suffix automaton

后缀自动机

yc

Suffix tree

后缀树是什么东西

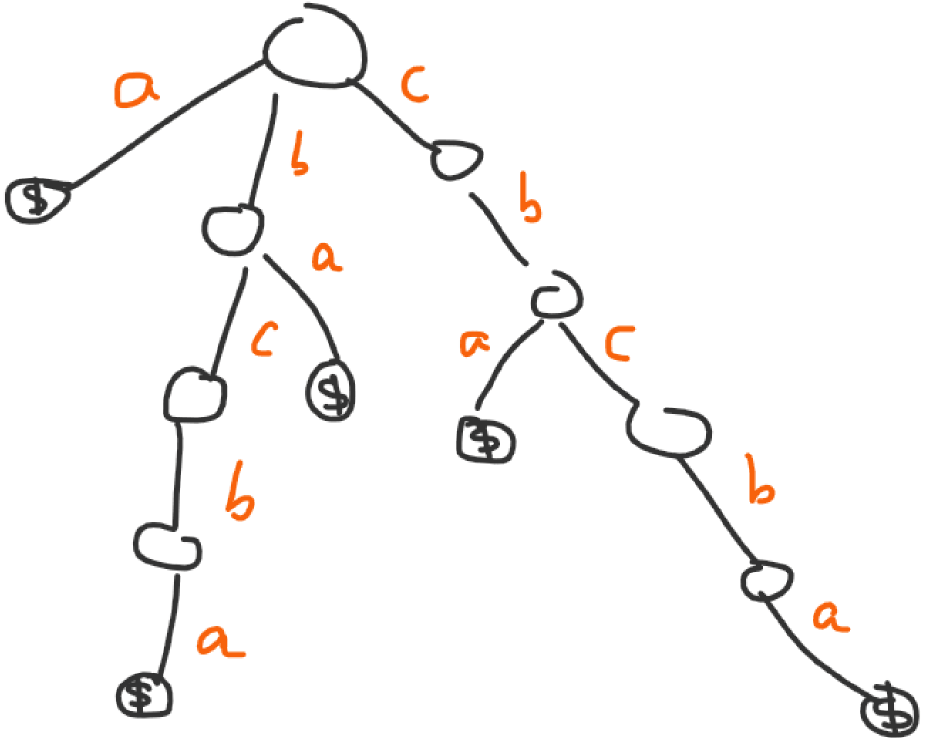
对于一个字符串，我们把它的所有后缀依次插入$Trie$树，最后得到的树就是后缀树

长什么样的?

以字符串cbcba为例

其后缀cbcba，bcba，cba, ba，a

结果如右图所示



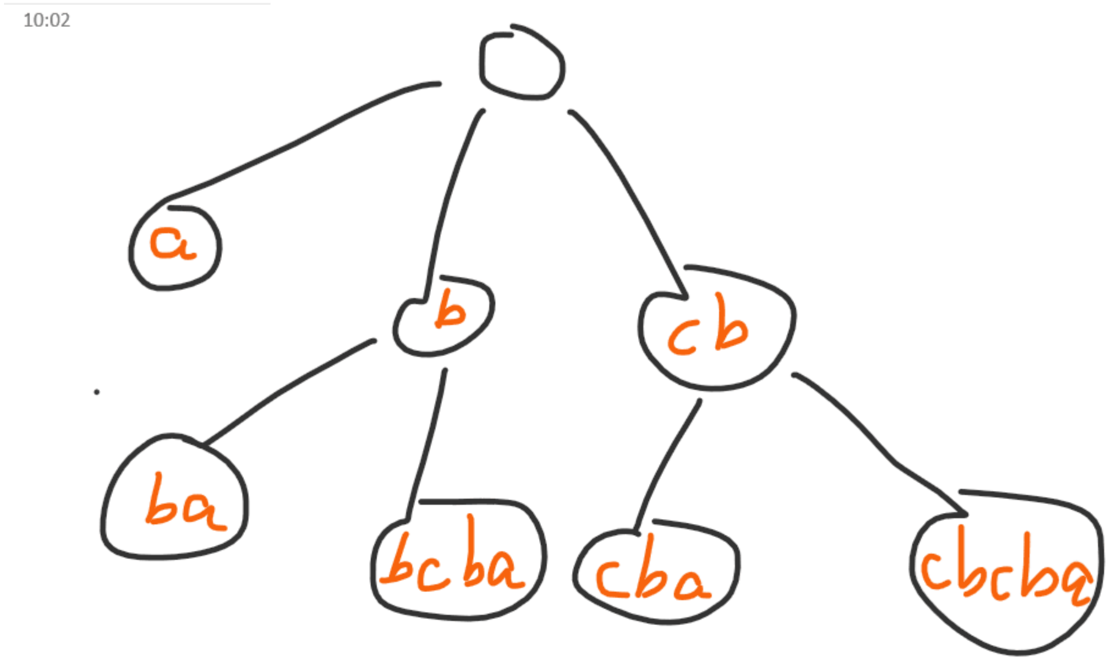
Suffix tree

后缀树是什么东西

看上去好像节点数的数量是$O(N^2)$级别的

注意到右图蓝线部分，在树上其实是单一转移，所以我们可以路径压缩

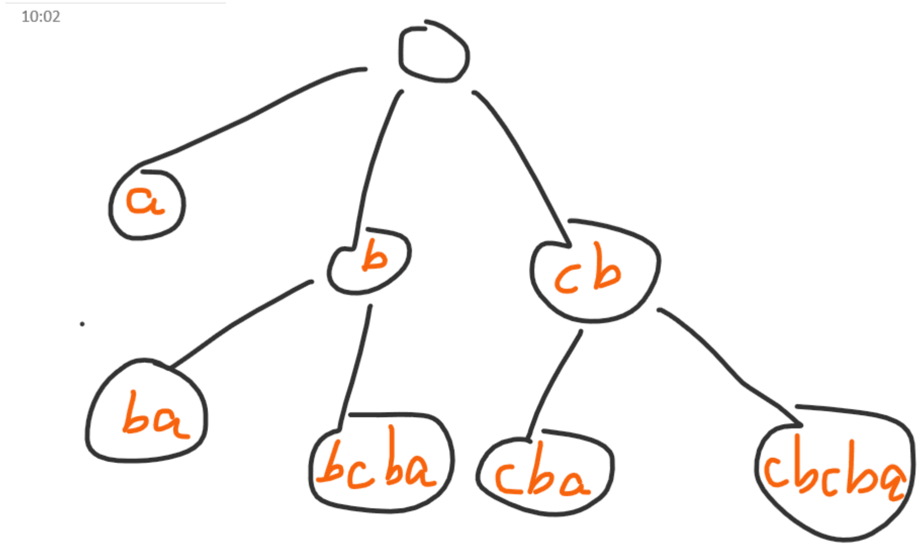
为什么这样点数量就是$O(N)$级别的呢?



Suffix tree

后缀树有什么性质

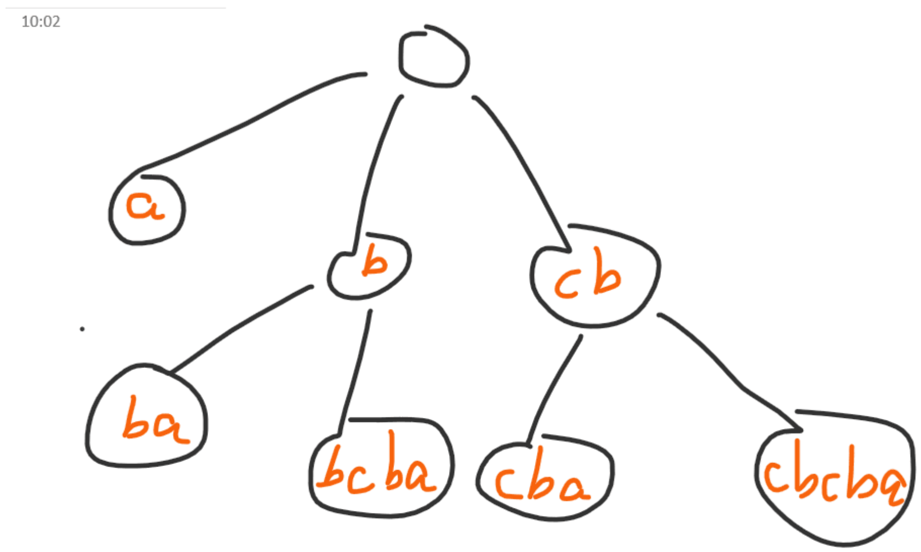
- 可以发现路径压缩之后，每个节点表示了原串s中的多个连续子串
- 记其中最短的长度为$lowest(x)$，最长的长度为$largest(x)$，$str(x)$为节点x上的字符串
- x管辖的字符为 $str(x)$长度在$[lowest(x), largest(x)]$的前缀
- $largest(x) = len(str(x))$
- $\mathrm{lowest}(x) = largest(fa_x)$
- 那么记S_x为这些子串的集合
- s的任意子串属于且仅属于一个集合



Suffix tree

后缀树有什么性质

- 记$startpos(t)$为子串t在s中所有出现位置开始位置的集合，$startpos(\text{“}cb\text{”}) = \{1,3\}$
- $startpos(t1) = startpos(t2)$，当且仅当$t1, t2 \in s_X$
- 不妨设$ss(x)$为节点x表示的任意字符串
- $startpos(ss(x)) \subseteq \mathrm{startpos}(\mathrm{ss}(\mathrm{fa}_x))$
- 若x, y不互为祖先，$startpos(x) \cap startpos(y) = \emptyset$
- $|startpos(t)| = t$在s中出现次数
- $|startpos(\mathrm{ss}(x))| = x$后缀树子树中终止符的个数



做一下题

以下问题假设你已经建出了后缀树，知道了$fa(x)$,$largest(x)$以及每个节点是否有终止符

BZOJ3238

一个长度为n的字符串S，令T_{i}表示它从第i个字符开始的后缀，求：

$$\sum_{1 \le i < j \le n} len(T_{\mathrm{i}}) + len\left(T_j\right) - 2 * lcp(T_{\mathrm{i}}, T_j)$$

BZOJ3238

$$=\sum_{1\le i<j\le n} len(T_{\mathrm{i}})+len(T_j)-2*\sum_{1\le i<j\le n} lcp(T_{\mathrm{i}},T_j)$$

$$=\sum_{1\le i<j\le n}(n-i+1)+(n-j+1)-2*\sum_{1\le i<j\le n} lcp(T_{\mathrm{i}},T_j)$$

$$=\frac{(\mathrm{n}+1)*\mathrm{n}*(\mathrm{n}-1)}{2}-2*\sum_{1\le i<j\le n} lcp(T_{\mathrm{i}},T_j)$$

关键是求出$\sum_{1\le i<j\le n} lcp(T_{\mathrm{i}},T_j)$

BZOJ3238

关键是求出$\sum_{1\le i<j\le n} lcp(T_{\mathrm{i}},T_j)$

设后缀T_{i},T_j在后缀树上的父亲为lca，那么$lcp\left(T_{\mathrm{i}},T_j\right)=largest(lca)$

预处理出$\mathrm{val}(x)=x$子树中终止符的个数，也就是有多少个后缀包含$str(x)$的前缀

那么树形背包dp就好了

BZOJ4556

- 有一个长度为$n \le 100000$的字符串和$m \le 100000$个询问
- 每次询问子串$s[a..b]$的所有子串和$s[c..d]$的最长公共前缀的长度的最大值是多少

BZOJ4556

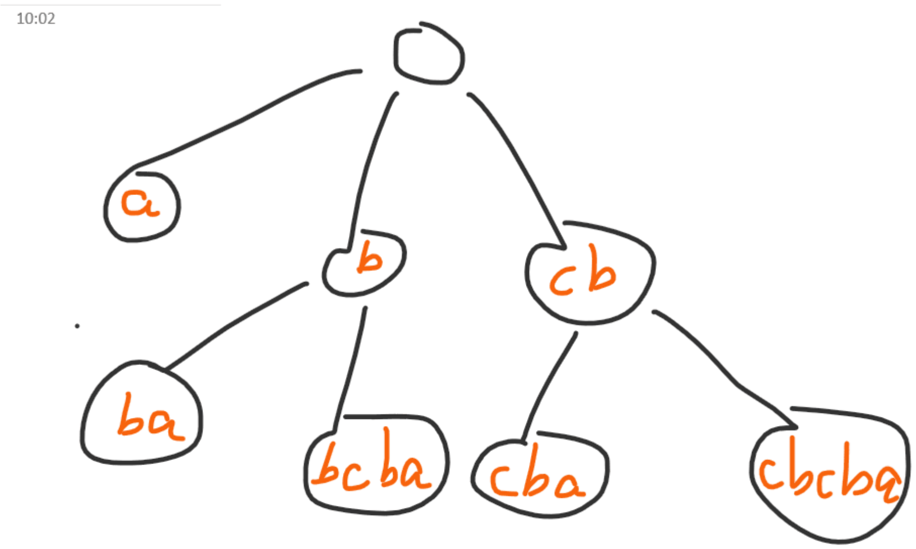
- 令T_{i}表示s从第i个字符开始的后缀，那么问题等价于求
- $\min(d-c+1, max_{a\le k\le b}(lcp(T_k, T_d)))$
- 二分答案ans，在后缀树上从后缀d所在的节点往上跳到$largest(x)\ge ans$且深度最深的节点x
- 那么我们只需要查询节点x的子树中是否存在$[c+ans-1, d]$范围内的后缀
- 每个节点开一棵权值线段树，启发式合并就好了
- 复杂度$O(n\ log^2\ n)$

Suffix tree

后缀树有什么性质

- 可以发现建出后缀树和得到相关信息以后，用dp或者数据结构维护后缀乃至子串的信息会变得非常方便

- 现在的问题就是如何建出后缀树



suffix automaton

后缀自动机是什么东西

- 对给定字符串s的后缀自动机是一个最小化确定有限状态自动机，它能够接收字符串s的所有后缀。

当前所匹配完成的字符串　　下一个匹配的字符

- DAG，V 表示状态，　　E表示转移

空集

- 某一状态$T0$被称作初始状态，由它能够到达其余所有状态
- 一个或多个状态被标记为终止状态。如果我们从初始状态$T0$经由任意路径走到某一终止状态，并顺序写出所有经过边的标记，得到的字符串必然是s的某一后缀。
- 在符合上述诸条件的所有自动机中，后缀自动机有最少的顶点数。

结束位置endpos，它们的性质及与后缀自动机的联系：

考虑字符串s的任意非空子串t。我们称终点集合endpos(t)为：s中所有是t出现位置终点的集合。

我们称两个子串t_1和t_2“终点等价”，如果它们的终点集合一致：endpos(t_1)=endpos(t_2)。因此，所有s的非空子串可以根据终点等价性分成若干类。

事实上对后缀自动机，终点等价字符串仍然保持相同性质。换句话说，后缀自动机中状态数等价于所有子串的终点等价类个数，加上初始状态。每个状态对应一个或多个拥有相同终点集合的子串。

我们将这一陈述作为假定，然后描述一个基于此假设的，线性时间构建后缀自动机的算法——正如我们不久后将会看到的，所有后缀自动机的必须性质，除最小性（即最少顶点数），都将被满足（最小性由Nerode产生，见参考文献）。

关于终点集合，我们给出一些简单但重要的事实。

endpos

结束位置的性质与后缀自动机的联系

- 考虑字符串s的任意非空子串t。我们称终点集合$endpos(t)$为：s中所有是t出现位置终点的集合。
- 我们称两个子串t_1和t_2“终点等价”，如果它们的终点集合一致：$endpos(t\_1) = endpos(t\_2)$。因此，所有$s$的非空子串可以根据终点等价性分成若干类。

- **引理**1：两个非空子串u和v（$length(u) \leq length(v)$）是终点等价的，当且仅当u在字符串s中仅作为w的后缀出现。

- **引理**2：考虑两个非空子集u, w（$length(u) \leq length(w)$）。它们的终点集合不相交，或者$endpos(w)$是$endpos(u)$的子集。进一步地，这取决于u是否是w的后缀

- **引理**3：考虑一个终点等价类。将该等价类中的子串按长度递减排序。排序后的序列中，每个子串将比上一个子串短，从而是上一个字串的后缀。换句话说，某一终点等价类中的字符串互为后缀，它们的长度依次取区间$[x, y]$内的所有数。

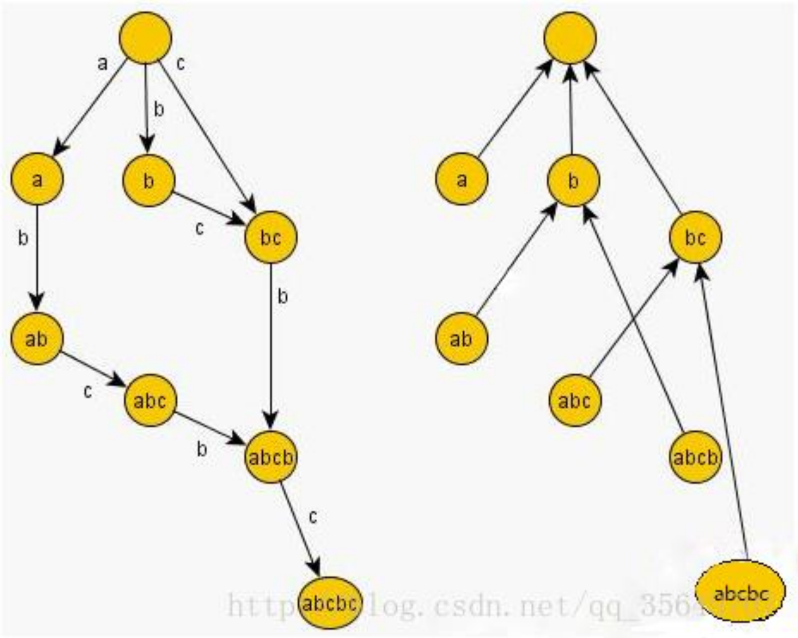
suffix link

后缀链接

- 考虑一个状态$v \neq t_0$.就我们目前所知，有一个确定的子串集合，其中元素和v有着相同的终点集合。

- 并且，如果我们记w是其中的最长者，其余子串均是w的后缀。我们还知道w的前几个后缀（按照长度降序）在同一个终点等价类中，其余后缀（至少包括空后缀）在别的终点等价类中。

- 令t是第一个这样的后缀——对它我们建立后缀链接。换言之，v的后缀链接$link(v)$指向在不同等价类中的w的最长后缀。



- 1. 令last为对应整个字符串的状态（最初last=0，在每次字符添加操作后我们都会改变last的值）。

- 2.建立一个新的状态cur，令len(cur)=len(last)+1，而link(cur)的值并不确定。

- 3. 我们最初在last，如果它没有字符c的转移，那就添加字符c的转移，指向cur，然后走向其后缀链接，再次检查——如果没有字符c的转移，就添加上去。如果在某个节点已有字符c的转移，就停止，并且令p为这个状态的编号。

- 4.如果“某节点已有字符c的转移”这一事件从未发生，而我们来到了空状态-1（经由t_0的后缀指针前来），我们简单地令link(cur)=0，跳出。

- 5.假设我们停在了某一状态q，是从某一个状态p经字符c的转移而来。现在有两种情况：len(p)+1=len(q)或不然。

- 6.如果len(p)+1=len(q)，那么我们简单地令link(cur)=q，跳出。

- 7.否则，情况就变得更加复杂。必须新建一个q的“拷贝”状态：建立一个新的状态clone，将q的数据拷贝给它（后缀链接，以及转移），除了len的值：需要令len(clone)=len(p)+1.

- 8.在拷贝之后，我们将cur的后缀链接指向clone，并将q的后缀链接重定向到clone。

- 9.最终，我们需要做的最后一件事情就是——从p开始沿着后缀链接走，对每个状态我们都检查是否有指向q的，字符c的转移，如果有就将其重定向至clone（如果没有，就终止循环）。

- 10.在任何情况下，无论在何处终止了这次添加操作，我们最后都将更新last的值，将其赋值为cur。

suffix automaton

后缀自动机的应用（1）

- 可以发现后缀树的$startpos(x)$和后缀自动机的$endpos(x)$存在$startpos(x) = rev(endpos(x))$的关系
- 也就是说原串的后缀树是反串后缀自动机建出的后缀链接
- 我们就可以用后缀自动机在$O(N)$的复杂度内建出后缀树，并处理后缀树相关的问题

suffix automaton

后缀自动机的应用（2）

- 显然作为自动机，可以把模式串t在文本串s中匹配，并求出t和s的某些关系，比如出现次数，最长公共前缀等信息

suffix automaton

后缀自动机的应用（3）

- 后缀自动机本身作为DAG，也可以在上面进行dp，如求不同子串的个数，第k小的子串等。

做一下题

BZOJ2555

在线完成两个操作，每次在当前字符串s后面插入一个字符串，或者询问串t在当前字符串中出现的次数

$$\sum \mathrm{len}_s \le 600000\ , \quad \sum \mathrm{len}_\mathrm{t} \le 300000$$

RunID	User	Problem	Result	Memory	Time	Language	Code_Length	Submit_Time
2526164	Matchperson	2555	Accepted	165608 kb	21580 ms	C++	3294 B	2018-01-17 16:45:37
2506600	Matchperson	2555	Accepted	165604 kb	21196 ms	C++	3314 B	2018-01-08 07:57:45

BZOJ2555

- 对于每个询问就是在后缀自动机上找到该子串所对应的节点 找不到返回0
- 这个节点在后缀树上$|startpos(x)|$的大小就是这个子串的出现次数
- 每次插入新的字符串的时候,后缀树的形态和权值会发生改变，所以需要LCT维护

HHHOJ187

你有一个字符串S，一开始为空串，要求支持两种操作：

ID	题目	提交者	结果	用时	内存	语言	文件大小	提交时间
#6823	#187. Hashit	Axcosin	100	67ms	19024kb	C++	1.5kb	2018-01-28 16:29:37
#6811	#187. Hashit	Axcosin	100	69ms	19288kb	C++	1.4kb	2018-01-28 13:38:02

问每次操作之后S有多少个两两不同的连续子串。

HHHOJ187

- 操作1就是后缀自动机的操作
- 本质不同的子串个数就是$\sum largest(x) - lowest(x) + 1 = \sum largest(x) - largest(fa_x)$，这个可以在建后缀树每次$link$和$cut$的时候维护
- 每插入一个字符的时候记录一下当前后缀自动机的更改（新建了哪些点和哪些边）。
- 删除的时候直接回退就可以了
- 删边的复杂度不知道能不能保证，因此建边的时候可以额外附带一个属性，表示这条边是在加第几个字符的时候建的，删除直接对这个字符打上删除标记，连新边的时候判断一下就好了
- 复杂度$O(N)$

HHHOJ193

题目描述

有一个字符串 S（初始为空），你的程序需要支持一下操作：

$+a$: a 是一个字符串，表示在 S 后面加上 a

$?a$: a 是一个字符串，表示询问 a 在 S 中出现的次数

R: 表示将 S 整个翻转

所有的字符串仅包含小写字母

输入格式

第一行一个正整数 n 表示操作次数，以后 n 行，每行表示一个操作。

输出格式

对于每个询问，输出一行答案

HHHOJ193

- 离线建出最终的字符串，从后往前考虑每个操作

- 操作3可以打标记

- 操作1可以根据操作3判断每次删除一个前缀还是一个后缀

- 操作2如果存在标记，就把模式串翻转，在后缀自动机中匹配，将匹配到的节点x在后缀树中考虑，若当前剩下的字符串区间为$[l,r]$，那么其出现次数就是x节点子树中$[l,r]$范围内后缀的个数。

tks

谢谢大家