# 从DFA到后缀自动机

张云帆

北京大学　信息科学技术学院

May 13, 2018

- 自动机理论简介
  - DFA 的相关定义
  - DFA 的最小化
- 后缀自动机
  - SAM 的定义
  - SAM 的性质
- 后缀自动机的线性构造算法
- 后缀自动机的拓展
  - Trie 树上的广义后缀自动机
- 后缀自动机的简单应用

设 $S \in \Sigma^*$ 是字符集 $\Sigma$ 上一个长度为 $n$ 的串，$\epsilon$ 表示空串。

- $|S| = n$ 表示串$S$的长度，$\sigma = |\Sigma|$ 为字符集大小。
- $S[i..j]$：表示由串$S$的第 $i \sim j$ 个字符形成的字符串。
- $S\mathbf{c}$：表示在串$S$末尾添加字符**c**后形成的新串。
- $Sw$：表示在串$S$末尾添加另一个串$w$后形成的新串。
- 前缀：$\epsilon, S[1..1], S[1..2], \cdots, S[1..n]$ 称为串$S$的前缀。
- 后缀：$\epsilon, S[1..n], S[2..n], \cdots, S[n..n]$ 称为串$S$的后缀。
- “不同”与“本质不同”：两个子串 $S[i_1..j_1], S[i_2..j_2]$ 不同当且仅当 $i_1 \neq i_2 \vee j_1 \neq j_2$，而本质不同当且仅当它们对应的字符串不相等，即 $S[i_1..j_1] \neq S[i_2..j_2]$ 。

$$\Sigma = \{0, 1\},\ S = 100110,\ n = |S| = 6$$

$$\mathbf{c} = 1,\ w = 111 \Rightarrow S\mathbf{c} = 100110{\color{red}1},\ Sw = 100110{\color{red}111}$$

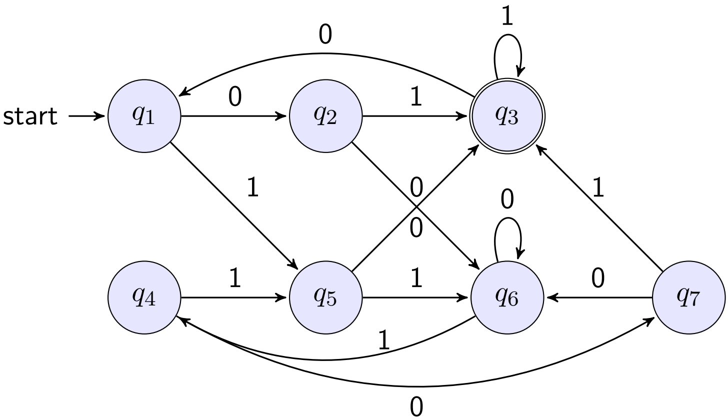
$$S[1..2] = 10 = S[5..6]$$

确定性有限状态自动机 (Deterministic Finite Automaton，简称DFA) 是一个五元组 $M = (\Sigma, Q, q_s, F, tr)$，其中

- $\Sigma$ 为一个有限字符集，其中每个字符$\mathbf{c}$称为一个输入符号；
- $Q$ 为一个有限状态集合；
- $q_s \in Q$ 称为初始状态；
- $F \subseteq Q$ 称为终结状态集合；
- $tr \in Q \times \Sigma \to Q$ 称为状态转换函数

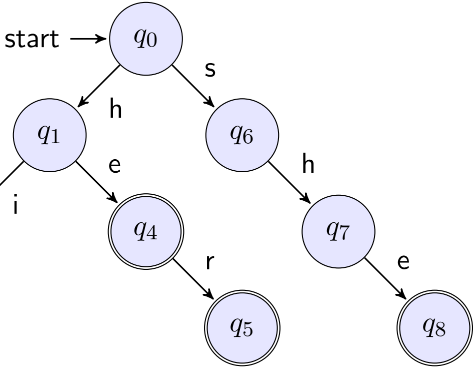
$$tr(q, \mathbf{c}) = q' \ (q, q' \in Q, \ \mathbf{c} \in \Sigma)$$

表示在状态 $q$ 输入符号 $\mathbf{c}$ 之后，自动机 $M$ 将转换至的下一个状态 $q'$ ，此时 $q'$ 称为 $q$ 的一个后继状态。




$$\Sigma = \{0, 1\},\ Q = \{q_1, q_2, \cdots, q_7\},\ q_s = q_1,\ F = \{q_3\}$$

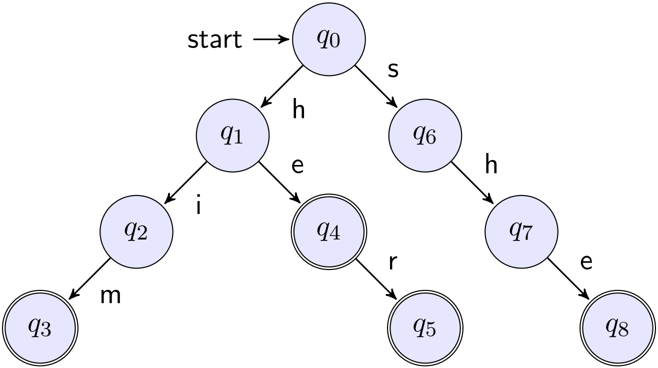
字典{him, her, he, she}构建出的Trie如下：




$$\Sigma = \{h, i, m, e, r, s\},\ q_s = q_0,\ F = \{q_3, q_4, q_5, q_8\}$$

在状态转换图中不存在的转换边，可以认为它们转换到了一个非法状态 $q_\phi$，且 $q_\phi$ 输入任何符号都只能转换到自己。

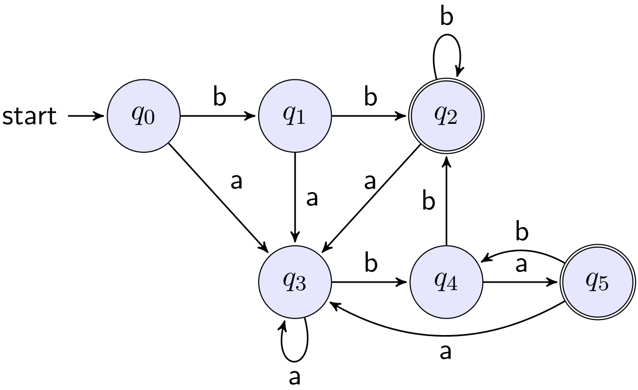
字典{him, her, he, she}构建出的Trie如下：




$$\Sigma = \{h, i, m, e, r, s\},\ \ q_s = q_0,\ \ F = \{q_3, q_4, q_5, q_8\}$$

$$Q = \{q_0, \cdots, q_8, q_\phi\}$$

字符串集{bb, aba}构建出的AC自动机如下：

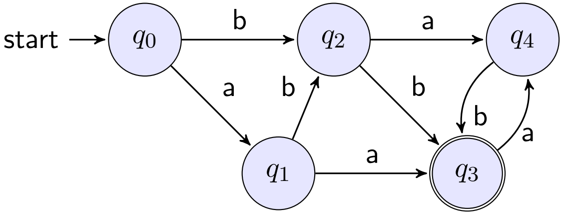



$$\Sigma = \{a, b\},\ Q = \{q_0, q_1, q_2, q_3, q_4, q_5\},\ q_s = q_0,\ F = \{q_2, q_5\}$$

对所有 $q \in Q, S \in \Sigma^*, \mathbf{c} \in \Sigma$，递归地扩展$tr$的定义如下：

$$tr(q, \epsilon) = q$$
$$tr(q, S\mathbf{c}) = tr(tr(q, S), \mathbf{c})$$

此时，称$M$接受/识别一个串$S$当且仅当 $tr(q_s, S) \in F$。



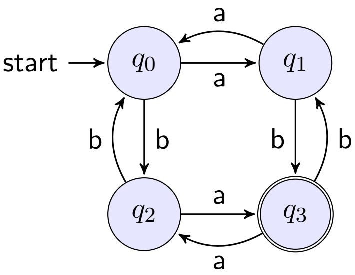

$$tr(q_s, \text{ababa}) = q_4,\ tr(q_2, \text{ab}) = q_3 \in F$$

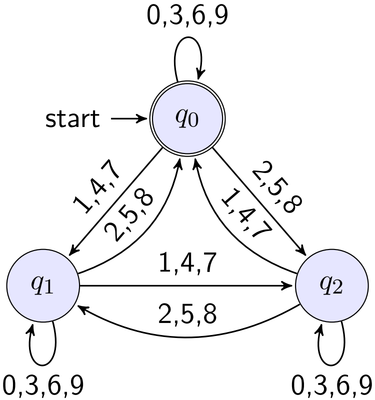
$$tr(q_1, \epsilon) = q_1,\ tr(q_4, \text{baa}) = q_\phi$$

定义集合 $L(M) = \{S \in \Sigma^* \mid tr(q_s, S) \in F\}$ 表示自动机 $M$ 能识别的所有串。

定义集合 $L(q) = \{S \in \Sigma^* \mid tr(q, S) \in F\}$ 表示从状态 $q \in Q$ 开始能识别的所有串。

练习：描述下面两个自动机能识别的所有串



对于任意 $p, q \in Q$，称 $p$ 与 $q$ 为等价状态当且仅当

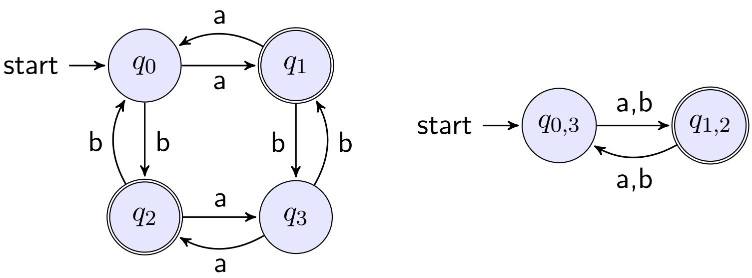
$$tr(p, S) \in F \iff tr(q, S) \in F \qquad \forall S \in \Sigma^*$$

即 $L(p) = L(q)$ 成立，此时记作 $p \sim q$ 。

若存在一个自动机 $M'$ ，使得 $L(M) = L(M')$ 且 $M'$ 的状态数 $|Q'|$ 最少，则称 $M'$ 是 $M$ 的最小状态自动机。

可以证明，最小状态自动机必定存在且在同构意义下唯一。

显然，最小状态自动机中任意两个状态都不等价。


start
$q_0$
$q_1$
$q_2$
$q_3$
a
a
a
a
b
b
b
b
start
$q_{0,3}$
$q_{1,2}$
a,b
a,b

**Theorem 2.1 (等价状态判定定理)**

设 $p, q \in Q$ 是两个不同的状态，则 $p \sim q$ 的充分必要条件是：

(1) 一致性条件：$p, q$ 必须同为终结状态或同为非终结状态，即

$$p \in F \iff q \in F$$

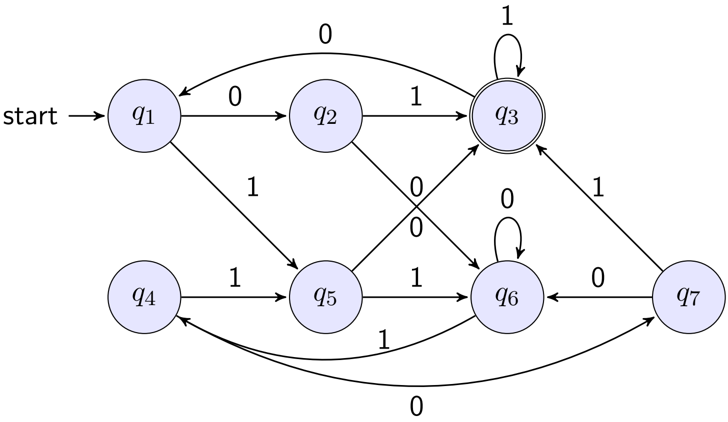
(2) 蔓延性条件：$p, q$ 输入同一个符号 $\mathbf{c}$ 后的新状态等价，即

$$tr(p, \mathbf{c}) \sim tr(q, \mathbf{c}) \qquad \forall \mathbf{c} \in \Sigma$$

根据上述条件，可以通过不断对等价类进行划分的方式来求出最小状态自动机。

**Algorithm 1** 等价类划分算法

初始划分 $\Pi \leftarrow \{F,\ Q\backslash F\}$
**repeat**
    $\Pi_{new} \leftarrow \Pi$
    **for all** $G \in \Pi \wedge |G| > 1$ **do**
        **for all** c $\in \Sigma$ **do**
            **if** $G$ 中状态对 c 的转换不全在 $\Pi$ 中的同一组 **then**
                根据转换后不同的组对 $G$ 进一步分类得到 $G^*_{\mathsf{c}}$
                $\Pi_{new} \leftarrow \Pi_{new}\backslash\{G\} \cup G^*_{\mathsf{c}}$
                **break**
            **end if**
**until** $\Pi = \Pi_{new}$


start
$q_1$
$q_2$
$q_3$
$q_4$
$q_5$
$q_6$
$q_7$
0
1
0
1
1
0
0
0
1
1
1
0
1
0

| | 0 | 1 |
|---|---|---|
| $q_1$ | $q_2$ | $q_5$ |
| $q_2$ | $q_6$ | $q_3$ |
| $q_3$ | $q_1$ | $q_3$ |
| $q_4$ | $q_7$ | $q_5$ |
| $q_5$ | $q_3$ | $q_6$ |
| $q_6$ | $q_6$ | $q_4$ |
| $q_7$ | $q_6$ | $q_3$ |

$[1,2,3,4,5,6,7]$

$[1,2,4,5,6,7]$ $[\ 3\ ]$

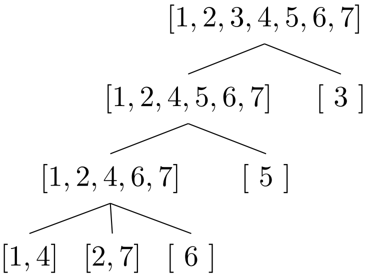
| | 0 | 1 |
|---|---|---|
| $q_1$ | $q_2$ | $q_5$ |
| $q_2$ | $q_6$ | $q_3$ |
| $q_3$ | $q_1$ | $q_3$ |
| $q_4$ | $q_7$ | $q_5$ |
| $q_5$ | $q_3$ | $q_6$ |
| $q_6$ | $q_6$ | $q_4$ |
| $q_7$ | $q_6$ | $q_3$ |

$[1, 2, 3, 4, 5, 6, 7]$

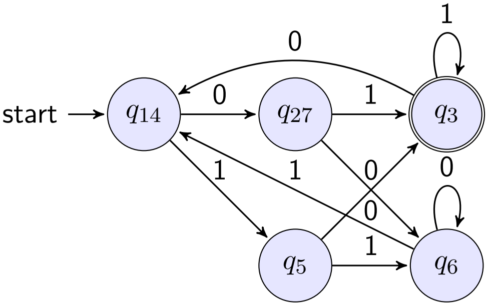
$[1, 2, 4, 5, 6, 7]$ $[\ 3\ ]$

$[1, 2, 4, 6, 7]$ $[\ 5\ ]$

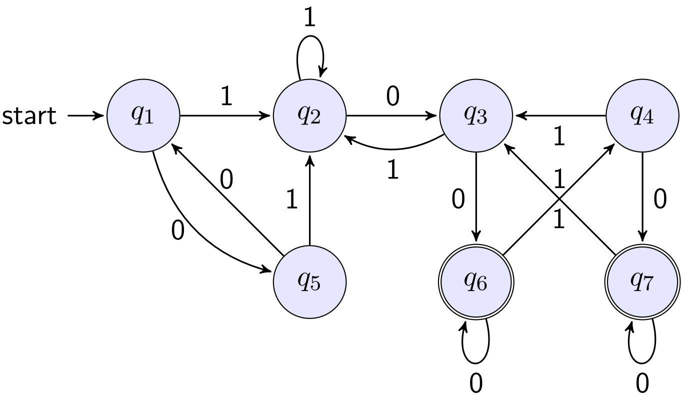
| | 0 | 1 |
|---|---|---|
| $q_1$ | $q_2$ | $q_5$ |
| $q_4$ | $q_7$ | $q_5$ |
| $q_2$ | $q_6$ | $q_3$ |
| $q_7$ | $q_6$ | $q_3$ |
| $q_3$ | $q_1$ | $q_3$ |
| $q_5$ | $q_3$ | $q_6$ |
| $q_6$ | $q_6$ | $q_4$ |



| | 0 | 1 |
|---|---|---|
| $\{q_1, q_4\}$ | $\{q_2, q_7\}$ | $\{q_5\}$ |
| $\{q_2, q_7\}$ | $\{q_6\}$ | $\{q_3\}$ |
| $\{q_3\}$ | $\{q_1, q_4\}$ | $\{q_3\}$ |
| $\{q_5\}$ | $\{q_3\}$ | $\{q_6\}$ |
| $\{q_6\}$ | $\{q_6\}$ | $\{q_1, q_4\}$ |

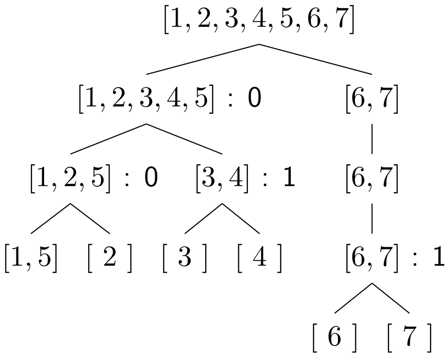


练习：将下面这个 DFA 最小化



| | 0 | 1 |
|---|---|---|
| $q_1$ | $q_5$ | $q_2$ |
| $q_2$ | $q_3$ | $q_2$ |
| $q_3$ | $q_6$ | $q_2$ |
| $q_4$ | $q_7$ | $q_3$ |
| $q_5$ | $q_1$ | $q_2$ |
| $q_6$ | $q_6$ | $q_4$ |
| $q_7$ | $q_7$ | $q_3$ |

| | 0 | 1 |
|---|---|---|
| $q_1$ | $q_5$ | $q_2$ |
| $q_2$ | $q_3$ | $q_2$ |
| $q_3$ | $q_6$ | $q_2$ |
| $q_4$ | $q_7$ | $q_3$ |
| $q_5$ | $q_1$ | $q_2$ |
| $q_6$ | $q_6$ | $q_4$ |
| $q_7$ | $q_7$ | $q_3$ |



**Definition 1 (后缀自动机)**

称一个串 $S$ 的后缀自动机 (Suffix Automaton，简称SAM) 为能够识别 $S$ 的所有后缀的最小状态自动机。

我们接下来分析这个后缀自动机应该具有哪些性质。

记 $n = |S|$ 为串长，$suf_i = S[i..n]$ 表示从 $i$ 开始的后缀。

特别地，定义 $suf_{n+1} = S[n+1..n] = \epsilon$ 表示空串。

**Lemma 3.1**

对于任意 $T \in \Sigma^*$，$tr(q_s, T) \neq q_\phi$ 当且仅当 $T$ 是 $S$ 的一个子串。

**Proof.**

若 $T = S[i..j]$ 为 $S$ 的子串，令 $w = suf_{j+1}$ 则 $Tw = suf_i$ 是串 $S$ 的一个后缀，因此 $L(tr(q_s, T)) \neq \phi \implies tr(q_s, T) \neq q_\phi$ 。

若 $T$ 不是 $S$ 的子串，则在 $T$ 后连接任何串都不可能成为 $S$ 的后缀，故 $L(tr(q_s, T)) = \phi = L(q_\phi)$ 。

再由 SAM 的最小性可知 $tr(q_s, T) = q_\phi$ 。 $\square$

**Lemma 3.1**

对于任意 $T \in \Sigma^*$，$tr(q_s, T) \neq q_\phi$ 当且仅当 $T$ 是 $S$ 的一个子串。

**Proof.**

若 $T = S[i..j]$ 为 $S$ 的子串，令 $w = suf_{j+1}$ 则 $Tw = suf_i$ 是串 $S$ 的一个后缀，因此 $L(tr(q_s, T)) \neq \phi \Longrightarrow tr(q_s, T) \neq q_\phi$ 。

若 $T$ 不是 $S$ 的子串，则在 $T$ 后连接任何串都不可能成为 $S$ 的后缀，故 $L(tr(q_s, T)) = \phi = L(q_\phi)$ 。

再由 SAM 的最小性可知 $tr(q_s, T) = q_\phi$ 。 □

因此，后缀自动机中只需保存$S$的所有子串对应的状态。在之后的讨论中，除非特别说明，否则不再考虑非法状态 $q_\phi$ 。

## Definition 2 (结束位置集合)

设 $T \in \Sigma^+$ 是一个非空字符串，定义 $T$ 在母串 $S$ 中所有的结束位置/右端点的集合如下：

$$right(T) := \{r \mid \exists 1 \le l \le r,\ T = S[l..r]\}$$

特别地，定义空串的结束位置集合为 $right(\epsilon) = \{0, 1, \cdots, n\}$ 。

$$S = \mathbf{aaababab},\ n = 8$$

$$right(\mathbf{ab}) = right(\mathbf{b}) = \{4, 6, 8\},\ right(\mathbf{aaa}) = \{3\}$$

$$right(\mathbf{abb}) = \phi,\ right(\epsilon) = \{0, 1, 2, \cdots, 8\}$$

**Lemma 3.2**

对于任意串 $T \in \Sigma^*$ 有 $L(tr(q_s, T)) = \{suf_{r+1} \mid r \in right(T)\}$ 。

**Proof.**

当 $T$ 不是 $S$ 的子串时，有 $right(T) = L(tr(q_s, T)) = \phi$ 。

否则任取串 $w \in L(tr(q_s, T))$ ，由 $Tw$ 是 $S$ 的一个后缀可知 $w$ 也是$S$的后缀。记 $w = suf_{r+1}$ ，显然 $r$ 必定是串 $T$ 的一个结束位置，故 $r \in right(T)$ 。

任取 $r \in right(T)$ ，则有 $T = s[l..r]$ 。令 $w = suf_{r+1}$ ，此时 $Tw = suf_l$ 是 $S$ 的一个后缀，故 $w \in L(tr(q_s, T))$ 。 □

## Theorem 3.3 (等价串判定定理)

设 $T_1, T_2 \in \Sigma^*$ 是任意两个串，则 $tr(q_s, T_1) \sim tr(q_s, T_2)$ 的充分必要条件是 $right(T_1) = right(T_2)$。此时称这两个串关于母串 $S$ 等价，记作 $T_1 \overset{S}{\sim} T_2$。

## Proof.

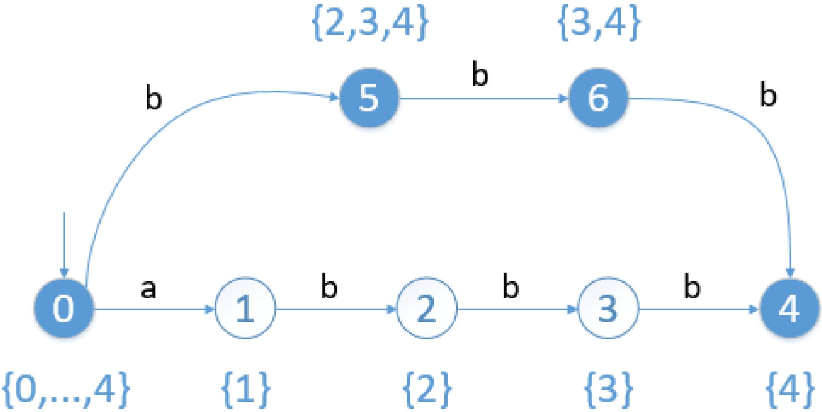
$$
\begin{aligned}
& tr(q_s, T_1) \sim tr(q_s, T_2) \\
\Longleftrightarrow\quad & L(tr(q_s, T_1)) = L(tr(q_s, T_2)) \\
\Longleftrightarrow\quad & \{suf_{r+1} \mid r \in right(T_1)\} = \{suf_{r+1} \mid r \in right(T_2)\} \\
\Longleftrightarrow\quad & right(T_1) = right(T_2)
\end{aligned}
$$

□

练习：从定义出发，画出字符串 $S = \mathbf{abbb}$ 的后缀自动机。

练习：从定义出发，画出字符串 $S = \mathbf{abbb}$ 的后缀自动机。

$$right(\mathbf{a}) = \{1\},\ \ right(\mathbf{ab}) = \{2\},\ \ right(\mathbf{abb}) = \{3\}$$
$$right(\mathbf{abbb}) = right(\mathbf{bbb}) = \{4\}$$
$$right(\mathbf{bb}) = \{3, 4\}$$
$$right(\mathbf{b}) = \{2, 3, 4\}$$
$$right(\epsilon) = \{0, 1, 2, 3, 4\}$$


{2,3,4}
{3,4}
b
b
b
5
6
0
1
2
3
4
a
b
b
b
{0,...,4}
{1}
{2}
{3}
{4}

因此，尽管 $S$ 的子串个数为 $O(n^2)$ 级别，但其中有很多子串是等价的，它们在后缀自动机中将会对应到同一个状态。

任取 $q \in Q$ ，那么对应到状态 $q$ 的所有子串，它们的 $right$ 集合必定相等，记这个集合为 $R_q$ 。

那么反过来，如何由 $R_q$ 得出状态 $q$ 对应着哪些子串呢？

任取 $r \in R_q$ ，则以 $r$ 为结束位置的共 $r+1$ 个串：

$$\epsilon, S[r..r], S[r-1..r], \cdots, S[1..r]$$

显然，$right$ 集合的大小关于它们的长度是单调减函数，因此所有 $right$ 集合恰好为 $R_q$ 的串，它们的长度对应着一个连续区间，称之为 $q$ 的合法长度区间，记作 $[minl_q, maxl_q]$。

可以发现，在状态转换图中，$minl_q, maxl_q$ 就分别对应着从初始状态 $q_s$ 到状态 $q$ 的最短路径和最长路径。

**Theorem 3.4 (三分律)**

任取两个不同状态 $p,q \in Q$ ，则下述三式有且仅有一成立

$$(1)\ R_p \cap R_q = \phi \qquad (2)\ R_p \subset R_q \qquad (3)\ R_q \subset R_p$$

**Proof.**

不妨设 $R_p \cap R_q \neq \phi$ 。任取结束位置 $r \in R_p \cap R_q$ ，由于状态 $p,q$ 对应的子串无交集，故 $[minl_p, maxl_p]$ 和 $[minl_q, maxl_q]$ 也不会有交集。

若 $maxl_p < minl_q$，则此时状态 $p$ 对应的所有子串长度都比状态 $q$ 短，而它们都以 $r$ 为结束位置，故 $p$ 对应的串都是 $q$ 的后缀，显然此时必定有 $R_q \subset R_p$ 。

而当 $maxl_q < minl_p$ 时，情况类似。 $\square$

因此，任意两个子串的 $right$ 集合，要么不相交，要么一个是另一个的子集。

对于任意 $p, q \in Q$ ，我们称状态 $p$ 是状态 $q$ 的 parent 状态，当且仅当 $R_p$ 是最小的真包含 $R_q$ 的集合。

显然 $R_{q_s} = \{0, 1, \cdots, n\}$ 真包含任何非空串的 $right$ 集合。

**Lemma 3.5**

任何状态 $q \in Q \backslash \{q_s\}$ 的 *parent* 状态存在且唯一。

**Proof.**

存在性显然，因为有 $R_q \subset R_{q_s}$ 。

假设有两个 parent 状态 $p_1, p_2$ ，即 $R_q \subset R_{p_1} \wedge R_q \subset R_{p_2}$ 。

由 $q \neq q_\phi$ 知 $R_q \neq \phi$，进而有 $R_{p_1} \cap R_{p_2} \neq \phi$，此时根据定理3.4，必有 $R_{p_1} \subset R_{p_2}$ 或 $R_{p_2} \subset R_{p_1}$ 成立。

这与 parent 状态 $right$ 集合的最小性矛盾。 □

因此，每个状态与 parent 状态之间的关系，可以构成一个树形结构，我们称这棵树为后缀自动机的 parent 树。

**Lemma 3.6**

若 $p$ 是 $q$ 的 *parent* 状态，则有 $maxl_p = minl_q - 1$ 。

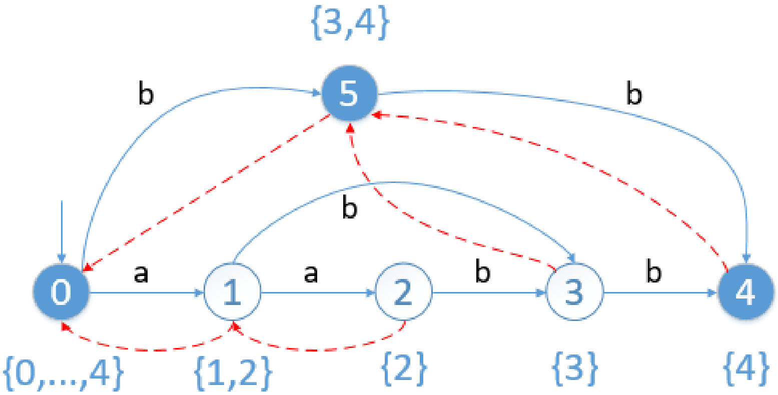
**Proof.**

由定理3.4的证明过程知，$maxl_p < minl_q$。

任取 $r \in R_q$，考虑以 $r$ 为结束位置长度且为 $minl_q - 1$ 的串 $T$ ，那么有 $R_q \subset right(T) \subseteq R_p$ 。而由 parent 状态 $right$ 集合的最小性知，$|T| \in [minl_p, maxl_p] \Rightarrow maxl_p \geq minl_q - 1$。 □

所以我们在后缀自动机中只需存储每个状态的 parent 状态，以及最大合法长度 $maxl$ 的值即可。

字符串 aabb 构建出的后缀自动机如下：



### Proposition 1

对串 $S$ 构建后缀自动机 $M$ ，则 $M$ 的状态数 $|Q| \le 2|S|+1$ 。

### Proof.

在 parent 树中，相同层数的节点其 $right$ 集合必定两两不交，否则由定理3.4知一个节点会是另一个的祖先。

设 $f(n)$ 表示根节点 $right$ 集合大小为 $n$ 的 parent 树，其节点数的最大值。考虑子节点 $right$ 集合的大小，可以得到:

$$f(n) \le f(x_1)+\cdots+f(x_k)+1 \quad (x_1+\cdots+x_k=n,\ k\ge 2)$$

由数学归纳法容易证明，$f(n) \le 2n-1$ 。

而 parent 树根节点 $right$ 集合大小为 $|R_{q_s}| = |S|+1$ ，因此有 $|Q| \le 2(|S|+1)-1 = 2|S|+1$ 。 □

## Proposition 2

对串 $S$ 构建后缀自动机 $M$ ，则合法状态转换边（即不转换到非法状态 $q_\phi$ 的边）的数目不超过 $3|S|$ 。

## Proof.

首先，以 $q_s$ 为根节点求出 $M$ 状态转换图中的一棵树形图，那么树形图中的边只有 $|Q|-1 \le 2|S|$ 条。

对于任意一条非树边 $p \to q$ ，构造路径 $q_s \to p \to q \to q_f$ ，要求 $q_s \to p$ 只经过树边，且 $q_f \in F$ 是某一个终结状态。

那么每一条这样的路径必定对应着 $S$ 的一个非空后缀，并且任意两条不同的路径上的第一条非树边都不相同。

因此，每个非空后缀最多对应一条非树边，而非空后缀总数不超过 $|S|$ ，故非树边数目也不超过 $|S|$ 。 $\square$

综上所述，可以得出 $S$ 的后缀自动机应该具有以下性质：

(1) 后缀自动机中只需存储 $S$ 的子串对应的状态；

(2) 一个状态对应的所有子串，它们的 $right$ 集合相等，且长度取值必定构成一个连续区间；

(3) 每个状态与 parent 状态的关系可以构成一棵 parent 树；

(4) 一个状态的最小合法长度恰好比其 parent 状态的最大合法长度多 1 ；

(5) 后缀自动机是一个线性结构。

综上所述，可以得出 $S$ 的后缀自动机应该具有以下性质：

(1) 后缀自动机中只需存储 $S$ 的子串对应的状态；

(2) 一个状态对应的所有子串，它们的 $right$ 集合相等，且长度取值必定构成一个连续区间；

(3) 每个状态与 parent 状态的关系可以构成一棵 parent 树；

(4) 一个状态的最小合法长度恰好比其 parent 状态的最大合法长度多 1 ；

(5) 后缀自动机是一个线性结构。

那么，接下来我们考虑如何线性构造这样一个后缀自动机。

考虑使用增量法，每个阶段在原串末尾添加一个字符，然后对已得到的原串 SAM 进行更新，使得它成为新串的 SAM 。

**Lemma 4.1**

对于任意两个串 $T_1, T_2$ ，若 $T_1 \overset{S}{\not\sim} T_2$ ，则必有 $T_1 \overset{S\mathbf{c}}{\not\sim} T_2$ 。

**Proof.**

由 $T_1 \overset{S}{\not\sim} T_2$ 可知 $right(T_1) \neq right(T_2)$ ，而在串 $S$ 后添加新字符 $\mathbf{c}$ 并不会影响 $n$ 之前的结束位置，因此在新串 $S\mathbf{c}$ 中 $T_1$ 和 $T_2$ 的 $right$ 集合也必定不相等。 □

考虑使用增量法，每个阶段在原串末尾添加一个字符，然后对已得到的原串 SAM 进行更新，使得它成为新串的 SAM 。

**Lemma 4.1**

对于任意两个串 $T_1, T_2$ ，若 $T_1 \overset{S}{\not\sim} T_2$ ，则必有 $T_1 \overset{S\mathbf{c}}{\not\sim} T_2$ 。

**Proof.**

由 $T_1 \overset{S}{\not\sim} T_2$ 可知 $right(T_1) \neq right(T_2)$ ，而在串 $S$ 后添加新字符 $\mathbf{c}$ 并不会影响 $n$ 之前的结束位置，因此在新串 $S\mathbf{c}$ 中 $T_1$ 和 $T_2$ 的 $right$ 集合也必定不相等。 □

因此，在末尾添加一个字符不会造成 SAM 中状态的合并，我们只需新增一些状态然后考虑状态转换边的变化。

在末尾添加一个字符 **c** 后，新增的子串必定都是串 $S$**c** 的后缀。而 $S$**c** 的后缀，就是 $S$ 的后缀在末尾连接一个字符 **c** 。

称 SAM 中所有终结状态为后缀状态，即那些 $right$ 集合包含 $n$ 的状态。那么只有这些后缀状态关于字符 **c** 的后继状态，它们的 $right$ 集合才可能发生变化。

因此，新增字符 **c** 后，只有后缀状态符号为 **c** 的转换边和对应的那些后继状态可能需要更新。

记 $p = tr(q_s, S)$，则有 $R_p = \{n\}$ ，那么在 parent 树上所有后缀状态必定都是 $p$ 的祖先。将这些状态按后代到祖先的顺序依次排列为：$p = v_1$，$v_2$，$v_3$，$\cdots$ ，$v_k = q_s$ 。

由于 $tr(q_s, S\mathbf{c}) = q_\phi$ ，因此需要新建状态 $np = tr'(q'_s, S\mathbf{c})$ 表示在新自动机中 $S\mathbf{c}$ 对应的状态，此时 $R'_{np} = \{n+1\}$ 。

**Lemma 4.2**

设 $q = tr(p, \mathbf{c})$，则有 $R_q = \{r+1 \mid r \in R_p \wedge S[r+1] = \mathbf{c}\}$ 。

**Proof.**

任取子串 $T$ 满足 $tr(q_s, T) = p$ ，那么 $tr(q_s, T\mathbf{c}) = q$ 。

设 $r \in right(T) \Longrightarrow T = S[l..r]$ ，此时若满足 $S[r+1] = \mathbf{c}$ ，则有 $T\mathbf{c} = S[l..r+1]$ ，因此 $r+1 \in right(T\mathbf{c})$ 。

设 $r+1 \in right(T\mathbf{c}) \Longrightarrow T\mathbf{c} = S[l..r+1]$ ，显然 $T = S[l..r]$，因此有 $r \in right(T)$ 且 $S[r+1] = \mathbf{c}$ 。

再由 $R_p = right(T)$，$R_q = right(T\mathbf{c})$ 即可得。 □

考虑某一个后缀状态 $v$ ，设 $R_v = \{r_1, r_2, \cdots, r_m = n\}$ 。

如果状态 $v$ 没有符号为 $\mathbf{c}$ 的合法转换边，那么 $R_v$ 中必定不存在除 $r_m$ 外满足 $S[r_i+1] = \mathbf{c}$ 的结束位置，因此它所有关于符号 $\mathbf{c}$ 的后继状态 $right$ 集合中只有 $n+1$ 。在这种情况下，我们只需新建一条从 $v$ 到 $np$ 符号为 $\mathbf{c}$的边。

由 parent 树的性质，$v_1, v_2, \cdots$ 的 $right$ 集合不断扩大，因此若节点 $v_j$ 存在符号为 $\mathbf{c}$ 的边，则 $v_{j+1}$ 也一定存在。

考虑某一个后缀状态 $v$ ，设 $R_v=\{r_1,r_2,\cdots,r_m=n\}$ 。

如果状态 $v$ 没有符号为 $\mathbf{c}$ 的合法转换边，那么 $R_v$ 中必定不存在除 $r_m$ 外满足 $S[r_i+1]=\mathbf{c}$ 的结束位置，因此它所有关于符号 $\mathbf{c}$ 的后继状态 $right$ 集合中只有 $n+1$ 。在这种情况下，我们只需新建一条从 $v$ 到 $np$ 符号为 $\mathbf{c}$的边。

由 parent 树的性质，$v_1,v_2,\cdots$ 的 $right$ 集合不断扩大，因此若节点 $v_j$ 存在符号为 $\mathbf{c}$ 的边，则 $v_{j+1}$ 也一定存在。

```
for (; p != NULL && p->tr[ch] == NULL; p = p->par)
    p->tr[ch] = np;
```

## Lemma 4.3

设 $T_1, T_2$ 是 $S$ 的两个子串，则 $T_1\mathbf{c} \overset{S}{\sim} T_2\mathbf{c} \wedge T_1\mathbf{c} \overset{S\mathbf{c}}{\nsim} T_2\mathbf{c}$ 的充要条件是 $n \in right(T_1) \oplus right(T_2)$ 。

## Proof.

添加字符 $\mathbf{c}$ 只会使一些子串的 $right$ 集合中新增一个 $n+1$，因此如果 $n \notin right(T_1) \oplus right(T_2)$，那么根据引理4.2可知 $n+1$ 也必定同时存在/不存在于 $T_1\mathbf{c}$ 和 $T_2\mathbf{c}$ 的新 $right$ 集合中。这与子串 $T_1\mathbf{c}$ 和 $T_2\mathbf{c}$ 不等价矛盾。

必要性类似讨论即可。 □

例如在串 $S = \mathbf{aabca}$ 后加入字符 **b** ，原来等价的串 $\mathbf{b}, \mathbf{ab}$ 和 **aab** 在新串中会被分裂为两个等价类 $\{\mathbf{b}, \mathbf{ab}\}$ 与 $\{\mathbf{aab}\}$ 。

因此添加字符 **c** 后，SAM 中的一些状态可能会发生分裂。这些状态满足既有后缀状态，也有非后缀状态能够通过符号 **c** 转换到它。

设 $v_p$ 是 $v_1, v_2, \cdots$ 中第一个存在合法转换边 $\mathbf{c}$ 的状态，记状态 $q = tr(v_p, \mathbf{c})$ 并任取 $r \in R_q$，将所有以 $r$ 为结束位置的非空子串按长度分别记为 $Q_1, \cdots, Q_r = S[1..r]$ 。

显然 $S[r] = \mathbf{c}$ ，因此不妨设 $Q_1 = P_0\mathbf{c}, \cdots, Q_r = P_{r-1}\mathbf{c}$ 。这 $r$ 个子串根据合法长度区间被划分为若干个等价类，其中 $q$ 对应的就是 $Q_{minl_q}, \cdots, Q_{maxl_q}$。

设 $v_p$ 是 $v_1, v_2, \cdots$ 中第一个存在合法转换边 $\mathbf{c}$ 的状态，记状态 $q = tr(v_p, \mathbf{c})$ 并任取 $r \in R_q$，将所有以 $r$ 为结束位置的非空子串按长度分别记为 $Q_1, \cdots, Q_r = S[1..r]$ 。

显然 $S[r] = \mathbf{c}$ ，因此不妨设 $Q_1 = P_0\mathbf{c}, \cdots, Q_r = P_{r-1}\mathbf{c}$ 。这 $r$ 个子串根据合法长度区间被划分为若干个等价类，其中 $q$ 对应的就是 $Q_{minl_q}, \cdots, Q_{maxl_q}$。

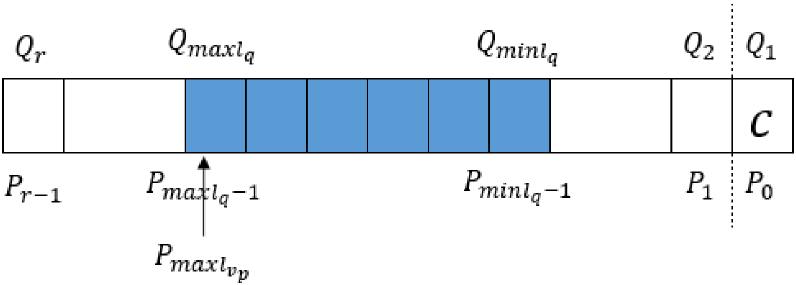
那么所有能通过符号 $\mathbf{c}$ 转换到 $q$ 的状态 $p$ ，对应的串只能在 $P_{minl_q-1}, \cdots, P_{maxl_q-1}$ 中，因此我们只需分析这些串是否是后缀串。

当 $P_j$ 是 $S$ 的后缀串时，$P_{j-1}$ 也必定是后缀串，而在状态 $p$ 对应的所有串 $P_k$ 中，最长的后缀串就是 $P_{maxl_{v_p}}$ 。

显然有 $maxl_q \geq maxl_{v_p} + 1 \geq minl_q$ ，分情况讨论：

(1) 若 $maxl_{v_p} = maxl_q - 1$ ，此时所有 $P_k$ 都是后缀串，因此通过 $\mathbf{c}$ 转换到 $q$ 的所有状态都是后缀状态，因此 $q$ 不会分裂。

而对于其余的后缀状态 $v$ ，它们是 $v_p$ 在 parent 树上的祖先状态，那么状态 $q_v = tr(v, \mathbf{c})$ 更不可能分裂。因为根据 $R_v \subset R_{v_p}$ 和引理4.2，$q_v$ 必定也以 $r$ 为结束位置且合法长度比 $q$ 还短，通过 $\mathbf{c}$ 转换到 $q_v$ 的必定只有后缀状态。

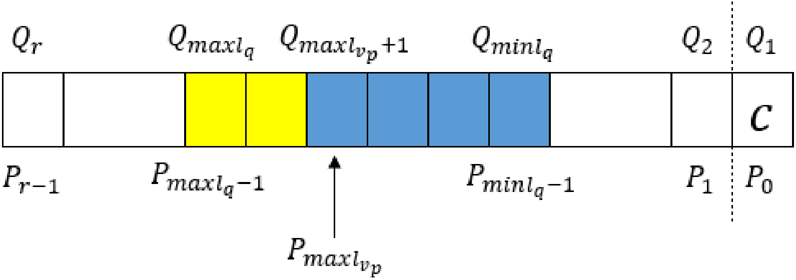

$Q_r$
$Q_{maxl_q}$
$Q_{minl_q}$
$Q_2$
$Q_1$
$c$
$P_{r-1}$
$P_{maxl_q-1}$
$P_{minl_q-1}$
$P_1$
$P_0$
$P_{maxl_{vp}}$

(2) 若 $maxl_{v_p} < maxl_q - 1$ ，则 $q$ 的合法长度区间需要分裂为两个 $[minl_q, maxl_{v_p}+1]$ 与 $[maxl_{v_p}+2, maxl_q]$ 。前者对应只由后缀状态通过 **c** 转换到的状态，后者对应只由非后缀状态通过 **c** 转换到的状态。

于是我们新建状态 $nq$ 来对应后缀状态通过 **c** 的转换，将其与状态 $q$ 区别开来，此时 $maxl_{nq} = maxl_{v_p}+1$ 。

跟据引理4.2可知 $R_{nq}$ 中新增的结束位置 $n+1$ 对状态转换没有任何作用，因此 $nq$ 的所有状态转换边跟 $q$ 完全相同。

对于其余的后缀状态 $v$ ，当 $tr(v, \mathbf{c}) = q$ 时我们需要更新为 $tr'(v, \mathbf{c}) = nq$ ，否则 $tr(v, \mathbf{c}) = q_v \neq q$ ，类似情况1可知 $q_v$ 只会由后缀状态转换过来，不会再发生分裂。


$Q_r$
$Q_{maxl_q}$
$Q_{maxl_{vp}+1}$
$Q_{minl_q}$
$Q_2$
$Q_1$
$C$
$P_{r-1}$
$P_{maxl_q-1}$
$P_{minl_q-1}$
$P_1$
$P_0$
$P_{maxl_{vp}}$

```
SamNode *q = p->tr[ch];
if (q->ml > p->ml + 1) {
    SamNode *nq = newNode(p->ml + 1);
    memcpy(nq->tr, q->tr, sizeof(q->tr));
    for (; p != NULL && p->tr[ch] == q; p = p->par)
        p->tr[ch] = nq;
}
```

接下来考虑所有新建/分裂的状态其 parent 状态如何变化。

首先考虑新建的代表 $S\mathbf{c}$ 的状态 $np$ ，若所有的 $v_1, v_2, \cdots$ 都不存在符号为 $\mathbf{c}$ 的合法转换边，则说明 $\mathbf{c}$ 在串中第一次出现，此时 $np$ 的 parent 状态为 $q_s$ 。

接下来考虑所有新建/分裂的状态其 parent 状态如何变化。

首先考虑新建的代表 $S\mathbf{c}$ 的状态 $np$ ，若所有的 $v_1, v_2, \cdots$ 都不存在符号为 $\mathbf{c}$ 的合法转换边，则说明 $\mathbf{c}$ 在串中第一次出现，此时 $np$ 的 parent 状态为 $q_s$ 。

否则由于 $R_{np} = \{n+1\}$ ，故其 parent 状态 $par_{np}$ 必定是由一个 $right$ 集合包含 $n$ 的后缀状态通过 $\mathbf{c}$ 转换得到，而且这个后缀的 $right$ 集合要尽可能小。

可以发现 $par_{np} = tr'(v_p, \mathbf{c}) = q/nq$ 。

在情况二中，我们将状态 $q$ 分裂成两个新状态 $q'$ 和 $nq$ 。设状态 $q$ 原来的parent状态为 $par_q$ ，则 $r \in R_{par_q} \cap R_{q'} \cap R_{nq}$ 。

根据定理3.4可知，这三个状态在parent树上构成祖孙关系，再由它们的合法长度区间之间的关系就可以可以推出：

$$par_{q'} = nq,\ par_{nq} = par_q$$

在情况二中，我们将状态 $q$ 分裂成两个新状态 $q'$ 和 $nq$ 。设状态 $q$ 原来的parent状态为 $par_q$ ，则 $r \in R_{par_q} \cap R_{q'} \cap R_{nq}$ 。

根据定理3.4可知，这三个状态在parent树上构成祖孙关系，再由它们的合法长度区间之间的关系就可以可以推出：

$$par_{q'} = nq,\ par_{nq} = par_q$$

至此，SAM的构造算法圆满结束~~

```
void samAppend(int ch) {
    SamNode *p = qlast;
    SamNode *np = newNode(p->ml + 1);
    qlast = np;
    for (; p != NULL && p->tr[ch] == NULL; p = p->par)
        p->tr[ch] = np;
    if (p == NULL)
        return np->par = qs, void();
    SamNode *q = p->tr[ch], *nq;
    if (q->ml > p->ml + 1) {
        nq = newNode(p->ml + 1);
        memcpy(nq->tr, q->tr, sizeof(q->tr));
        nq->par = q->par, np->par = q->par = nq;
        for (; p != NULL && p->tr[ch] == q; p = p->par)
            p->tr[ch] = nq;
    }
    else np->par = q;
}
```

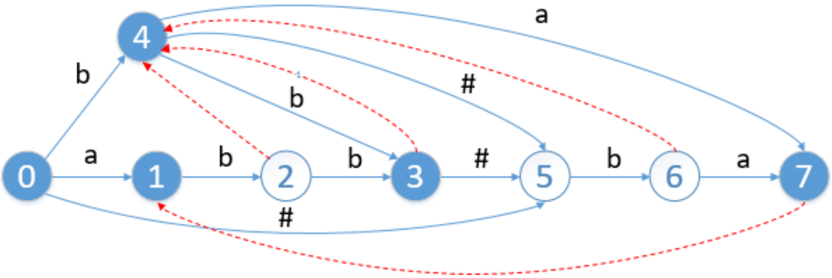
在前面的讨论中，我们只是针对单字符串的后缀自动机进行了分析，但实际上我们可能需要面对一些涉及两个或多个字符串的问题。因此，我们需要对后缀自动机进行相应的拓展。

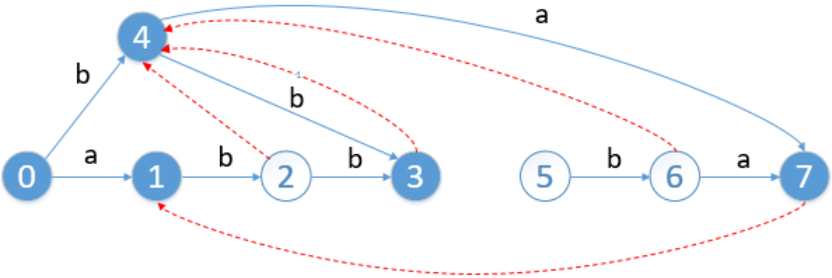
设 $\mathcal{S} = \{S_1, S_2, \cdots, S_m\} \subseteq \Sigma^+$ 为一个字符串集合，考虑构造一个有限状态自动机，使得它能接受 $\mathcal{S}$ 中每个串的后缀。

最简单的想法就是以一个特殊字符 # 为分隔，先将 $\mathcal{S}$ 中的所有串顺次连接起来，即 $S^* = \overline{S_1\#S_2\#\cdots\#S_m}$ 。

对串 $S^*$ 构造出其后缀自动机后，再将所有符号为 # 的状态转换边全部删去。

以 $\mathcal{S} = \{\mathbf{abb}, \mathbf{ba}\}$ 为例，构造自动机如下：


0
1
2
3
4
5
6
7
a
b
b
b
#
b
a
b
#
#
#
a


4
b
a
b
1
0
a
1
b
2
b
3
5
b
6
a
7

这样构造出的自动机，其状态数和合法状态转换边的数目都是 $O\left(\sum_i |S_i|\right)$ 级别的。而且从上一个例子还可以发现，自动机中可能存在大量未合并的等价状态，甚至还有不可达的死状态。

考虑用一棵字典树 $\mathcal{T}$ 来描述这个字符串集合，其中所有对应着单词的节点的集合为 $L(\mathcal{T}) \subseteq \mathcal{T}$。

记节点 $v_x \in \mathcal{T}$ 到其子树中某个节点 $v_y$ 的路径上，所有字符顺次连接所得到的串为 $T_{x,y}$。

**Definition 3 (结束位置集合)**

定义串 $w \in \Sigma^*$ 在字典树 $\mathcal{T}$ 中的结束位置集合为：

$$right(w) := \{v_r \in \mathcal{T} \mid \exists v_l \in \mathcal{T},\ T_{l,r} = w\}$$

类似地，我们以 $right$ 集合来对所有串进行等价类划分，即定义 $\Sigma^*$ 上的右不变等价关系 $\sim_R$ 如下：

$$w_1 \sim_R w_2 \iff right(w_1) = right(w_2)$$

记所有与 $w$ 等价的串所构成的集合为 $[w] := \{w' \mid w \sim_R w'\}$。

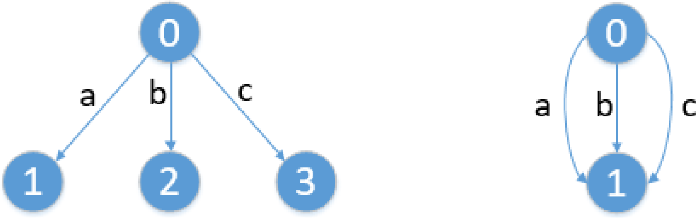
此时，我们可以形式化地定义字典树 $\mathcal{T}$ 上的广义后缀自动机 $M = (\Sigma, Q, q_s, F, tr)$ 如下：

- 状态集合： $Q = \{\ [w] \mid w \in \Sigma^*\ \}$
- 起始状态： $q_s = [\epsilon]$
- 终结状态： $F = \{\ [w] \mid right(w) \cap L(\mathcal{T}) \neq \emptyset\ \}$
- 状态转换函数： $tr([w], \mathbf{c}) = [w\mathbf{c}]$

对于单字符串的情形，两个串 $w_1, w_2$ 对应的状态等价当且仅当 $w_1 \sim_R w_2$，此时按上述定义得到的必定是最小自动机。

那么对于字典树的情形，是否仍能得到最小自动机呢？

由于 $w_1 \sim_R w_2$ 只是 $tr(q_s, w_1) \sim tr(q_s, w_2)$ 的充分不必要条件，广义后缀自动机中仍可能存在未合并的等价状态。




因此，广义后缀自动机的状态数/合法转换边数可能不再具有单字符串 SAM 的线性规模。

事实上通过分析可以得出，广义后缀自动机的状态数关于字典树大小 $|\mathcal{T}|$ 仍为线性，但其合法转换边数可能达到 $O(|\mathcal{T}| \cdot |\Sigma|)$ 的级别，跟字符集大小相关。

至于构造算法，类似单字符串 SAM 的构造进行讨论即可[1]。

[1]刘研绎《后缀自动机在字典树上的拓展》

在前面介绍自动机理论时，我们知道字典树其实就是一个特殊的 DFA 。那么能否在 DFA 上直接建立后缀自动机，并通过事先对 DFA 进行最小化来减少后缀自动机的规模呢？

为了使字符串集合 $S$ 是有限集，要求这个 DFA 只能接受有限个串，即它是一个无环自动机。

设字符串集合 $\mathcal{S}$ 以一个 DFA 的形式给出，即对于给定的无环 min DFA $A = (\Sigma, Q_A, q_{s_A}, F_A, tr_A)$ 有 $\mathcal{S} = L(A)$ 成立。

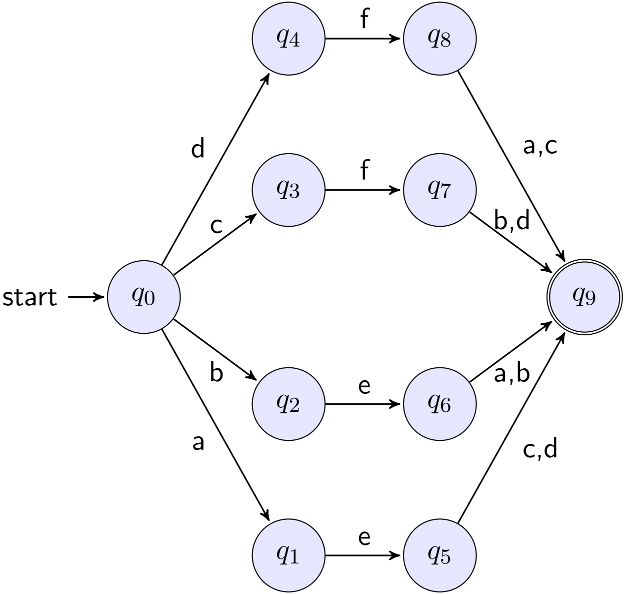
**Definition 4 (结束位置集合)**

定义串 $w \in \Sigma^*$ 在 DFA $A$ 中的结束位置集合为:

$$right(w) := \{q_r \in Q_A \mid \exists q_l \in Q_A,\ tr_A(q_l, w) = q_r\}$$

类似字典树上的讨论，我们定义一个右不变等价关系 $\sim_R$ ，然后形式化地给出 DFA $A$ 的广义后缀自动机 $M$ 。

那么在 $A$ 已经最小化的情况下，这样构造出的广义后缀自动机 $M$ 是否是最小状态自动机呢？




$$right(\text{e}) = \{q_5, q_6\}$$

$$right(\text{f}) = \{q_7, q_8\}$$

$$\therefore \text{e} \not\sim_R \text{f}$$

$$q_e = tr(q_s, \text{e})$$

$$q_f = tr(q_s, \text{f})$$

$$L(q_e) = \{\text{a,b,c,d}\}$$

$$L(q_f) = \{\text{a,b,c,d}\}$$

$$\therefore q_e \sim q_f$$

**Definition 5 (suffix-unique)**

称一个无环自动机 $A$ 是 suffix-unique 的，当且仅当 $L(A)$ 中的任意两个串都没有相等的非空后缀。

**Proposition 3**

*若给定的无环 min DFA $A$ 满足 suffix-unique 性质，则定义出的广义后缀自动机 $M$ 是最小状态自动机，且它具有线性的状态数和合法转换边数（与字符集大小无关），同时存在线性构造算法[a]。*

[a] Mohri, Mehryar, P. Moreno, and E. Weinstein. *General Suffix Automaton Construction Algorithm and Space Bounds*

PKU1509 Glass Beads

题目大意：求一个串的所有循环同构串中字典序最小的串。

PKU1509 Glass Beads

题目大意：求一个串的所有循环同构串中字典序最小的串。

$S$ 的循环同构串必定是 $SS$ 的子串，因此对 $SS$ 建立 SAM 后，找到一条从初始状态开始，字典序最小且长度为 $|S|$ 的路径即可。

SPOJ SUBLEX

题目大意：求一个串的所有本质不同的子串中，字典序第 $k$ 小的子串。

SPOJ SUBLEX

题目大意：求一个串的所有本质不同的子串中，字典序第 $k$ 小的子串。

对串 $S$ 建立SAM后，预处理从每个节点出发可以得到多少本质不同的子串即可。

注意对 SAM 进行拓扑排序的小技巧（可根据 $maxl$ 进行基数排序）。

## SPOJ LCS

题目大意：求两个串的最长连续公共子串。

首先，对其中任意一个串 $P$ 构建 SAM，然后把另一个串 $S$ 在 SAM 中进行匹配。由于 $x$ 是 $S$ 的子串，当且仅当 $x$ 是 $S$ 的某个前缀串的后缀。因此我们只要对于每个 $S$ 的前缀串，求出它的一个最长后缀满足这个后缀必须要在 $P$ 中出现即可。

记 $q_i$ 表示 $S[1..i]$ 满足要求的最长后缀在 SAM 中对应的状态，$l_i$ 表示这个最长后缀的长度。设 $\mathbf{c} = S[i+1]$ ，若 $tr(q_i, \mathbf{c}) \neq q_\phi$ ，则显然 $q_{i+1} = tr(q_i, \mathbf{c})$ 并且 $l_{i+1} = l_i + 1$ 。

否则找到离 $q_i$ 最近的一个祖先状态 $p$ 使得 $tr(p, \mathbf{c}) \neq q_\phi$，则 $q_{i+1} = tr(p, \mathbf{c})$ 。由于 $maxl_p < l_i$ ，故 $l_{i+1} = maxl_p + 1$ 。如果不存在这样的状态 $p$ ，那么说明 $\mathbf{c}$ 在 $P$ 中不存在，此时显然有 $q_{i+1} = q_s$，$l_{i+1} = 0$ 。

最后，对所有的$l_i$取最大值即可。

## HDU4641 K-string

题目大意：给定一个字符串 $S$ ，要求实现两种操作：

1. 在 $S$ 末尾添加一个字符 $\mathbf{c}$ ；

2. 询问当前 $S$ 所有本质不同的子串中，出现次数不小于 $k$ 的有多少个。

一个串 $P$ 在 $S$ 中的出现次数，等于 $S$ 所有前缀串中满足 $P$ 是其后缀的个数。从 parent 树的角度来说，就是 $P$ 对应节点的子树中有多少个前缀串。

再考虑每个状态 $q$ 对应多少个本质不同的子串，由合法长度区间知 $q$ 对应 $maxl_q - minl_q + 1 = maxl_q - maxl_{par_q}$ 个子串。

考虑离线处理，求出每个前缀对应的状态，并记录它们的出现时间。

此时，对于 SAM 的每个状态 $q$ ，它能够给答案提供贡献的时间，就是 $q$ 的子树中出现时间第 $k$ 小的前缀对应的出现时间。于是，我们使用主席树在DFS序上进行区间 $k$ 小值查询，就能得出每个状态的贡献时间。

而一个询问的答案，就是所有贡献时间在该操作出现之前的状态，它们对应的子串个数之和。因此，只需做一遍前缀和预处理即可。

1. 自动机理论
   - 定义、最小化算法
2. 后缀自动机
   - 定义、结构、性质
3. 后缀自动机的构造
   - 增量法、等价类
4. 广义后缀自动机
   - 字符串集、字典树、无环自动机
5. 后缀自动机的简单应用
   - 子串、匹配

## 参考文献


- 陈立杰《后缀自动机》
- 刘研绎《后缀自动机在字典树上的拓展》
- AlfredV.Aho, MonicaS.Lam, Ravi Sethi, 赵建华, & 郑滔等. (2009). 编译原理.
- Mohri, M., Moreno, P., & Weinstein, E. (2009). *General suffix automaton construction algorithm and space bounds*. Elsevier Science Publishers Ltd.

- Thanks for listening!
- Q & A