# 后缀自动机SAM:

【性质】

* 每个状态包含多个字符串，每个字符串仅属于一个状态。
* 同一状态内p的字符串endpos(p)集合相等。
* 同一状态p内的字符串为几段长度互不相同且连续的子串，长度介于[maxlen(p),maxlen(link(p))+1]中，且这一状态内的所有串都是长度为maxlen(p)的串的后缀。
  + 推论:一个状态p所表示的本质不同的字符串有恰好maxlen(p)-maxlen(link(p))个。
* 同一状态p内每个字符串在原串中出现次数相同，都为mx(p)
  + 推论:Σmx(p)(maxlen(p)-maxlen(link(p))) = n(n+1)/2
* 原串的所有子串都可以通过从根开始的连续转移边c(p,x)到达SAM的某个节点；原串的所有后缀都可以通过从根开始的连续转移边c(p,x)到达SAM的某个accept(p)=true的节点。
* link(p)形成一棵内向树结构
* link树上任意一个点到根的路径包含了这个点长度为maxlen(p)的字符串的每一个后缀。
* SAM总是一个DAG，c(p,x)和link树都不一定是按照编号连续的拓扑序，但是link树拓扑序就是maxlen递减序。
* SAM的总节点个数不超过2n，如果稀疏存储，总边数不超过3n。

【代码】

* c(x,y)转移边
* maxlen(x)状态x代表的字符串的最长长度
* link(x)状态x的endpos集合的最大合法子集所在状态
* mx(x)状态x代表的每个字符串在原串的出现次数，SAM建完后按照link树拓扑序维护(注意特判mx(1)=0，对计算num有影响)
* ch(x)状态x节点的字符
* end(x)状态x中的字符串在原串中第一次出现的右端点下标
* num0(x)从状态x出发，后继本质不同的子串个数
* num1(x)从状态x出发，后继子串个数，例如:num1(1)=n(n+1)/2
* clone(x)状态x是否为q生成的nq克隆状态，克隆状态初始mx=0，其他继承原状态
* accept(x)状态x所代表的字符串集合是否是原串后缀
* link树拓扑序按照maxlen桶排
* num0为后继c(p,x)的num0之和+1，而num1是+mx，注意根节点num0不+1，因为不计算空串。
* 维护Right集合(这个点内字符串的所有出现位置的右端点): 在每个点开一个动态开点线段树，按照link树进行线段树合并。只在非克隆状态p处插入end(p)。

【应用】

* 本质不同子串个数 = Σmaxlen(p)-maxlen(link(p)) = num0(1)，不含空串。
* 一个串是否为子串:按转移边运行是否能停在某个节点上
* 一个串是否为后缀:按转移边运行是否能停在某个accept节点上
* 求字典序第k小子串:通过num0(本质不同)或num1(所有子串)像平衡树一样在转移边上转移求出答案。
* 最小循环表示:原串倍长后字典序转移走n步。
* 维护不同子串的一些价值:如果价值可以拆成每一位来计算(比如数字串的值)，按照转移边DP；如果价值只与长度和出现次数有关，直接用maxlen和mx来计算。
* 查询某个串出现次数:状态的mx(在线查询LCT)
* 查询某个串第一次出现:状态的[end-|S|+1,end]
* 单次查询输出某个串所有出现位置: 遍历link子树内，输出每一个非clone节点的end。(结合下一条考虑)。
* 多次查询关于某个串出现位置的信息(例如在[l,r]内的出现次数):主席树维护Right集合，相当于查询S对应的节点在区间的Right集合中[l+|S|-1,r]中数字个数。见【CF700E】和【CF666E】。
* 查询不是文本串子串的最短串:转移边上BFS
* 求多串LCSubstring:每个串匹配第一个串，将每个节点的匹配长度标记按照link树上传，节点取min，最后对所有节点取max。

# 广义后缀自动机SAM:

【代码】

* end, num0, num1在广义SAM中没有明确定义，其他与SAM相同
* 每次插入字符串从last=1重新开始
* 访问已存在节点且maxlen合法时不新增节点，置last为当前点并将mx+1。
* 如果是给定Trie树建对应的SAM，直接在Trie树上DFS，进入一个点时记录当前点，建立时直接传入p而不是p=last。见【BZOJ3926诸神眷顾的幻想乡】。

【应用】

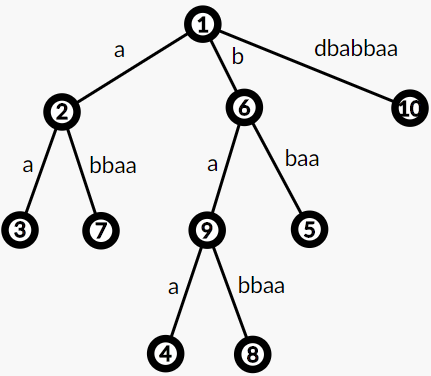
* SAM推广到多串问题

# 后缀自动机构造后缀数组

【性质】

* 串S的后缀自动机link树与反串S的后缀树同构:

例:反串aabbabd的link树与原串dbabbaa的后缀树都为:



* 后缀树字典序DFS序访问每个后缀标记节点，就得到SA序

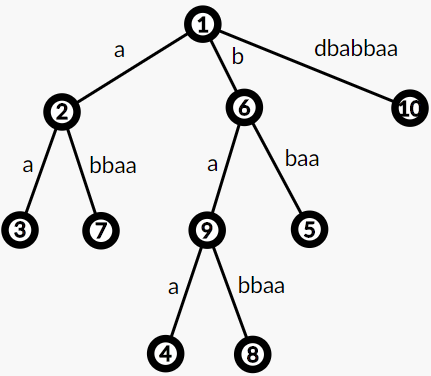
【代码】

问题就是如何将自动机的link树转化为后缀树，因为link边的字典序关系是不确定的。注意到link边就已经确定了LCP，例如所有连向2的link边(3,7)都有LCP = "a"。

又由于后缀树是由Trie压缩而来，同一LCP位置(深度相同的位置)，转移边字符一定不同，所以这个作为唯一关键字就可以完成排序了。

所以问题的关键就是确定每条link边代表的字符串的首字母。

原串dbabbaa，保持原串不变，逆序建后缀自动机:

对于每个节点记录beg(i)表示状态i(而不是i的字符串)最后一次出现的下标。与平常SAM的end记录第一次出现的下标不同，区别就在于每次clone一个节点的时候将新的时间置为clone节点的时间。

因为这时逆序建后缀自动机，所以这实际上找到了当前状态第一次出现的开头位置(ab的反串ba的后一位a就是原串ab的前一位)。要找到其状态的LCP，就是其前驱状态的最长长度。

所以这个位置就是beg(i)+maxlen(link(i))。

所以最后用后缀自动机构建后缀数组的方法就有了:从n到1建后缀自动机并维护beg，将每个节点按照s[beg(i)+maxlen(link(i))]桶排，从小到大加边(如果是vector从小到大；如果是邻接表，边后进先出，从大到小)，然后DFS就可以了。DFS的时候只考虑非clone节点，将其beg赋为当前的SA就可以了。

实现见代码SAMSA.cpp，代码中也用后缀数组的方法构造了Height可以直接通过UOJ35。

如果字符集更大求后缀数组，只能用map或倍增SA了。

--2018.03.03 模板修正

- 修改一些换行

- 消除start

- SAMSA Height与SA同步

- SAMSA end 重命名为 beg