 赢。证明对于给定的卡片的起始格局，判定哪位选手有必胜策略的问题是 PSPACE 完全的。
- （续）再证明 PSPACE 中的每个 A 都多项式时间可归约到 GG_P：考虑如下将 $TQBF$ 多项式时间归约到 GG_P。给定公式 ϕ（可以假定 ϕ 中 $\exists$ 和 $\forall$ 交替出现且 ψ 是合取范式，为什么？），构造 $\langle c_1, \ldots, c_n \rangle$：
- 习题 7.39 可将 ψ 转换成 $\langle c_1, \ldots, c_n \rangle$（没有使用到 SAT 有别于 $3SAT$ 的特性）且变量的赋值方式和卡片的放置方式一一对应。ϕ 以 $\exists/\forall$ 开头则让选手 I/II 选手先放第一张卡片。
- 对于使得 ψ 被满足的变量赋值方式对应的卡片放置方式，能将盒子的底部完全覆盖。因而 $\phi \in TQBF \iff \phi$ 为真 $\iff \langle c_1, \ldots, c_n \rangle$ 的双人版游戏中选手 I 有必胜策略 $\iff \langle c_1, \ldots, c_n \rangle \in GG_P$。
- 又由 PSPACE 中的每个 A 都多项式时间可归约到 $TQBF$，且上述构造能在 $|\phi|$ 的多项式时间内完成，故 PSPACE 中的所有语言都多项式时间可归约到 GG_P。
- 综上，GG_P 是 PSPACE 完全的。

第 8 章习题

- 8.33 设 A 是由正确嵌套的圆括号组成的语言。例如，(()) 和 (()(()))() 属于 A；而)(则不属于 A。证明 A 属于 L。
- 构造使用多项式空间判定 A 的双带只读输入图灵机 M：
  $M=$ "对输入 w，w 是由 (和) 构成的字符串：
    1. 计数器 cnt 初始为 0；
    2. 从左向右读 w，遇到 (时，cnt 加一；遇到) 时，如果 cnt 为 0 则拒绝，否则 cnt 减一；
    3. 如果读完 w 时 cnt 为 0，则接受；否则拒绝。
- 如果 M 接受 w，则 w 中使得 cnt 第 k 次从 i 变成 $i+1$ 的 (与使得 cnt 第 k 次从 $i+1$ 变成 i 的) 配对，故 $w \in A$。如果 $w \in A$，由结构归纳可证明 M 接受 w：首先 M 接受 ε 和 ()；如果 M 接受 w_1 和 w_2，则 M 接受 (w_1) 和 w_1w_2。故 M 接受 $w \iff w \in A$。
- 由于 M 仅使用一个计数器，cnt 不超过 w 中 (的个数、从而不超过 w 的长度，故 M 仅需使用 $log(n)$ 的空间，其中 $n=|w|$。
- 即存在使用 $O(log(n))$ 空间的双带只读输入图灵机判定 A，故 $A \in$L。

1. 第 7 章：时间复杂性
2. 第 8 章：空间复杂性
3. 第 9 章：难解性
4. 第 10 章：复杂性理论高级专题

第 9 章习题

- 9.1 证明 $\mathrm{TIME}(2^n)$ =$\mathrm{TIME}(2^{n+1})$。
- $\mathrm{TIME}(2^n) = \{L|L$ 是被确定型图灵机在 $O(2^n)$ 时间内判定的语言 $\}$。
- $\mathrm{TIME}(2^{n+1}) = \{L|L$ 是被确定型图灵机在 $O(2^{n+1})$ 时间内判定的语言 $\}$。
- 由 $O(2^{n+1}) = O(2 \cdot 2^n) = O(2^n)$（大 O 记法可忽略常数因子），$\mathrm{TIME}(2^n)$ =$\mathrm{TIME}(2^{n+1})$。

第 9 章习题

- 9.2 证明 TIME(2^n) $\subsetneq$TIME(2^{2n})。
- 记 $t_1(n) = 2^n, t_2(n) = 2^{2n}$。
- 由 $lim_{n\to\infty}\frac{t_1(n)}{t_2(n)/logt_2(n)} = lim_{n\to\infty}\frac{2^n}{2^{2n}/2n} = lim_{n\to\infty}\frac{n}{2^{n-1}} = 0$，$t_1(n) = o(t_2(n)/logt_2(n))$。
- 在 $O(2^{2n})$ 的时间内，图灵机显然可以写一个 1 后面跟着 $2n$ 个 0；故 $t_2(n) = 2^{2n}$ 是时间可构造的。
- 由推论 9.11 可知 TIME($t_1(n)$) $\subsetneq$TIME($t_2(n)$)，即 TIME(2^n) $\subsetneq$TIME(2^{2n})。

第 9 章习题

- 9.3 证明 $\mathrm{NTIME}(n) \subsetneq \mathrm{PSPACE}$。
- 由 $\mathrm{NTIME}(n) = \{L|L$ 是被非确定型图灵机在 $O(n)$ 时间内判定的语言 $\}$，且 $O(n)$ 的时间最多使用 $O(n)$ 的空间，故 $\mathrm{NTIME}(n) \subseteq \mathrm{NSPACE}(n) = \{L|L$ 是被 $O(n)$ 空间的非确定型图灵机判定的语言 $\}$。
- 由萨维奇定理，$\mathrm{NSPACE}(n) \subseteq \mathrm{SPACE}(n^2)$。
- 由 $lim_{n\to\infty}\frac{n^2}{n^3} = lim_{n\to\infty}\frac{1}{n} = 0$，$n^2 = o(n^3)$。在 $O(n^3)$ 的空间内，图灵机显然可以写出 n^3 的二进制表示；故 n^3 是空间可构造的。由推论 9.4 可知 $\mathrm{SPACE}(n^2) \subsetneq \mathrm{SPACE}(n^3)$。
- 由 $\mathrm{PSAPCE} = \cup_k \mathrm{SPACE}(n^k)$，$\mathrm{SPACE}(n^3) \subseteq \mathrm{PSAPCE}$。
- 综上，$\mathrm{NTIME}(n) \subseteq \mathrm{NSPACE}(n) \subseteq \mathrm{SPACE}(n^2) \subsetneq \mathrm{SPACE}(n^3) \subseteq \mathrm{PSAPCE}$，即 $\mathrm{NTIME}(n) \subsetneq \mathrm{PSPACE}$。

第 9 章习题

- 9.9 证明若 NP=P^{SAT}，则 NP=coNP。
- 已知 coNP⊆P^{SAT}（NP⊆P^{SAT}，且 P^{SAT} 是一个确定型复杂性类、在补运算下封闭）；如果 NP=P^{SAT}，则 coNP⊆P^{SAT} =NP。只需证明 NP⊆coNP 即可。
- 任取语言 A ∈NP，由于 NP=P^{SAT}，即存在谕示 (确定型) 图灵机 M_{SAT} 在多项式时间内判定 A。设计谕示 (确定型) 图灵机 M'_{SAT}：在输入 w 上运行 M_{SAT}，如果 M_{SAT} 接受则拒绝，如果 M_{SAT} 拒绝则接受；则 $M'_{\overline{SAT}}$ 在多项式时间内判定 $\overline{A}$。因此 $\overline{A}$ ∈P^{SAT}=NP，进而 $A = \overline{\overline{A}}$ ∈coNP。
- 综上，如果 NP=P^{SAT}，则有 NP⊆coNP；又 coNP⊆P^{SAT} =NP，故有 NP=coNP。

第 9 章习题

- 9.11 在问题 7.19 中给出了语言 $MAX-CLIQUE$，证明 $MAX-CLIQUE \in \mathrm{P}^{SAT}$。
- $MAX-CLIQUE=\{\langle G,k\rangle \mid G$ 中最大团的大小恰好为 $k\}$。
- 不难发现 $MAX-CLIQUE=\{\langle G,k\rangle \mid \langle G,k\rangle \in CLIQUE$ 但 $\langle G,k+1\rangle \notin CLIQUE\}$，其中 $CLIQUE=\{\langle G,k\rangle \mid G$ 是包含 k 团的无向图 $\}$。
- 由定理 7.20 可知 $CLIQUE \in \mathrm{NP} \subseteq \mathrm{P}^{SAT}$，即 $\overline{CLIQUE} \in \mathrm{coNP} \subseteq \mathrm{P}^{SAT}$。设计谕示 (确定型) 图灵机 M_{SAT}：先判断 $\langle G,k\rangle$ 是否属于 $CLIQUE$，如果不是则拒绝；再判断 $\langle G,k+1\rangle$ 是否不属于 $CLIQUE$，如果是则接受，否则则拒绝。则 M_{SAT} 在多项式时间内判定 $MAX-CLIQUE$，因此 $MAX-CLIQUE \in \mathrm{P}^{SAT}$。

第 9 章习题

- 9.14 如问题 9.13 一样定义问题 $majority_n$。证明它可以用 $O(n)$ 规模的电路计算。
- 先证明可以用 $O(n)$ 规模的电路计算两个 n 位二进制整数的和（结果为 $n+1$ 位二进制整数）：令 $x = x_n ... x_2 x_1$，$y = y_n ... y_2 y_1$；对于 $i = 1, 2, ..., n$，依次计算 c_{i+1} 和 z_i，使得 $2(c_{i+1}) + z_i = x_i + y_i + c_i$（其中 $c_1 = 0$）；最后得到结果 $z = z_{n+1} z_n ... z_2 z_1$（其中 $z_{n+1} = c_{n+1}$）。由于每次计算 c_{i+1} 和 z_i 仅需常数个元件，故可以用 $O(n)$ 规模的电路计算出 z。
- 先填充使得 $2^{k-1} < n \leq n' = 2^k$，其中 n' 为填充后的长度：若 n 为奇数，填充时 0 的个数比 1 的个数多 1；若 n 为偶数，填充时 0 和 1 的个数一样多。采用分而治之的想法：先计算 2^{k-1} 次两个 1 位二进制整数的和，得到 2^{k-1} 个 2 位二进制整数；再计算 2^{k-2} 次两个 2 位二进制整数的和，得到 2^{k-2} 个 3 位二进制整数；……；最后计算 1 次两个 k 位二进制整数的和，得到 $k+1$ 位的结果 $z = \sum x_i$。

第 9 章习题

- 9.14 如问题 9.13 一样定义问题 $majority_n$。证明它可以用 $O(n)$ 规模的电路计算。
- （续）记 $z = z_{k+1}z_k...z_2z_1$。若 z_{k+1} 或 z_k 为 1，则 $\Sigma x_i \geq 2^{k-1} = n'/2$，输出 1；否则输出 0。
- 上述计算 Σx_i 的电路规模为 $2^{k-1}O(1) + 2^{k-2}O(2) + ... + 2O(k-1) + O(k) = \Sigma O(i2^{k-i}) = O(2^k) = O(n') = O(n)$。或记 $n' = 2^k$ 个输入时电路规模为 $S(n')$，则 $S(n') = 2S(n'/2) + O(k) = 2S(n'/2) + O(log(n'))$；由主定理可知 $S(n') = O(n') = O(n)$。
- 又由比较 Σx_i 和 $n'/2$ 的电路仅需常数个元件，故总的电路规模为 $O(n)$。即 $majority_n$ 可以用 $O(n)$ 规模的电路计算。

第 9 章习题

- 9.21 考虑函数 $pad : \Sigma^* \times \mathbb{N} \to \Sigma^*\#^*$ 定义如下：令 $pad(s, l) = s\#^j$，其中 $j = max(0, l - m)$，m 是 s 的长度。于是 $pad(s, l)$ 就是在 s 的末尾添加足够多的新符号 $\#$，使得结果的长度至少是 l。对于任何语言 A 和函数 $f : \mathbb{N} \to \mathbb{N}$，定义语言 $pad(A, f)$ 为：$pad(A, f) = \{pad(s, f(m)) | s \in A$, m 是 s 的长度$\}$。证明：若 $A \in$TIME(n^6)，则 $pad(A, n^2) \in$TIME(n^3)。
- 由于 $A \in$TIME(n^6)，存在 $O(n^6)$ 时间内判定 A 的确定型图灵机 M。构造 $O(n^3)$ 时间内判定 $pad(A, n^2)$ 的确定型图灵机 M_{pad}：
  M_{pad} = "对于输入 x：
  1. 判断 x 是否满足 pad 的形式（即是否存在 $w \in \Sigma^*$，使得 $x = pad(w, |w|^2)$），若不满足则拒绝；
  2. 在 w 上运行 M，如果 M 接受则接受，如果 M 拒绝则拒绝。"
- 如果 M_{pad} 接受 x，则 $x = pad(w, |w|^2)$ 且 $w \in A$，即 $x \in pad(A, n^2)$；反之依然。故 M_{pad} 接受 $x \iff x \in pad(A, n^2)$。
- 由于步骤 1 至多在 $O(|x|^2)$ 的时间内检查 x 的形式（区分出 w 和 $\#^*$，再判断 $|x|$ 与 $|w|^2$ 是否相等），步骤 2 在 $O(|w|^6) = O(|x|^3)$ 的时间内完成运行，故 M_{pad} 在 $O(|x|^2) + O(|x|^3) = O(|x|^3)$ 的时间内完成判定。
- 即判定 M_{pad} 有时间为 $O(n^3)$ 的算法，故 $M_{pad} \in$TIME(n^3)。

第 9 章习题

- 9.22 证明：若 NEXPTIME≠EXPTIME，则 P≠NP。你会发现问题 9.21 中定义的函数 pad 对证明本问题是有用的。
- 方法一：如果 NEXPTIME≠EXPTIME，由于 EXPTIME⊆ NEXPTIME，即存在语言 A，有指数时间内判定 A 的非确定型图灵机 N，但不存在指数时间内判定 A 的确定型图灵机。
- 设 N 能在 $O(2^{n^c})$ 时间内判定 A，考虑语言 $A_{pad} = \{pad(w, 2^{|w|^c}) | w \in A\}$。与 9.21 类似，可以构造多项式时间内判定 A_{pad} 的非确定型图灵机 N_{pad}；因此 A_{pad} ∈NP。
- 下面用反证法说明 A_{pad} ∉P：假设 A_{pad} ∈P，则存在多项式时间内判定 A_{pad} 的确定型图灵机 M_{pad}。设计确定型图灵机 M：将输入 w 填充为 $pad(w, 2^{|w|^c})$，然后运行 M_{pad} 决定接受与否；则 M 在输入长度 $|w|$ 的指数时间能判定 A。与 A ∉EXPTIME 矛盾，故假设不成立。
- 即找到语言 A_{pad}，A_{pad} ∈NP 且 A_{pad} ∉P；故 P≠NP。

第 9 章习题

- 9.22 证明：若 NEXPTIME≠EXPTIME，则 P≠NP。你会发现问题 9.21 中定义的函数 *pad* 对证明本问题是有用的。
- 方法二：考虑证明其**逆否命题**“若 P=NP，则 NEXPTIME=EXPTIME”。由于 EXPTIME⊆NEXPTIME，只需证明 NEXPTIME⊆EXPTIME 即可。
- 任取语言 A ∈NEXPTIME，设非确定型图灵机 N 能在 $O(2^{n^c})$ 时间内判定 A，考虑语言 $A_{pad} = \{pad(w, 2^{|w|^c}) | w \in A\}$。与 9.21 类似，可以构造多项式时间内判定 A_{pad} 的非确定型图灵机 N_{pad}；因此 A_{pad} ∈NP。
- 由于 P=NP，存在确定型图灵机 M_{pad} 在多项式时间内判定 A_{pad}。设计确定型图灵机 M：将输入 w 填充为 $pad(w, 2^{|w|^c})$，然后运行 M_{pad} 决定接受与否；则 M 在输入长度 $|w|$ 的指数时间能判定 A。因此 A ∈EXPTIME。
- 综上，如果 P=NP，则有 NEXPTIME⊆EXPTIME，即 NEXPTIME=EXPTIME；故原命题成立。

第 9 章习题

- 9.24 证明 $TQBF \notin \mathrm{SPACE}(n^{1/3})$。
- 先证明一个比较强的结论：PSPACE 中的每一个语言 A 都**对数空间**可归约到 $TQBF$。
- 回顾证明 $TQBF$ 是 PSPACE 完全的过程：任取 PSPACE 中的语言 A、有图灵机 M 在 $O(n^k)$ 空间内判定 A。由于 M 在长为 n 的输入上格局数为 $2^{O(n^k)}$，尝试将问题转换成 $\phi_{c_{start},c_{accept},h}$，其中 h 为 $2^{O(n^k)}$。递归化简 $\phi_{c_1,c_2,t} = \exists m_1$ $\forall(c_3, c_4) \in \{(c_1, m_1), (m_1, c_2)\}[\phi_{c_3,c_4,t/2}]$。又 $\phi_{c_1,c_2,1}$ 容易构造，从而将 A 多项式时间归约到 $TQBF$。
- 说明上述过程可在对数空间内完成：由于递归过程中 t 每次下降为 $t/2$，只需记录 $log(t) = O(n^k)$、需要 $log(O(n^k)) = O(log(n))$ 的空间。递归过程中不需要显示写出表达式化简的过程，只需输出增加的部分 $\exists m_1 \forall(c_3, c_4) \in \{(c_1, m_1), (m_1, c_2)\}$，然后将计数器 t 变为 $t/2$（$log(t)$ 减 1）即可；直至 t 减少为 1。
- 综上，证明了 PSPACE 中所有语言都对数空间可归约到 $TQBF$。

第 9 章习题

- 9.24 证明 $TQBF \notin \text{SPACE}(n^{1/3})$。
- 再回顾一个结论：若 $A \in$PSPACE 是有图灵机 M 在 $O(n^k)$ 空间内判定的语言，上述将 A 归约到 $TQBF$ 的过程中，由于递归每一层增加的公式的长度与格局的长度呈线性关系，为 $O(n^k)$；递归的层数为 $O(n^k)$，所以得到的公式长度为 $O(n^{2k})$。
- 现在假设 $TQBF \in \text{SPACE}(n^{1/3})$。由空间层次定理，对于任意 $\varepsilon > 0$，存在语言 A，A 可在 $O(n^{1+\varepsilon})$ 空间内判定、但不能在 $O(n)$ 的空间内判定。由于 $A \in$PSPACE，A 对数空间可归约到 $TQBF$ 且公式长度为 $O(n^{2(1+\varepsilon)})$。由于 $TQBF \in \text{SPACE}(n^{1/3})$，将 A 归约到 $TQBF$ 再进行判定仅需 $O(n^{2(1+\varepsilon)/3} + log(n))$ 的空间。进而 $\varepsilon \leq 1/2$ 时 A 可在 $O(n)$ 的空间内判定，矛盾；故假设不成立，$TQBF \notin \text{SPACE}(n^{1/3})$。

1. 第 7 章：时间复杂性
2. 第 8 章：空间复杂性
3. 第 9 章：难解性
4. 第 10 章：复杂性理论高级专题

第 10 章习题

- 10.2 证明：12 不能通过费马测试，从而不是伪素数。
- $2^{(12-1)} = 2^{11} = 2048$，$2048\ mod\ 12 = 8$，得到的结果不是 1。由此可说明 12 一定不是素数；即 12 没有通过费马测试，不是伪素数。

第 10 章习题

- 10.3 证明：如果 $A \leq_L B$ 且 B 在 NC 中，则 A 也在 NC 中。
- 由 $B \in$NC，设 $B \in$NC^k，即存在多项式规模和 $O((log(n))^k)$ 深度的一致电路族识别 B。
- 又 $A \leq_L B$，且由定理 10.37 可知 L$\subseteq$NL$\subseteq$NC^2，故存在 NC^2 中的计算函数 f，使得 $f(x) \in B \iff x \in A$。
- 因而 A 可由多项式规模和 $O((log(n))^{k+2})$ 深度的一致电路族识别，即 $A \in$NC$^{k+2} \subseteq$NC。

第 10 章习题

- 10.4 证明：有 n 个输入的奇偶函数能用 $O(n)$ 个顶点的分支程序计算。
- 初始时查询变量 x_1，如果为 0 则当前有偶数个 1，如果为 1 则当前有奇数个 1。依次查询变量 $x_2, x_3, \ldots, x_n$，根据当前 1 个数的奇偶性以及 x_i 的值记录查询 x_i 后 1 个数的奇偶性：如果当前有偶数/奇数个 1 且 x_i 为 0/1，则查询 x_i 后有偶数个 1；如果当前有奇数/偶数个 1 且 x_i 为 0/1，则查询 x_i 后有奇数个 1。最后，根据查询 x_n 后 1 个数的奇偶性决定输出 0 还是 1：查询 x_n 后有偶数个 1，则输出 0；有奇数个 1，则输出 1。
- 上述的分支程序仅使用 $1 + 2(n-1) = 2n - 1$ 个查询顶点和 2 个输出顶点，总顶点数为 $O(n)$。故有 n 个输入的奇偶函数能用 $O(n)$ 个顶点的分支程序计算。

第 10 章习题

- 10.5 证明：有 n 个输入的多数函数能用 $O(n^2)$ 个顶点的分支程序计算。
- 初始时查询变量 x_1，如果为 0 则当前有 0 个 1，如果为 1 则当前有 1 个 1。依次查询变量 $x_2, x_3, ..., x_n$，根据当前 1 的个数以及 x_i 的值记录查询 x_i 后 1 的个数：如果当前有 $k(0 \le k \le i-1)$ 个 1 且 x_i 为 0，则查询 x_i 后仍有 k 个 1；如果当前有 $k(0 \le k \le i-1)$ 个 1 且 x_i 为 1，则查询 x_i 后有 $k+1$ 个 1。最后，根据查询 x_n 后 1 的个数决定输出 0 还是 1：查询 x_n 后 1 的个数 $< n/2$，则输出 0；1 的个数 $\ge n/2$，则输出 1。
- 上述的分支程序仅使用 $1+2+...+n = n(n-1)/2$ 个查询顶点和 2 个输出顶点，总顶点数为 $O(n^2)$。故有 n 个输入的多数函数能用 $O(n^2)$ 个顶点的分支程序计算。
- 在上述的分支程序中，某些顶点可以被删去：如果当前 1 的个数已经 $\ge n/2$，则可直接输出 1；如果 1 的个数已经不可能 $\ge n/2$（例如 $n=5$ 时查询 x_3 后仅有 0 个 1），则可直接输出 0。

第 10 章习题

- 10.6 证明：任何有 n 个输入的函数都能用 $O(2^n)$ 个顶点的分支程序计算。
- 依次查询变量 $x_1, x_2, ..., x_n$，由于每个变量有 2 种取值、初始仅有 1 种情况，查询 x_i 前共有 2^{i-1} 种情况、查询 x_i 后共有 2^i 种情况。最后，根据查询 x_n 后的 2^n 种情况决定输出 0 还是 1 即可。
- 上述分支程序可计算任何有 n 个输入的函数，使用 $1 + 2 + 4 + ... + 2^{n-1} = 2^n - 1$ 个查询顶点和 2 个输出顶点，总顶点数为 $O(2^n)$。故任何有 n 个输入的函数都能用 $O(2^n)$ 个顶点的分支程序计算。

第 10 章习题

- 10.8 如果 BPL 是概率对数空间图灵机以错误概率 $\frac{1}{3}$ 判定的语言集合，证明 BPL$\subseteq$P。
- 思路：如果能在多项式时间内模拟出所有计算分支的概率，计算接受分支的概率和，即可在多项式时间内判定该语言。
- 任取语言 $A\in$BPL、图灵机 M 仅需对数空间以错误概率 $\frac{1}{3}$ 判定 A。构造如下的图灵机 M'：对于输入 x，考虑在 x 上运行 M 的所有格局 C；得到矩阵 $P_{C\times C}$，其中 $P(c_i, c_j)$ 为从格局 c_i 经过一步到达格局 c_j 的概率（如果从 c_i 掷硬币的选择之一是到 c_j，则 $P(c_i, c_j) = 1/2$；否则 $P(c_i, c_j) = 0$）；由于 P 的 k 次幂表示经过 k 步从格局 c_i 到格局 c_j 的概率，计算出 P 的所有幂次即可得到所有接受分支的概率、加起来便可得到接受概率；如果接受概率 $\geq 2/3$，说明 M 接受 x，则接受，如果接受概率 $< 1/3$，说明 M 拒绝 x，则拒绝。
- 由于对数空间对应的格局总数不超过多项式，因而得到 P、计算 P 的所有幂次、得到所有接受分支的概率、计算接受概率的过程均可在多项式时间内完成。故 M' 的运行时间不超过输入长度的多项式，即 $A\in$P。进而有 BPL$\subseteq$P。

第 10 章习题

- 10.11 证明：如果 NP⊆BPP，则 NP=RP。
- RP⊆NP：任取语言 A ∈RP，M 为对应的概率图灵机；仅需将掷硬币的非确定性步改为非确定型图灵机的不同分支；对于输入 x，如果没有分支接受，则 $x \notin A$，如果有分支接受，则 x 不可能 $\notin A$、$x \in A$；故 A ∈NP。
- 只需证明 NP⊆BPP 时，NP⊆RP。由于 NP 中的所有语言都能在多项式时间内归约到 SAT，证明 SAT ∈RP 即可证明 NP⊆RP。
- 由于 NP⊆BPP，存在图灵机 M_{SAT} 在多项式时间内以错误概率 $p(p>0)$ 判定 SAT。构造如下的图灵机 M'_{SAT}：对于含有变量 $x_1, x_2, ..., x_n$ 的公式 ϕ，在 ϕ 上运行 M_{SAT}，如果 M_{SAT} 拒绝则拒绝；否则将 x_1 取值为 0、在赋值后的 ϕ 上运行 M_{SAT}，如果 M_{SAT} 接受则固定 x_1 的取值为 0，否则固定 x_1 的取值为 1；对 $i=2,...,n$ 依次重复上述过程，当所有变量取值确定后 ϕ 为真则接受、ϕ 为假则拒绝。

第 10 章习题

- 10.11 证明：如果 NP⊆BPP，则 NP=RP。
- （续）如果 $\phi \notin SAT$，则 ϕ 要么在一开始就被 M'_{SAT} 拒绝、要么在最后被 M'_{SAT} 拒绝，总之一定会被 M'_{SAT} 拒绝。
- 如果 $\phi \in SAT$，则 M'_{SAT} 一开始就拒绝 ϕ 的概率为 p；到给 x_i 赋值的环节，赋值后 ϕ 不可满足的概率至多为 p（无论是 x_i 为 0 时不可满足还是 x_i 为 1 时不可满足）。因而 M'_{SAT} 接受 ϕ 的概率超过 $(1-p)^{n+1} \geq 1-(n+1)p$，只要 $p \leq \frac{1}{2(n+1)}$ 即可保证接受概率超过 $\frac{1}{2}$。由引理 10.5，在多项式时间内错误概率可以达到指数量级的下降，因而存在与 M_{SAT} 等价、错误概率 $\leq \frac{1}{2(n+1)}$ 的多项式时间概率图灵机。
- 综上，由图灵机 M'_{SAT} 可知 $SAT \in$RP。命题得证。

Thanks!