B0911005Y-01: Introduction to Theory of Computation **2023 Spring**

Homework 7 — May 26

Lecturer: Mingji Xia *Completed by:* 吉骏雄

第 7.1 次作业：7.1abef，7.2abef，7.4，7.6，7.8，7.10

7.1 下面各项是真是假？

(a) $2n = \mathrm{O}(n)$

(b) $n^2 = \mathrm{O}(n)$

(e) $3^n = 2^{\mathrm{O}(n)}$

(f) $2^{2^n} = \mathrm{O}(2^{2^n})$

解

(a) 真

(b) 假

(e) 真

(f) 真

7.2 下面各项是真是假？

(a) $n = \mathrm{o}(2n)$

(b) $2n = \mathrm{o}(n^2)$

(e) $n = \mathrm{o}(\log n)$

(f) $1 = \mathrm{o}(1/n)$

解

(a) 假

(b) 真

(e) 假

(f) 假

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绘的表：

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

解 表格如下: 右上角有 S, 所以可以生成串 $w = baba$.

	1	2	3	4
b 1	T	$R \mid T$	S	$S \mid R \mid T$
a 2		R	S	S
b 3			T	$R \mid T$
a 4				R

7.6 证明 P 在并、连接和补运算下封闭。

证明 并运算:

$\forall L_1, L_2 \in P$, 根据定义, 存在两个图灵机 M_1, M_2 分别能够判定 L_1, L_2, 且所需时间 $f_1(n), f_2(n)$ 均为多项式时间. 我们构造一个图灵机 M':

M' = "对于输入 ω, 其中 ω 是字符串:

1. 将输入复制一份在带上输入右面, 左面的原输入暂时封存, 用不会出现在 M_1, M_2 带字符表的符号将之分隔开.
2. 使用右侧复制的内容, 模拟在输入 ω 上运行 M_1.
3. 如果 M_1 接受, 则接受; 如果 M_1 拒绝, 则清空分隔符右侧的内容, 返回原输入状态.
4. 模拟在输入 ω 上运行 M_2.
5. 如果 M_2 接受, 则接受; 如果 M_2 拒绝, 则拒绝. "

可以看到, M' 的每一步都是多项式时间的, 且多步顺序执行. M' 能判定 $L_1 \cup L_2$, 因此 $L_1 \cup L_2 \in P$. 因此 P 在并运算下封闭.

连接运算:

假设同上. 我们构造一个图灵机 M':

M' = "对于输入 ω, 其中 ω 是字符串:

1. 将输入复制一份在带上输入右面, 左面的原输入暂时封存, 用不会出现在 M_1, M_2 带字符表的符号将之分隔开.
2. 遍历每一种将 ω 分为两个前后连接的字符串 ω_1, ω_2 的可能性. 在每一种可能性中, 使用另一个不会出现在 M_1, M_2 带字符表的符号将之分隔开. 此时带上最后的字符串为 ω_2.

3. 模拟在输入 ω_2 上运行 M_2.

4. 如果 M_2 接受, 则清空 ω_2, 继续下一步; 如果 M_2 拒绝, 则恢复输入, 开启下一步循环.

5. 模拟在输入 ω_1 上运行 M_1.

6. 如果 M_2 接受, 则接受; 如果 M_2 拒绝, 则恢复输入, 开启下一步循环. ”

可以看到, M' 的每一步都是多项式时间的, 且循环最多进行 $n+1$ 次, 复杂多也是多项式级别的 (因为是 $n \cdot \mathrm{O}(f_1(n) + f_2(n) + p(n))$, 其中 $p(n)$ 为一个多项式, 是其他步骤计算花费的时间). M' 能判定 L_1L_2, 因此 $L_1L_2 \in P$. 因此 P 在连接运算下封闭.

补运算:

$\forall L \in P$, 根据定义, 存在图灵机 M 能够判定 L, 且所需时间 f 为多项式时间. 我们构造一个图灵机 M', 其与 M 的区别仅为接受状态与拒绝状态互换, 即 q_{accept} 与 q_{reject} 互换. 这样构造的图灵机依然可以对任意输入停机, 所需的计算时间一致, 且接受的语言恰为 $\overline{L}$. 因此 P 在补运算下封闭. □

7.8 令 CONNECTED = $\{\langle G\rangle \mid G$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。

证明 书中给出一种图灵机的构造:

$M =$ “输入是图 G 的编码 $\langle G\rangle$:

1. 选择 G 的第一个顶点，并标记之。

2. 重复下列步骤，直到没有新的顶点可作标记。

3. 对于 G 的每一个顶点，如果能通过一条边将其连到另一个已被标记的顶点，则标记该顶点。

4. 扫描 G 的所有顶点，确定它们是否都已作了标记。如果是，则接受，否则拒绝。”

在此构造中, 我们不难发现, 循环不超过 n 次 (因为字符串长度肯定超过图的顶点个数); 每次循环即便检查其他所有边, 也不会检查超过 n 次 (边数同样小于字符串长度); 最多标记 n 个顶点, 每标记一次顶点, 肯定在多项式时间内能够解决 (譬如将剩余字符串后移, 在顶点后添加固定长度标记, 这样的复杂度是 $\mathrm{O}(n)$); 而且这个图灵机可以停机. 综合来看, 这个图灵机总是能在多项式时间内完成判定. 所以此语言属于 P. □

7.10 证明 $ALL_{DFA} \in P$.

证明 构造图灵机 M 判定 ALL_{DFA}, 如下:

$M' =$ “对于输入 $\langle A\rangle$, 其中 $\langle A\rangle$ 应当是 DFA 的编码:

1. 判断 A 是否是一台 DFA, 如果不是则拒绝.

2. 寻找 A 中是否有拒绝状态 q_{reject}. 如果没有, 则接受; 如果有, 重复下一步骤，直到没有新的顶点可作标记.

3. 对于 A 的每一个状态, 如果能转换到另一个已被标记的状态, 则标记该状态.

4. 检查 A 的初始状态 q_0, 确定它是否已作了标记. 如果是, 则拒绝, 否则接受. ”

因为一台 DFA 只要有能够从初始状态进入的拒绝状态, 那么一定会对某些字符串拒绝, 因此这台图灵机能够判定语言 ALL_{DFA} □

第 7.2 次作业：7.5，7.7，7.12，7.35. 选做：7.13（较难），7.42

7.5 下面的公式是可满足的吗？

$$(x \vee y) \wedge (x \vee \overline{y}) \wedge (\overline{x} \vee y) \wedge (\overline{x} \vee \overline{y})$$

解 不是. 化简:

$$\begin{aligned} & (x \vee y) \wedge (x \vee \overline{y}) \wedge (\overline{x} \vee y) \wedge (\overline{x} \vee \overline{y}) \\ = & [(x \vee y) \wedge (x \vee \overline{y})] \wedge [(\overline{x} \vee y) \wedge (\overline{x} \vee \overline{y})] \\ = & x \wedge \overline{x} \\ = & 0 \end{aligned}$$

因此不可满足.

7.7 证明 NP 在并和连接运算下封闭。

证明 给出两种识别非确定型图灵机的构造如下:

(1) 并运算: 任意给定两个语言 $A, B \in$ NP, 那么存在两个非确定型图灵机 M_1, M_2 能够在多项式时间内判定 A, B. 我们构造一个非确定型图灵机 M': "对于输入 ω, 其中 ω 是字符串:

(a) 以 ω 为输入, 模拟运行 M_1. 如果接受, 则继续下一步; 如果拒绝, 则拒绝.

(b) 以 ω 为输入, 模拟运行 M_2. 如果接受, 则接受; 如果拒绝, 则拒绝.

"

(2) 连接运算: 任意给定两个语言 $A, B \in$ NP, 那么存在两个非确定型图灵机 M_1, M_2 能够在多项式时间内判定 A, B. 我们构造一个非确定型图灵机 M': "对于输入 ω, 其中 ω 是字符串:

(a) 将原输入封存, 用不会出现在 M_1, M_2 带字符表的符号 (以 # 为例) 做标记.

(b) 在原输入的最末端添加一个不会出现在 M_1, M_2 带字符表的符号 (以 * 为例) 做标记.

(c) 重复下 2 个步骤, 直到无法继续:

(d) 将 * 左侧的原始输入复制到右侧, 作为新的输入, 模拟运行 M_1. 若接受, 则继续下一步; 若拒绝, 则继续回到第 2 步.

(e) 将 * 右侧的原始输入复制到左侧, 作为新的输入, 模拟运行 M_2. 若接受, 则接受; 若拒绝, 则继续回到第 2 步.

(f) 若 * 抵达原输入 ω 的最左侧, 即便使用空串运行 M_1 也不能接受, 则拒绝.

" 由于中间的重复/计算都是多项式时间内可计算的, 于是这两个新的非确定型图灵机也是多项式时间内可计算的. 因此 NP 在并和连接运算下封闭.

□

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 (isomorphic) 。令 ISO $= \{\langle G, H\rangle \mid G$ 和H 是同构的图$\}$。证明 ISO $\in$ NP.

证明 任意给定两个图 G, H, 我们构造一个非确定型图灵机 M: "对于输入 $\langle G, H\rangle$, 其中 G, H 是图的编码:

1. 检查 G, H 的结点数, 边数是否相同. 如果不同, 则拒绝.
2. 对每一个 G 的结点 v, 非确定地选择一个 H 的结点 w, 并记录. 这共要进行 n 次选择, 其中 n 是 G 的结点数.
3. 对每一对选中的结点 v, w, 检查 G 中 v 的邻接结点是否与 H 中 w 的邻接结点相同, 即它们对其他每一对结点, 是否同时有/没有边分别与之相连. 这对每对结点来说, 共要进行 n 次检查, 其中 n 是 G 的结点数. 如果相同, 则接受. 如果不同, 则拒绝. "

"

这个非确定型图灵机的多项式时间性质是显然的, 因为共需要进行 $n+n^2$ 次多项式时间内即可完成的可计算操作, 且 $n < |\langle G, H\rangle|$. 于是 ISO $\in$ NP. □

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。）

证明 任意给定一个布尔公式 ϕ, 我们构造一个非确定型图灵机 M: "对于输入 ϕ, 其中 ϕ 是布尔公式:

1. 列出 ϕ 中的所有变量.
2. 非确定地对 ϕ 中的每一个变量进行赋值, 并记录. 这共要进行 n 次选择, 每次选择在 $0, 1$ 中进行, 其中 n 是 ϕ 的变量数.
3. 调用 SAT 的验证器, 验证 ϕ 在这组赋值下是否可满足. 如果是, 则接受. 如果不是, 则拒绝.

"

列出变量显然是多项式时间内可解的. 非确定地对每一个变量进行赋值, 也是多项式时间内可解的, 因为任意一条计算分支只需要赋值 n 次, 且 $n < |\langle\phi\rangle|$, 每次赋值的时间是多项式的. 调用 SAT 的验证器也只需要多项式时间, 因为 P = NP 且 SAT $\in$ NP, 因此 SAT $\in$ P. 因此这个非确定型图灵机的多项式时间性质是显然的. 于是 SAT $\in$ NP. □

第 7.3 次作业：7.29，7.39，7.45，7.48. 选做：7.38.

7.29 令 SET-SPLITTING = $\{\langle S, C\rangle \mid S$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。

证明

1. 首先, 可以证明是 SET-SPLITTING $\in$ NP. 给定一个 $\langle S, C\rangle$, 我们构造一个非确定型图灵机 M: "对于输入 $\langle S, C\rangle$, 其中 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, $k > 0$:

   (a) 非确定地对 S 中的每一个元素进行染色, 并记录. 这共要进行 $|S|$ 次选择, 每次选择在红/蓝中进行.

   (b) 对于每一个 C_i, $i \in \{1, \ldots, k\}$, 验证 C_i 中的元素不会被染成同一种颜色. 如果不是, 则拒绝.

   (c) 如果均不会被染成同一种颜色, 则接受.

   "

   在这个构造中, 我们显然仅使用了多项式时间就能完成问题的解决. 因此 SET-SPLITTING $\in$ NP.

2. 通过证明 3-SAT $\leq_p$ SET-SPLITTING 来证明 SET-SPLITTING 是 NP 完全的. 构造一个图灵机 M: "对于输入 ϕ, 其中 ϕ 是 3CNF-布尔公式:

   (a) 列出 ϕ 中的所有变量, 构成集合 V.

   (b) 构建集合 $V' = \{a_i \mid a \in V,\ i\ \in \{0, 1\}\}$

   (c) 构造集合 $S = V' \cup \{s\}$, 其中 $s \notin V \cup V'$ 是一个新的变量.

   (d) 再构造集合 $A = \{\{a_i, b_j, c_k, s\} \mid a_i, b_j, c_k \in V';\ \forall x \in V$, 我们取$x_0 = x, x_1 = \overline{x}$, 且在这种赋值下, ϕ 中存在一个子句, 形如$a_i \vee b_j \vee c_k\}$, 以及集合 $B = \{\{a_0, a_1\} \mid a \in V\}$. 取 $C = A \cup B$.

   (e) 以 $\langle S, C\rangle$ 为输入, 模拟运行 SET-SPLITTING. 若接受, 则接受; 若拒绝, 则拒绝.

   "

   第 5 步会得到 S 中是否存在满足条件的一种染色, 能让 C 中所有元素 (每个元素都是一个 S 的子集), 其内的元素 (也就是 S 中的一些元素) 获得的染色均不同. A 中集合染色不同, 说明至少有一个元素与 s 的颜色不同; B 中染色不同, 说明 a_0, a_1 的颜色不同. 这样, 我们不妨以与 s (s 表示 standard) 颜色不同作为某个变量的赋值为 1/0 的标志: 若 a_1 与 s 颜色不同, 即 a_0 与 s 颜色相同, 这代表了 a 被赋值为 1; 若 a_1 与 s 颜色相同, 即 a_0 与 s 颜色不同, 这代表了 a 被赋值为 0. 根据这样的意义表示, A 限制 ϕ 的每一个子句都是可满足 (存在一个为 1 的文字), B 要求了 ϕ 的每一个变量的正确赋值.

   这样, 我们用 SET-SPLITTING 解决了 3-SAT, 这个归约成立. 并且很显然, 这个归约中的函数是多项式时间内即可计算的: 所有变量数少于输入字符串 $|\langle\phi\rangle|$ 长度, 构造集合的难度也都在 $O(|V|^3)$ 范围内. 因此, SET-SPLITTING 是 NP 完全的.

□

7.39 如图 7.1, 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，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 = $\{\langle c_1, \ldots, c_k\rangle \mid$ 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。

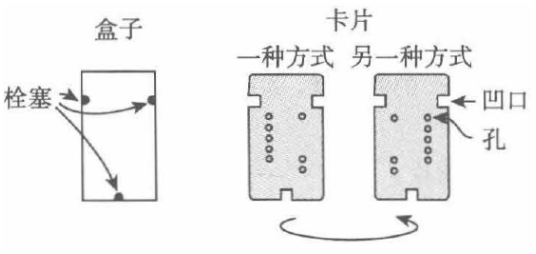

图 7.1. 题 7.39 图

证明

1. 首先, 可以证明是 PUZZLE $\in$ NP. 给定一个 $\langle c_1, \ldots, c_k\rangle$, 我们构造一个非确定型图灵机 M: "对于输入 $\langle c_1, \ldots, c_k\rangle$, 其中每个 c_i 代表一张卡片:

   (a) 非确定地对每一个卡片进行放置, 并记录. 这共要进行 k 次选择, 每次选择在两种放置方式 (对称的两种) 中进行.

   (b) 全部放好后, 检查盒子的底是否被完全覆盖. 如果是, 则接受. 如果不是, 则拒绝.

   "

   这个方法中, 我们显然仅使用了多项式时间就能完成问题的解决. 因此 PUZZLE $\in$ NP.

2. 再证明 3-SAT $\leq_p$ PUZZLE. 构造一个图灵机 M: "对于输入 ϕ, 其中 ϕ 是 3CNF-布尔公式:

   (a) 列出 ϕ 中的所有变量, 构成集合 V, 并排序. 设 ϕ 中有 n 个子句.

   (b) 将 $\overline{\phi}$ 转写为 3DNF 的析取式, 比如 $\phi = (x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_2 \vee x_4)$, 则对应的 $\overline{\phi} = \overline{(x_1 \vee x_2 \vee \overline{x}_3)} \vee \overline{(\overline{x}_1 \vee x_2 \vee x_4)} = (\overline{x}_1 \wedge \overline{x}_2 \wedge x_3) \vee (x_1 \wedge \overline{x}_2 \wedge \overline{x}_4)$

   (c) 考虑 V 中的所有变量, 构成 $|V|$ 张卡片, 第 i 张卡片的内容如下. 从上向下共有 n 对孔, 其中第 j 对孔, 代表 V 中第 i 个元素 v_i 在 $\overline{\phi}$ 的第 j 个子句中的出现情况. 若未出现, 则有两个孔; 若出现 v_i, 则左侧有一个孔; 若出现 $\overline{v}_i$, 则右侧有一个孔 (合理地, 同时出现即可化简作未出现).

   例如, 若 $\overline{\phi} = (\overline{x}_1 \wedge \overline{x}_2 \wedge x_3) \vee (x_1 \wedge \overline{x}_2 \wedge \overline{x}_4)$, 则:

   i. 第 1 张卡片的内容为 "自上而下, 第一处为右侧的孔, 第二处为左侧的孔",

   ii. 第 2 张卡片的内容为 "自上而下, 第一处为右侧的孔, 第二处为右侧的孔",

   iii. 第 3 张卡片的内容为 "自上而下, 第一处为左侧的孔, 第二处为一对孔",

   iv. 第 4 张卡片的内容为 "自上而下, 第一处为一对孔, 第二处为右侧的孔".

这样的话, 如果四张卡片均正放, 就可以发现左侧一列两个孔洞均为封闭状态 (右侧我们并不关心). 这说明, 存在一组赋值使得 $\overline{\phi}$ 的任意子句都是 0 (以卡片开孔代表对应文字值为 1, 孔洞位置开口对应子句值为 1; 而封闭为 0), 进而 $\overline{\phi} = 0, \phi = 1$. 这样, 我们就证明了 ϕ 是可满足的.

(d) 我们还需要检查左侧一列的所有洞是否都为封闭状态. 只要再加入一张卡牌 v_0, 其左侧一列均为封闭, 右侧一列均开孔, 与之前的 $|V|$ 张卡片一起形成卡片集合 $\{v_0, v_1, \ldots, v_k\}$. 由于对称性, 我们不妨认为 v_0 是正放的, 因为即便其反放, 我们也有一种相反的赋值方法, 使得问题存在一种对称的解.

(e) 以 $\langle v_0, v_1, \ldots, v_k \rangle$ 作为输入, 模拟运行 PUZZLE. 若接受, 则接受; 若拒绝, 则拒绝.

上文中的所有计算过程均为多项式时间可计算的. 因此, 3-SAT $\leq_p$ PUZZLE. 我们证明了 PUZZLE 是 NP 完全的.

□

7.45 证明若 P = NP, 则除了语言 $A = \varnothing$ 和 $A = \Sigma^*$ 以外，所有语言 $A \in$ P 都是 NP 完全的。

证明 首先, 显然, P $\in$ NP. 所以除了语言 $A = \varnothing$ 和 $A = \Sigma^*$ 以外, 所有语言 $A \in$ P 都是属于 NP 的.

然后, 对于所有非平凡的语言 $A \in$ P, 有 $A \in$ NP 根据题目假设. 那么根据定义, 对于任意语言 $B \in$ NP, 都有 $B \leq_p A$. 这件事其实很显然, 因为根据题目假设有 $B \in$ P, 因此只需要用多项式时间计算出某个输入字符串 ω 是否在 B 中, 然后将根据结果, 将一个是或不是在 A 中的串作为输入, 运行 A 本身, 就能调用 A 计算 B. 构造图灵机如下: "对于输入 ω, 其中 ω 是字符串:

1. 事先准备好两个字符串 ω_a, ω_r, 其中 $\omega_a \in A, \omega_r \notin A$.
2. 用多项式时间计算出 ω 是否在 B 中. 如果是, 则以 ω_a 作为下一问的输入; 如果不是, 则以 ω_r 作为下一问的输入.
3. 将上一问选定的字符串作为输入, 运行 A. 如果接受, 则接受; 如果拒绝, 则拒绝.

因此, $B \leq_p A$. 这说明, A 是 NP 完全的. □

7.48 设 G 表示无向图，令

$$\text{SPATH} = \{\langle G, a, b, k \rangle \mid G \text{ 包含从} a \text{ 到} b\text{、长度至多为} K \text{ 的简单路径}\}$$

以及

$$\text{LPATH} = \{\langle G, a, b, k \rangle \mid G \text{ 包含从} a \text{ 到} b\text{、长度至少为} K \text{ 的简单路径}\}$$

a. 证明 SPATH $\in$ P。

b. 证明 LPATH 是 NP 完全的。

证明

a. 构造图灵机 M 能够识别 SPATH: "对于输入 $\langle G, a, b, k \rangle$, 其中 G 是无向图, a, b 是 G 的两个顶点, k 是一个正整数:

(a) 从 a 开始, 以 1 为每条边的权重, 使用 Dijkstra 算法, 计算出从 a 到 b 的最短路径长度 d. 如果 $d \leq k$, 则接受; 如果 $d > k$, 则拒绝.

”

Dijkstra 算法的时间复杂度是 $\mathrm{O}(n^2)$, 因此这个图灵机的多项式时间性质是显然的. 因此 SPATH $\in$ P.

b. 首先证明 LPATH $\in$ NP. 构造非确定型图灵机 M 能够识别 LPATH: “对于输入 $\langle G, a, b, k\rangle$, 其中 G 是无向图, a, b 是 G 的两个顶点, k 是一个正整数:

(a) 从 a 开始, 记录一条路径. 重复下一步骤 $n-1$ 次, 直到选到顶点 b 或没有任何相邻的顶点可选.

(b) 在路径最前方的顶点处, 非确定地确定其任意一个未被选进路径中的相邻顶点, 并将其加入路径中. 每次一共有至多 $n-m$ 种选择, 其中 m 是路径中已有的顶点数. 将这个顶点视作路径最前方的结点.

(c) 如果选取到顶点 b, 且得到的路径长度至少为 k, 则接受; 否则, 则拒绝.

”

这个非确定型图灵机的多项式时间性质是显然的. 因此 LPATH $\in$ NP.

再来证明 UHAMPATH $\leq_p$ LPATH. 构造一个图灵机 M: “对于输入 $\langle G, s, t\rangle$, 其中 G 是无向图, s, t 是 G 的两个顶点:

(a) $G = (V, E)$, 计算 $|V| = n$ 为图 G 的顶点数.

(b) 以输入 $\langle G, s, t, n-1\rangle$, 运行 LPATH. 如果接受, 则接受; 如果拒绝, 则拒绝.

”

很好理解, 因为 G 中的简单路径长度不可能超过 $n-1$, 并且长度为 $n-1$ 的路径一定是哈密顿路径. 很简单可以证明这个归约是多项式时间内可计算的, 因此 LPATH 是 NP 完全的.

□