后缀自动机 (SAM)

fjy666

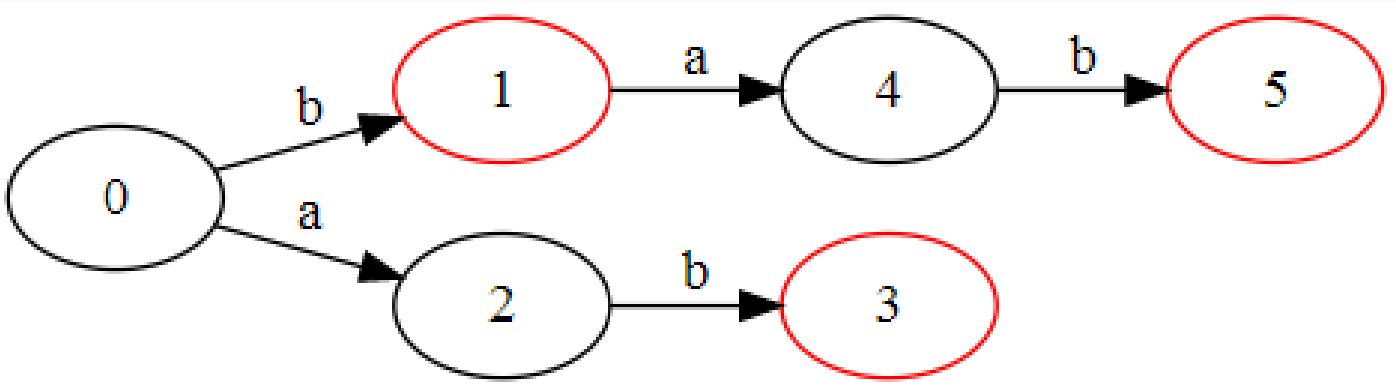
June 16th, 2022

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="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**。如果我们从初始状态 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
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 endpos(T)={3,6}。
对于空串，我们定义它的 endpos 为 $\{0,1,2,3,\cdots,|S|\}$
是不是非常 Easy？这玩意必须记住，这是重中之重。

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？

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.