

后缀自动机 (SAM)

fjy666

June 16th, 2022

引入

首先，SAM 是什么？

Suffix AutoMaton, 后缀自动机。

这是 OI 中字符串算法的最高点了。

虽然如此，我们要清楚一个概念：

SAM 和 SA(后缀数组) 没有任何关系。

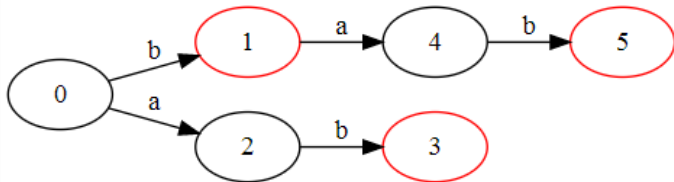
那么，就开始吧！

介绍

SAM 是一种什么结构？

我们先不管它，先来看一个东西：

字符串 $S = \text{"bab"}$ 和它的「后缀 Trie」（即把所有后缀扔到一个 Trie 上）



介绍

这玩意有个非常棒的性质：它包括了 S 的所有子串的信息。
从节点 0 开始，随便走一段必定是 S 的子串，
而 S 的子串也必定是 0 到某一个节点的路径。
并且，这个「后缀 Trie」是一个 DAG，可以很方便的 dp。

介绍

这玩意有个非常棒的性质：它包括了 S 的所有子串的信息。
从节点 0 开始，随便走一段必定是 S 的子串，
而 S 的子串也必定是 0 到某一个节点的路径。
并且，这个「后缀 Trie」是一个 DAG，可以很方便的 dp。
唯一也是致命的缺点：这玩意的时空复杂度是 $\mathcal{O}(n^2)$ 的！
看到这里，你应该清楚 SAM 是个什么东西了吧！

没错，SAM 就是一个具有上述性质，并且时空复杂度均为 $\mathcal{O}(n \log \Sigma)$ 的结构！

定义

虽说如此，SAM 的概念还是有必要提一句的。

字符串 s 的 SAM 是一个接受 s 的所有后缀的最小 DFA (确定性有限自动机或确定性有限状态自动机)。

换句话说：

- SAM 是一张有向无环图。结点被称作 **状态**，边被称作状态间的 **转移**。
- 图存在一个源点 t_0 ，称作 **初始状态**，其它各结点均可从 t_0 出发到达。
- 每个 **转移** 都标有一些字母。从一个结点出发的所有转移均 **不同**。
- 存在一个或多个 **终止状态**。如果从初始状态 t_0 出发，最终转移到了一个终止状态，则路径上的所有转移连接起来一定是字符串 s 的一个后缀。 s 的每个后缀均可用一条从 t_0 到某个终止状态的路径构成。
- 在所有满足上述条件的自动机中，SAM 的结点数是最少的。

From oi-wiki.org

endpos

endpos 是什么？

考虑原串 S 的任意非空子串 T ，那么

$\text{endpos}(T)$ 被定义为 T 在 S 中出现时末尾位置所组成的集合（下标从 1 开始）。

这个可能有点难懂，所以我举个例子：

$S = "114514", T = "14"$ ，

那么 $\text{endpos}(T) = \{3, 6\}$ 。

对于空串，我们定义它的 endpos 为 $\{0, 1, 2, 3, \dots, |S|\}$

是不是非常 Easy？这玩意必须记住，这是重中之重。

endpos

我们定义 endpos 等价类为一堆 endpos 相等的子串所组成的集合。

显然，两个不同的 endpos 等价类不可能有相同的元素。

那么这样我们就把一共 $\mathcal{O}(n^2)$ 种子串分成了 $\mathcal{O}(n)$ 种 endpos 等价类。

有人要问了：为啥是 $\mathcal{O}(n)$ ？自己翻 OI-wiki 去/xyx

link

link，即后缀链接，是「SAM 上的 fail 指针」。
这玩意很玄学，我们来看看 OI-wiki 的定义吧！

考虑 SAM 中某个不是 t_0 的状态 v 。我们已经知道，状态 v 对应于具有相同 endpos 的等价类。
我们如果定义 w 为这些字符串中最长的一个，则所有其它的字符串都是 w 的后缀。

我们还知道字符串 w 的前几个后缀（按长度降序考虑）全部包含于这个等价类，且所有其它后缀（至少有一个——空后缀）在其它的等价类中。我们记 t 为最长的这样的后缀，然后将 v 的后缀链接连到 t 上。

换句话说，一个后缀链接 $\text{link}(v)$ 连接到对应于 w 的最长后缀的另一个 endpos 等价类的状态。

link

有人会问了：fjy 你这样没良心的抄 OI-wiki 好吗？

额……

我们很容易地发现：如果定义一个结点 x 的父节点为 $\text{link}[x]$ ，那么这就是！一课！树！

link

有人会问了：fjy 你这样没良心的抄 OI-wiki 好吗？

额……

我们很容易地发现：如果定义一个结点 x 的父节点为 $\text{link}[x]$ ，那么这就是！一课！树！恭迎！凸包之神！俞开！！ 1111

link

有人会问了：fjy 你这样没良心的抄 OI-wiki 好吗？

额……

我们很容易地发现：如果定义一个结点 x 的父节点为 $\text{link}[x]$ ，那么这就是！一课！树！恭迎！凸包之神！俞开!! 1111



而根据树的定义，我们似乎也可以把 SAM 叫做凸包/ xyx 。

node

自动机吗，肯定是有一个个节点组成的。

那么 SAM 的节点是什么呢？

由于有 $\mathcal{O}(n)$ 种 endpos 等价类（下称等价类），

每个 SAM 节点都代表一个等价类内所有的子串的集合！

当然实现的时候不可能真存一堆字符串，也不会存下 endpos，
否则空间炸出翔。

node

那一个 node 里存啥捏？别急，我们先来引入一些记号：

s 的子串可以根据它们结束的位置 endpos 被划分为多个等价类；

SAM 由初始状态 t_0 和与每一个 endpos 等价类对应的每个状态组成；

对于每一个状态 v ，一个或多个子串与之匹配。我们记 $\text{longest}(v)$ 为其中最长的一个字符串，记 $\text{len}(v)$ 为它的长度。类似地，记 $\text{shortest}(v)$ 为最短的子串，它的长度为 $\text{minlen}(v)$ 。那么对应这个状态的所有字符串都是字符串 $\text{longest}(v)$ 的不同的后缀，且所有字符串的长度恰好覆盖区间 $[\text{minlen}(v), \text{len}(v)]$ 中的每一个整数。

对于任意不是 t_0 的状态 v ，定义后缀链接为连接到对应字符串 $\text{longest}(v)$ 的长度为 $\text{minlen}(v) - 1$ 的后缀的一条边。从根节点 t_0 出发的后缀链接可以形成一棵树。这棵树也表示 endpos 集合间的包含关系。

对于 t_0 以外的状态 v ，可用后缀链接 $\text{link}(v)$ 表达 $\text{minlen}(v)$ ：

$$\text{minlen}(v) = \text{len}(\text{link}(v)) + 1.$$

如果我们从任意状态 v_0 开始顺着后缀链接遍历，总会到达初始状态 t_0 。这种情况下我们可以得到一个互不相交的区间 $[\text{minlen}(v_i), \text{len}(v_i)]$ 的序列，且它们的并集形成了连续的区间 $[0, \text{len}(v_0)]$ 。

总结

SAM 确实是一种比较强大的 string DS。
它可以很方便地解决很多和后缀有关的东西。
有些本质不同子串问题也可以用它。
总而言之，遇到不会的题，SAM 淦它就对了！

Goodbye

Thank you for your listening!

Made by fjy666.

参考链接:

<https://oi-wiki.org/string/sam/>

<https://www.luogu.com.cn/problem/solution/P3804>

<https://alpha1022.gitee.io/sam-visualizer/>

https://blog.csdn.net/qq_42101694/article/details/111740597

Special Thanks

Special Thanks to lym(fix \LaTeX error in my computer), the oi-wiki and luogu.