高级字符串算法：后缀自动机、后缀数组、回文自动机

Ebola

Institute of Mathematics,
Zhejiang University.

July, 2023

模板题

给定一个只包含小写字母的字符串 S，求 S 的所有出现次数不为 1 的子串的出现次数乘上该子串长度的最大值。$|S| \leq 10^6$

什么是 SAM

后缀自动机（SAM）是一种特殊的有限状态自动机（DFA），它符合以下性质：

- 转移图是有向无环图，且每个转移接受且只接受一个字符
- 接受且只接受 S 的所有后缀
- 在符合以上性质的基础上，节点数最少

如果只考虑前两条，其实只要把 S 的所有后缀插入字典树即可，节点数有 $O(n^2)$ 个。

代码

```
1  void insert(int c)
2  {
3      int p=lst,np=++tot;len[np]=len[p]+1;
4      while(p&&!ch[p][c]) ch[p][c]=np,p=prt[p];
5      if(!p) prt[np]=1;
6      else
7      {
8          int q=ch[p][c];
9          if(len[p]+1==len[q]) prt[np]=q;
10         else
11         {
12             int nq=++tot;len[nq]=len[p]+1;
13             memcpy(ch[nq],ch[q],sizeof(ch[nq]));
14             prt[nq]=prt[q];prt[q]=prt[np]=nq;
15             while(ch[p][c]==q) ch[p][c]=nq,p=prt[p];
16         }
17     }
18     sz[np]=1;lst=np;
19 }
```

这是 SAM 的核心代码，很短。

endpos 等价类

我们首先提出 endpos 等价类这个概念。$endpos(P)$ 表示模式串 P 在 S 中所有出现的结束位置的集合，那么 $endpos$ 相同的模式串就可以共用一个节点。例如串"aababc" 中，"b" 和"ab" 的 endpos 都是 $\{3, 5\}$，说明他们总是一起出现，所以可以共用一个节点。我们把 $endpos$ 相同的模式串组成的集合称为一个 $endpos$ **等价类**。

endpos 等价类

我们首先提出 endpos 等价类这个概念。$endpos(P)$ 表示模式串 P 在 S 中所有出现的结束位置的集合，那么 $endpos$ 相同的模式串就可以共用一个节点。例如串"aababc" 中，"b" 和"ab" 的 endpos 都是 $\{3,5\}$，说明他们总是一起出现，所以可以共用一个节点。我们把 $endpos$ 相同的模式串组成的集合称为一个 $endpos$ **等价类**。

显然，对于一个等价类内的两个模式串 P_1 和 P_2，如果 $|P_1| < |P_2|$，那么 P_1 一定是 P_2 的后缀，因此，在一个等价类中，一定存在一个**最长**的字符串 P，等价类中的所有字符串都是 P 的后缀。

endpos 等价类

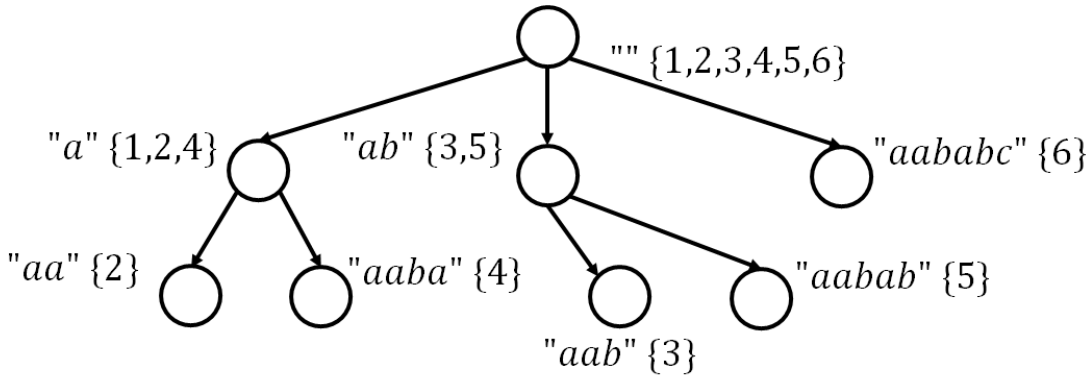
我们首先提出 endpos 等价类这个概念。$endpos(P)$ 表示模式串 P 在 S 中所有出现的结束位置的集合，那么 $endpos$ 相同的模式串就可以共用一个节点。例如串"aababc" 中，"b" 和"ab" 的 endpos 都是 $\{3,5\}$，说明他们总是一起出现，所以可以共用一个节点。我们把 $endpos$ 相同的模式串组成的集合称为一个 $endpos$ **等价类**。

显然，对于一个等价类内的两个模式串 P_1 和 P_2，如果 $|P_1| < |P_2|$，那么 P_1 一定是 P_2 的后缀，因此，在一个等价类中，一定存在一个**最长**的字符串 P，等价类中的所有字符串都是 P 的后缀。

根据位于 P **前面的那一个字符**是什么，可以把原等价类划分为若干个等价类。例如在"aababc" 中，我们已经知道"ab" 出现了两次，但首次出现时前面的那一个字符是"a"，再次出现时前面那个字符是"b"，而"aab" 与"bab" 属于两个不同的等价类，因此我们可以把等价类 $\{3,5\}$ 划分成 $\{3\}$ 和 $\{5\}$

endpos 等价类

我们可以认为，空串的 $endpos$ 等价类是 $\{1, ..., n\}$，而所有等价类都是由此一步一步划分来的，形成一个树的结构，总节点数不会超过 $2n - 1$



这棵树就是 parent 树，他的节点和 SAM 的状态一一对应。注意，划分过程中可能会丢失元素，例如图中"a" 第一个出现的位置前面没有字符，所以划分后会丢失一个元素，这一点在做 dp 的时候很关键。

构造 SAM

采取动态构造的方法，一个一个地把字符加进去。考虑向一个已知的字符串的后面添加一个字符，它的 SAM 会如何变化。

构造 SAM

采取动态构造的方法，一个一个地把字符加进去。考虑向一个已知的字符串的后面添加一个字符，它的 SAM 会如何变化。

设 go[st][ch] 表示状态 st 接受字符 ch 后转移到的状态，fa[st] 为该状态在 parent 树上的父节点，len[st] 为该状态对应的 $endpos$ 等价类中最长串的长度，last 为加入新字符前整个字符串所在的等价类对应的状态。

构造 SAM

采取动态构造的方法，一个一个地把字符加进去。考虑向一个已知的字符串的后面添加一个字符，它的 SAM 会如何变化。

设 go[st][ch] 表示状态 st 接受字符 ch 后转移到的状态，fa[st] 为该状态在 parent 树上的父节点，len[st] 为该状态对应的 $endpos$ 等价类中最长串的长度，last 为加入新字符前整个字符串所在的等价类对应的状态。

一个基本的观察是：从 last 开始在 parent 树上往上爬，一定能遍历加入新字符前所有后缀的对应节点。加入一个新的字符，其实就是给之前部分后缀新增了转移。

构造 SAM

采取动态构造的方法，一个一个地把字符加进去。考虑向一个已知的字符串的后面添加一个字符，它的 SAM 会如何变化。

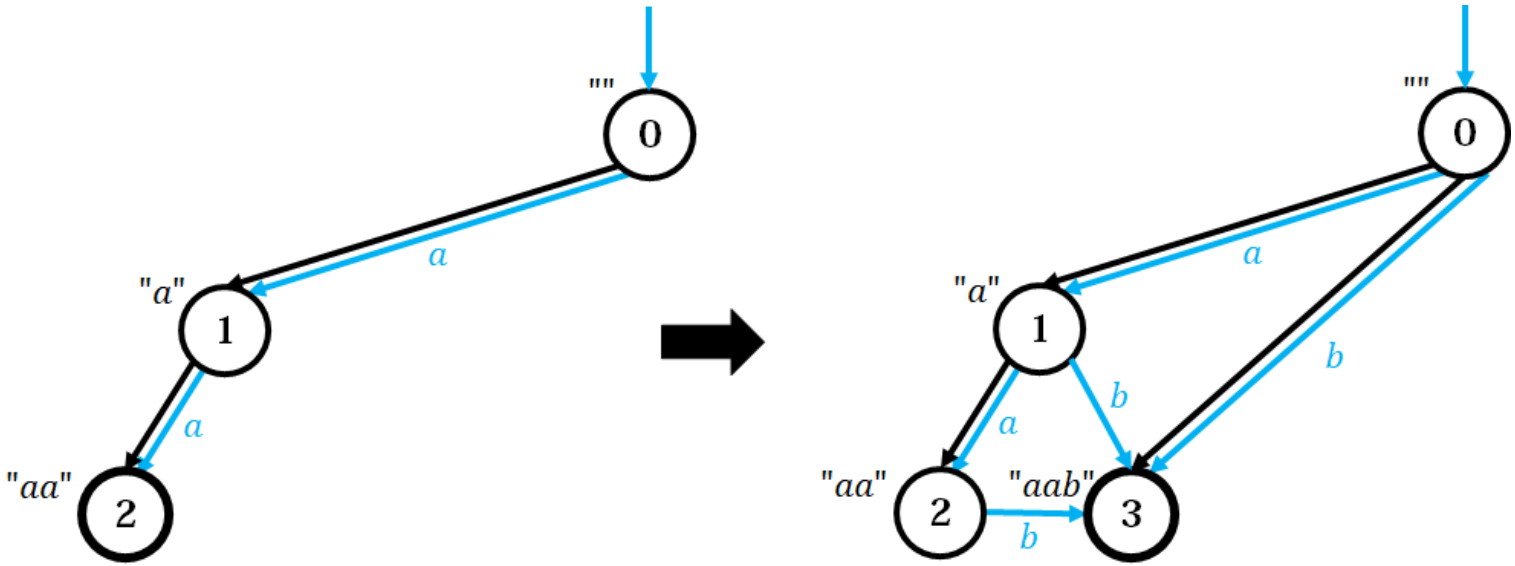
设 go[st][ch] 表示状态 st 接受字符 ch 后转移到的状态，fa[st] 为该状态在 parent 树上的父节点，len[st] 为该状态对应的 $endpos$ 等价类中最长串的长度，last 为加入新字符前整个字符串所在的等价类对应的状态。

一个基本的观察是：从 last 开始在 parent 树上往上爬，一定能遍历加入新字符前所有后缀的对应节点。加入一个新的字符，其实就是给之前部分后缀新增了转移。

所以当新增字符 ch 时，我们创建一个新状态 cur（其中 len[cur]=len[last]+1），然后从 last 开始往上爬，对于遇到的每个状态 p，如果 p 还不能通过 ch 转移，那我们就新增一个转移 go[p][ch]=cur，然后继续往上爬，直到某个 p 可以通过 ch 转移到状态 q ，或者处理完根节点为止。这分为三种情况

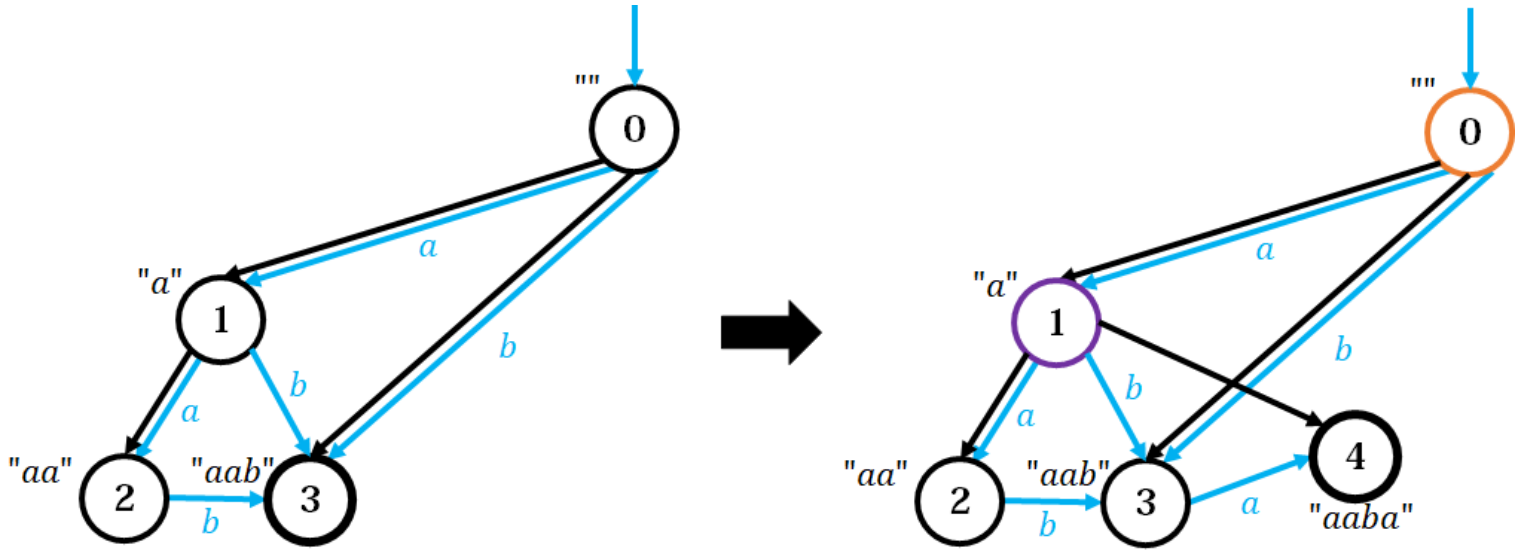
第一种情况

没有找到要找的 q。这只可能出现在加入了从未加入过的字符的时候，此时直接令 fa[cur] 为根节点然后退出即可。（当然退出后还需要把 last 设为 cur，这对三种情况都是一样的）



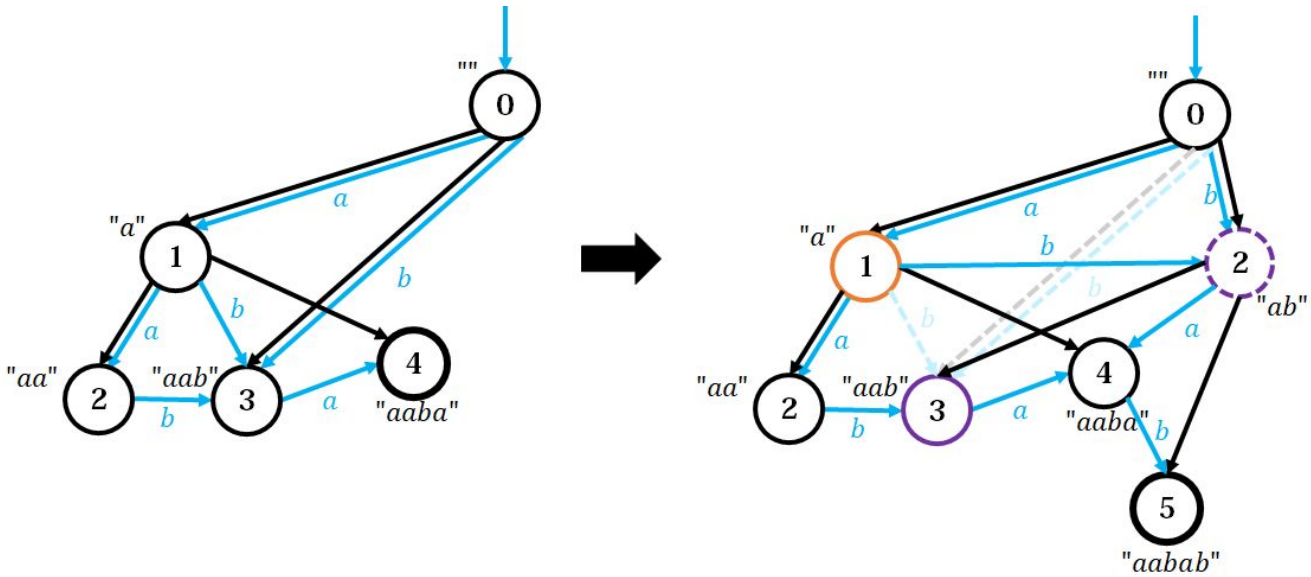
第二种情况

找到了 q，且 len[p]+1==len[q]。这种情况下我们直接令 fa[cur]=q 然后退出即可。这是因为 p 所对应集合中每个字符串在后面加上一个 ch 都能构成一个 q 对应集合的字符串，而 p 对应集合都是原字符串的后缀，所以 q 对应集合都是新字符串的后缀，应作为 cur 的父亲节点。



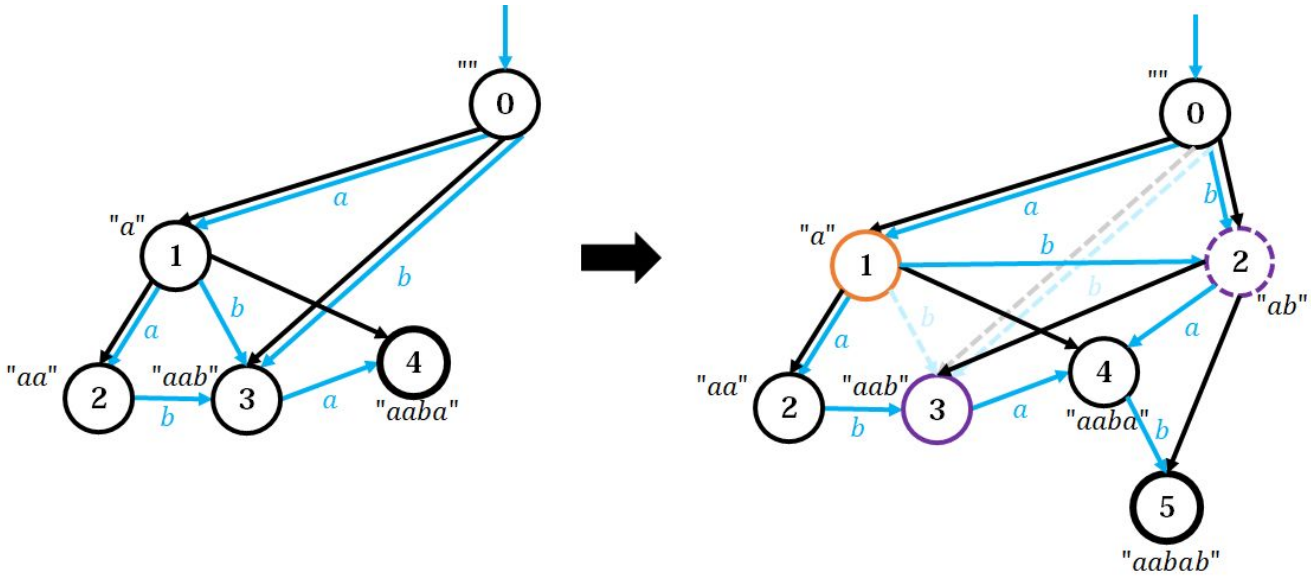
第三种情况

找到了 q，但 len[p]+1≠len[q]。我们新建一个 r 节点，它拥有 q 节点的所有出边，且 fa 也与 q 节点相同，但是 len[r]=len[p]+1。我们从 p 节点继续往上爬，把所有接受 ch 而到达 q 的转移的目标改为 r（注意只要有一个节点不能接受 ch 那它的祖先都不能接受 ch，要及时退出循环）。最后令 fa[cur]=fa[q]=r。



第三种情况

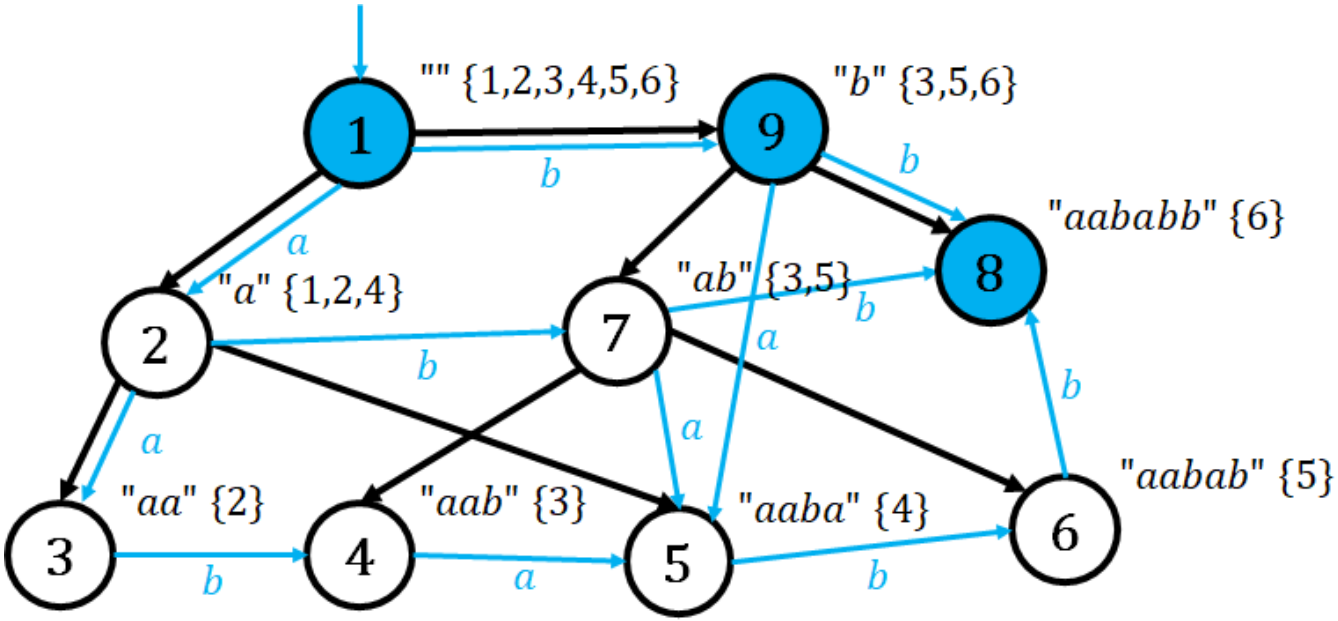
找到了 q，但 len[p]+1≠len[q]。我们新建一个 r 节点，它拥有 q 节点的所有出边，且 fa 也与 q 节点相同，但是 len[r]=len[p]+1。我们从 p 节点继续往上爬，把所有接受 ch 而到达 q 的转移的目标改为 r（注意只要有一个节点不能接受 ch 那它的祖先都不能接受 ch，要及时退出循环）。最后令 fa[cur]=fa[q]=r。



这里和第二种情况有本质区别，因为 q 不仅仅包含新字符串的后缀，比如下图中 3 号点除了"ab" 还包含了"aab"，我们不得不将它拆分开。

一个 SAM 的例子

下图是字符串"aababb" 构成的 SAM



复杂度：总状态数不超过 $2n-1$，总转移数不超过 $3n-4$，构建 SAM 的复杂度为 $O(mn)$，其中 m 为字符集大小。证明见 OIwiki

例题选讲

求模式串 P 是不是字符串 S 的子串。

例题选讲

求模式串 P 是不是字符串 S 的子串。
【解】建 S 的 SAM，把 P 放进去跑即可。

例题选讲

求模式串 P 在字符串 S 中的出现次数。

例题选讲

求模式串 P 在字符串 S 中的出现次数。

【解】建 S 的 SAM，其实就是求 p 对应节点的 $endpos$ 集合的大小，在 parent 树上 dp。

例题选讲

求模式串 P 在字符串 S 中的出现次数。

【解】建 S 的 SAM，其实就是求 p 对应节点的 $endpos$ 集合的大小，在 parent 树上 dp。

对于划分后丢失元素的节点（他所代表的串是 S 的前缀），dp 值为子节点 dp 值之和再加一，其余点则无需加一。其实只要在建 SAM 的时候让 dp[cur]=1 即可。

例题选讲

求模式串 P 在字符串 S 中的出现次数。

【解】建 S 的 SAM，其实就是求 p 对应节点的 $endpos$ 集合的大小，在 parent 树上 dp。

对于划分后丢失元素的节点（他所代表的串是 S 的前缀），dp 值为子节点 dp 值之和再加一，其余点则无需加一。其实只要在建 SAM 的时候让 dp[cur]=1 即可。

小技巧：在 parent 树上 dp 的时候不需要 dfs，只要把节点按 len 做一个桶排，然后从大到小枚举，就等价于在 parent 树上自底向上 dp。

例题选讲

求字符串 S 本质不同的子串个数。

例题选讲

求字符串 S 本质不同的子串个数。

【解】建 S 的 SAM，从 parent 树的角度考虑，每个节点代表了 len[p]-len[fa[p]] 个子串（因为长度不同，自然互不相同）。另外由 parent 树的划分意义，不同节点代表的子串一定都不同，且任何一个子串必然被 parent 树上某一点代表。累加该值即可。

例题选讲：SPOJ 1811

求字符串 S 与 T 的最长公共子串。

例题选讲：SPOJ 1811

求字符串 S 与 T 的最长公共子串。

【解】建 S 的 SAM，对 T 的每个前缀，求他在 S 中出现过的最长后缀。其实就是把 T 放进去跑，一直到跑不动了就跳到 parent 树上的父节点继续跑，跳 parent 的时候要维护当前最长后缀的长度。

例题选讲：SPOJ 1811

求字符串 S 与 T 的最长公共子串。

【解】建 S 的 SAM，对 T 的每个前缀，求他在 S 中出现过的最长后缀。其实就是把 T 放进去跑，一直到跑不动了就跳到 parent 树上的父节点继续跑，跳 parent 的时候要维护当前最长后缀的长度。

```
1  int lcs(char *s)
2  {
3      int m=strlen(s), l=0, p=1, ans=0;
4      for(int i=0;i<m;i++)
5      {
6          int c=s[i]-'a';
7          while(p!=1 && go[p][c]==0)
8              p=fa[p], l=len[p]; //维护当前最长后缀长度
9          if(go[p][c]) p=go[p][c], l++;
10         ans=max(ans,l);
11     }
12     return ans;
13 }
```

例题选讲：(UVA 719) (P1368) 最小表示法

有 N 组数据，每组给你一串字符串，但是这串字符串是环形的，让你找个位置切开，使得它的字典序最小，输出切开的位置（如果答案不唯一，输出最小位置）

例题选讲：(UVA 719) (P1368) 最小表示法

有 N 组数据，每组给你一串字符串，但是这串字符串是环形的，让你找个位置切开，使得它的字典序最小，输出切开的位置（如果答案不唯一，输出最小位置）

【解】遇到循环串的问题一般都先把串复制一遍得到 SS，建 SS 的 SAM。

例题选讲：(UVA 719) (P1368) 最小表示法

有 N 组数据，每组给你一串字符串，但是这串字符串是环形的，让你找个位置切开，使得它的字典序最小，输出切开的位置（如果答案不唯一，输出最小位置）

【解】遇到循环串的问题一般都先把串复制一遍得到 SS，建 SS 的 SAM。

SS 的 SAM 能且仅能接受 SS 的所有子串，所以从根节点出发跑 $|S|$ 步得到的就是 S 的某个循环排列，一直沿最小字典序的转移边跑即可。

例题选讲：UVA 719

给出一个字符串 S，长度不超过 90000。询问 q 次，每次给一个 k，求所有本质不同的子串中，字典序第 k 小的。

例题选讲：UVA 719

给出一个字符串 S，长度不超过 90000。询问 q 次，每次给一个 k，求所有本质不同的子串中，字典序第 k 小的。

【解】先建 S 的 SAM，求出 $size[p]$ 表示从 p 出发能跑出几个子串，逆拓扑序 dp 即可。

例题选讲：UVA 719

给出一个字符串 S，长度不超过 90000。询问 q 次，每次给一个 k，求所有本质不同的子串中，字典序第 k 小的。

【解】先建 S 的 SAM，求出 $size[p]$ 表示从 p 出发能跑出几个子串，逆拓扑序 dp 即可。

对于一个询问，从根节点出发，每次从小到大枚举字母 ch，如果 $size[go[cur][ch]] \geq k$，就往 ch 走；否则 $k = k - size[go[cur][ch]]$，继续枚举 ch

例题选讲：LNOI2022 串

多组数据，每次给定一个英文小写字母构成的字符串 S，你需要找到一个尽可能长的字符串序列 $(T_0, T_1, \ldots, T_l)$，满足：

- T_0 是 S 的子串；
- $\forall 1 \le i \le l$，$|T_i| - |T_{i-1}| = 1$；
- $\forall 1 \le i \le l$，存在 S 的一个长度为 $|T_i| + 1$ 的子串 S'_i，使得 S'_i 的长度为 $|T_{i-1}|$ 的前缀为 T_{i-1}，长度为 $|T_i|$ 的后缀为 T_i。

输出这样的字符串序列的长度的最大值（即 l 的最大值）。

设 $\sum |S|$ 表示测试点中所有测试数据的字符串长度和。对于 100% 的测试数据，$1 \le |S| \le 5 \times 10^5$，$1 \le \sum |S| \le 1.5 \times 10^6$。

例题选讲：LNOI2022 串

【题解】观察条件：

- T_0 是 S 的子串；
- $\forall 1 \le i \le l$，$|T_i| - |T_{i-1}| = 1$；
- $\forall 1 \le i \le l$，存在 S 的一个长度为 $|T_i|+1$ 的子串 S'_i，使得 S'_i 的长度为 $|T_{i-1}|$ 的前缀为 T_{i-1}，长度为 $|T_i|$ 的后缀为 T_i。

用 $[i,j]$ 表示子串 $S[i...j]$，我们构造一个序列 $[i,j] \to [i-1,j-2] \to [i-2,j-4]...$，一直这样下去直到长度为 0 或到头了，得到一列子串，把它倒过来，就符合上面的条件。所以答案显然至少为 $\left\lfloor \frac{n}{2} \right\rfloor$。

例题选讲：LNOI2022 串

【题解】观察条件：

- T_0 是 S 的子串；
- $\forall 1 \le i \le l$，$|T_i| - |T_{i-1}| = 1$；
- $\forall 1 \le i \le l$，存在 S 的一个长度为 $|T_i|+1$ 的子串 S'_i，使得 S'_i 的长度为 $|T_{i-1}|$ 的前缀为 T_{i-1}，长度为 $|T_i|$ 的后缀为 T_i。

用 $[i,j]$ 表示子串 $S[i...j]$，我们构造一个序列 $[i,j] \to [i-1,j-2] \to [i-2,j-4]...$，一直这样下去直到长度为 0 或到头了，得到一列子串，把它倒过来，就符合上面的条件。所以答案显然至少为 $\left\lfloor \frac{n}{2} \right\rfloor$。

现在考虑正向过程。我们从 $T_0 = \varnothing$ 开始，每次左端点往右移一个，同时长度加一。如果想要答案更大，一定要有一个左端点回跳的过程。如果现在位于 $[l_2, r_2]$，前面恰好有一个 $[l_1, r_1]$ 和 $[l_2, r_2]$ 是一样的，就可以跳回前面。

例题选讲：LNOI2022 串

枚举最后一次回跳。那么我们现在的过程就是，挑选某一个出现至少两次的子串，设位置分别为 $[l_1, r_1]$ 和 $[l_2, r_2]$，从空集开始构造到 $[l_2, r_2]$，然后回跳到 $[l_1, r_1]$，接下来一直向右构造到底。答案就是

$$r_1 - l_1 + 1 + \left\lfloor \frac{n - r_1}{2} \right\rfloor$$

例题选讲：LNOI2022 串

枚举最后一次回跳。那么我们现在的过程就是，挑选某一个出现至少两次的子串，设位置分别为 $[l_1, r_1]$ 和 $[l_2, r_2]$，从空集开始构造到 $[l_2, r_2]$，然后回跳到 $[l_1, r_1]$，接下来一直向右构造到底。答案就是

$$r_1 - l_1 + 1 + \left\lfloor \frac{n - r_1}{2} \right\rfloor$$

还有一个问题：怎么确保能从空串一直构造到 $[l_2, r_2]$？

例题选讲：LNOI2022 串

枚举最后一次回跳。那么我们现在的过程就是，挑选某一个出现至少两次的子串，设位置分别为 $[l_1, r_1]$ 和 $[l_2, r_2]$，从空集开始构造到 $[l_2, r_2]$，然后回跳到 $[l_1, r_1]$，接下来一直向右构造到底。答案就是

$$r_1 - l_1 + 1 + \left\lfloor \frac{n - r_1}{2} \right\rfloor$$

还有一个问题：怎么确保能从空串一直构造到 $[l_2, r_2]$？

可以从 $[l_2, r_2]$ 开始，向左反向构造，到了 $[l_1, r_1]$ 内就跳回到 $[l_2, r_2]$ 中，再继续向左反向构造。

例题选讲：LNOI2022 串

枚举最后一次回跳。那么我们现在的过程就是，挑选某一个出现至少两次的子串，设位置分别为 $[l_1, r_1]$ 和 $[l_2, r_2]$，从空集开始构造到 $[l_2, r_2]$，然后回跳到 $[l_1, r_1]$，接下来一直向右构造到底。答案就是

$$r_1 - l_1 + 1 + \left\lfloor \frac{n - r_1}{2} \right\rfloor$$

还有一个问题：怎么确保能从空串一直构造到 $[l_2, r_2]$？

可以从 $[l_2, r_2]$ 开始，向左反向构造，到了 $[l_1, r_1]$ 内就跳回到 $[l_2, r_2]$ 中，再继续向左反向构造。

需要 SAM。对于一个节点（一个 $endpos$ 等价类），你需要知道他表示的最长串出现了多少次（这需要自底向上 dp 累加）、第一个 $endpos$（这需要自底向上 dp 取 min）、这个等价类中最长的串长（也就是 len），然后按上式算答案取最大值即可。

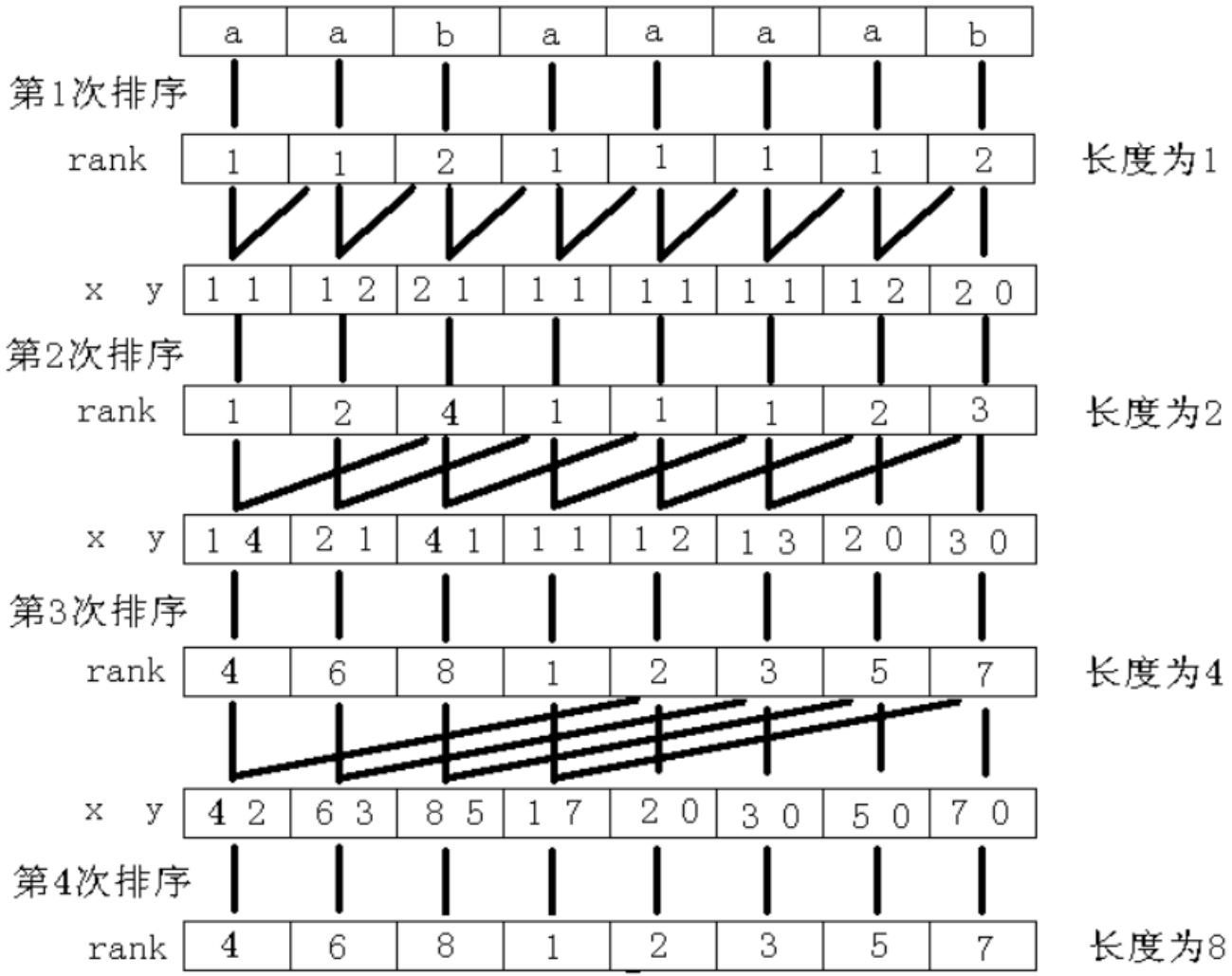
问题引入

给定字符串 S，$|S| = n$，求 S 排名为 1,2,...,n 的后缀的起始位置。

基本数组

- `sa[i]`：排名为 i 的后缀的起始位置
- `rk[i]`：起始位置为 i 的后缀的排名

倍增法


a a b a a a a b
第1次排序
rank 1 1 2 1 1 1 1 2
长度为1
x y 1 1 1 2 2 1 1 1 1 1 1 2 2 0
第2次排序
rank 1 2 4 1 1 1 2 3
长度为2
x y 1 4 2 1 4 1 1 1 1 2 1 3 2 0 3 0
第3次排序
rank 4 6 8 1 2 3 5 7
长度为4
x y 4 2 6 3 8 5 1 7 2 0 3 0 5 0 7 0
第4次排序
rank 4 6 8 1 2 3 5 7
长度为8

倍增法

- rk[i] 的初值是s[i]

倍增法

- rk[i] 的初值是s[i]
- 对 (rk[i],rk[i+1]) 排序，可以得到从每个 i 开始的长为 2 的子串（长度不足 2 的末尾补空字符）的排名，将结果更新到rk[i]

倍增法

- rk[i] 的初值是s[i]
- 对 (rk[i],rk[i+1]) 排序，可以得到从每个 i 开始的长为 2 的子串（长度不足 2 的末尾补空字符）的排名，将结果更新到rk[i]
- 对 (rk[i],rk[i+2]) 排序，可以得到从每个 i 开始的长为 4 的子串（长度不足 4 的末尾补空字符）的排名，将结果更新到rk[i]
- ……

倍增法

- rk[i] 的初值是s[i]
- 对 (rk[i],rk[i+1]) 排序，可以得到从每个 i 开始的长为 2 的子串（长度不足 2 的末尾补空字符）的排名，将结果更新到rk[i]
- 对 (rk[i],rk[i+2]) 排序，可以得到从每个 i 开始的长为 4 的子串（长度不足 4 的末尾补空字符）的排名，将结果更新到rk[i]
- ……

如果每一步的排序用sort，就得到了 $O(n \log^2 n)$ 的做法。

朴素倍增法

```
1  for (i = 1; i <= n; ++i) sa[i] = i, rk[i] = s[i];
2  for (w = 1; w < n; w <<= 1) {
3    sort(sa + 1, sa + n + 1, [](int x, int y) {
4      return rk[x] == rk[y] ? rk[x + w] < rk[y + w] : rk[x] < rk[y];
5    });  // 这里用到了 lambda
6    memcpy(oldrk, rk, sizeof(rk));
7    // 由于计算 rk 的时候原来的 rk 会被覆盖，要先复制一份
8    for (p = 0, i = 1; i <= n; ++i) {
9      if (oldrk[sa[i]] == oldrk[sa[i - 1]] &&
10         oldrk[sa[i] + w] == oldrk[sa[i - 1] + w]) {
11       rk[sa[i]] = p;
12     } else {
13       rk[sa[i]] = ++p;
14     }  // 若两个子串相同，它们对应的 rk 也需要相同，所以要去重
15   }
16 }
```

双关键字基数排序

注意到对 (rk[i],rk[i+w]) 排序时，关键字的值域是 $0, ..., n$，所以可以用基数排序来做这个双关键字排序。(基数排序是对相同关键字是保原序的，所以先排第二关键字，然后再排第一关键字)

```
1  // 对第二关键字：id[i] + w进行计数排序
2  memset(cnt, 0, sizeof(cnt));
3  memcpy(id + 1, sa + 1, n * sizeof(int));
4  for (i = 1; i <= n; ++i) ++cnt[rk[id[i] + w]];
5  for (i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
6  for (i = n; i >= 1; --i) sa[cnt[rk[id[i] + w]]--] = id[i];
7
8  // 对第一关键字：id[i]进行计数排序
9  memset(cnt, 0, sizeof(cnt));
10 memcpy(id + 1, sa + 1, n * sizeof(int));
11 for (i = 1; i <= n; ++i) ++cnt[rk[id[i]]];
12 for (i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
13 for (i = n; i >= 1; --i) sa[cnt[rk[id[i]]]--] = id[i];
```

常数优化

上面的代码复杂度为 $O(n \log n)$，常数大约为 7，可以进一步把常数优化到 5。第二关键字排序的实质，其实就是把超出字符串范围（即 sa[i]+w>n）的 sa[i] 放到 sa 数组头部，然后把剩下的依原顺序放入，所以第二关键字的排序可以写为：

```
1  for (p = 0, i = n; i > n - w; --i) id[++p] = i;
2  for (i = 1; i <= n; ++i) {
3      if (sa[i] > w) id[++p] = sa[i] - w;
4  }
```

完整代码

```
m = 130;
for(int i=1;i<=n;i++) c[rk[i]=s[i]]++;
for(int i=1;i<m;i++) c[i+1]+=c[i];
for(int i=n;i>=1;i--) sa[c[rk[i]]--]=i;
for(int k=1;k<=n;k<<=1)
{
    int num=0;
    for(int i=n-k+1;i<=n;i++) y[++num]=i;
    for(int i=1;i<=n;i++) if(sa[i]>k) y[++num]=sa[i]-k;
    for(int i=1;i<=m;i++) c[i]=0;
    for(int i=1;i<=n;i++) tmp[i]=rk[y[i]];
    for(int i=1;i<=n;i++) c[tmp[i]]++;
    for(int i=1;i<m;i++) c[i+1]+=c[i];
    for(int i=n;i>=1;i--) sa[c[tmp[i]]--]=y[i];
    num=1;swap(rk,y);rk[sa[1]]=1;
    for(int i=2;i<=n;i++)
        rk[sa[i]]=(y[sa[i]]!=y[sa[i-1]]||y[sa[i]+k]!=y[sa[i-1]+k])?++num:num;
    if(num==n) break;
    m=num;
}
```

例题选讲：「USACO07DEC」Best Cow Line

给你一个字符串，长度不超过 5×10^5，每次从首或尾取一个字符组成字符串，问所有能够组成的字符串中字典序最小的一个。

例题选讲：「USACO07DEC」Best Cow Line

给你一个字符串，长度不超过 5×10^5，每次从首或尾取一个字符组成字符串，问所有能够组成的字符串中字典序最小的一个。

【题解】枚举选头还是选尾，把反串接在正串后面，中间用一个 ASCII 码很大的分隔符隔开，借助后缀数组 $O(1)$ 判断即可。

height 数组

要想让 SA 变得更有用，我们还需要 height 数组：
`height[i]=LCP(sa[i],sa[i-1])`，特别地，`height[1]=0`

符号`LCP(i,j)`表示字符串 S 从 i 开始的后缀与从 j 开始的后缀的最长公共前缀长度

构建 height 数组

我们有性质：`height[rk[i]] >= height[rk[i-1]]-1`

构建 height 数组

我们有性质：height[rk[i]] >= height[rk[i-1]]-1

暴力求即可。

```
1 for (i = 1, k = 0; i <= n; ++i) {
2     if (rk[i] == 0) continue;
3     if (k) --k;
4     while (s[i + k] == s[sa[rk[i] - 1] + k]) ++k;
5     height[rk[i]] = k;
6 }
```

构建 height 数组

证明性质：`height[rk[i]] >= height[rk[i-1]]-1`

构建 height 数组

证明性质：height[rk[i]] >= height[rk[i-1]]-1

【证】当 height[rk[i-1]] <= 1 时，显然成立，现在设height[rk[i-1]] > 1。
由定义，height[rk[i-1]]=LCP(i-1,sa[rk[i-1]-1])，记：[i-1,n]=aAD，[sa[rk[i-1]-1],n]=aAB，其中a 是字母，A,B,D 都是字符串，从排名来看应该有B<D，另外：
[i,n]=AD，[sa[rk[i-1]-1]+1,n]=AB
由定义，height[rk[i]]=LCP(i,sa[rk[i]-1])，因此根据排名，有
AB <= [sa[rk[i]-1],n] <= AD
所以[sa[rk[i]-1],n]=AC，因此height[rk[i]] >= A||，证毕。

例题选讲

给定一个字符串，求两个子串[a,m] 和[b,p] 的最长公共前缀，多组询问。

例题选讲

给定一个字符串，求两个子串[a,m] 和[b,p] 的最长公共前缀，多组询问。

【题解】性质：
LCP(sa[i],sa[j])=min(height[i+1],height[i+2],...,height[j])

例题选讲

给定一个字符串，求两个子串[a,m] 和[b,p] 的最长公共前缀，多组询问。

【题解】性质：
LCP(sa[i],sa[j])=min(height[i+1],height[i+2],...,height[j])

令i=rk[a]，j=rk[b]，对height 数组做 RMQ 即可。
若LCP(sa[i],sa[j]) 超过了min(m-a,p-b)+1，那么答案应该是后者。

例题选讲：「USACO06DEC」Milk Patterns

给定字符串 S，求出现至少 k 次的子串的最大长度

例题选讲：「USACO06DEC」Milk Patterns

给定字符串 S，求出现至少 k 次的子串的最大长度

【题解】出现至少 k 次意味着后缀排序后有至少连续 k 个后缀以这个子串作为公共前缀。

例题选讲：「USACO06DEC」Milk Patterns

给定字符串 S，求出现至少 k 次的子串的最大长度

【题解】出现至少 k 次意味着后缀排序后有至少连续 k 个后缀以这个子串作为公共前缀。

所以，求出每相邻 $k-1$ 个 $height$ 的最小值，再求这些最小值的最大值就是答案。

例题选讲：「USACO06DEC」Milk Patterns

给定字符串 S，求出现至少 k 次的子串的最大长度

【题解】出现至少 k 次意味着后缀排序后有至少连续 k 个后缀以这个子串作为公共前缀。

所以，求出每相邻 $k-1$ 个 $height$ 的最小值，再求这些最小值的最大值就是答案。

做滑动窗口即可，可以单调队列，也可以 set

例题选讲：Distinct Substrings

给定一个字符串，求该字符串含有的本质不同的子串数量

例题选讲：Distinct Substrings

给定一个字符串，求该字符串含有的本质不同的子串数量

【题解】子串就是后缀的前缀，所以可以枚举每个后缀，计算前缀总数，再减掉重复。前缀总数其实就是子串个数，为 $n(n+1)/2$。

例题选讲：Distinct Substrings

给定一个字符串，求该字符串含有的本质不同的子串数量

【题解】子串就是后缀的前缀，所以可以枚举每个后缀，计算前缀总数，再减掉重复。前缀总数其实就是子串个数，为 $n(n+1)/2$。

如果按后缀排序的顺序枚举后缀，每次新增的子串就是除了与上一个后缀的 LCP 剩下的前缀。这些前缀一定是新增的，且只有这些前缀是新增的。

例题选讲：Distinct Substrings

给定一个字符串，求该字符串含有的本质不同的子串数量

【题解】子串就是后缀的前缀，所以可以枚举每个后缀，计算前缀总数，再减掉重复。前缀总数其实就是子串个数，为 $n(n+1)/2$。

如果按后缀排序的顺序枚举后缀，每次新增的子串就是除了与上一个后缀的 LCP 剩下的前缀。这些前缀一定是新增的，且只有这些前缀是新增的。

所以答案为：

$$\frac{n(n-1)}{2} - \sum_{i=2}^{n} height[i]$$

例题选讲：Musical Themes

给定一个字符串，求长度至少为 4 且不重叠出现至少两次的最长子串

字符集 88，串长 10^5（洛谷那个题只有 5000，我们加强一下）

本题历史地位：楼教主男人八题、2009 年集训队论文例题

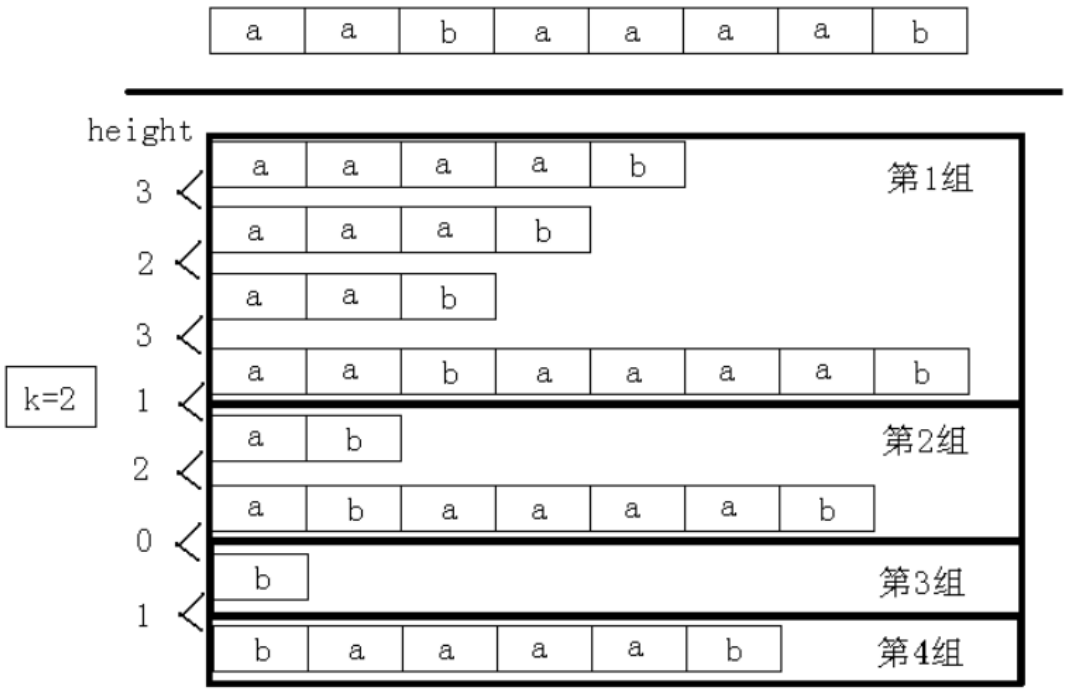
例题选讲：Musical Themes

【题解】二分答案，转化为判定是否存在长度至少为 4 且不重叠出现至少两次的长为 k 的子串。

例题选讲：Musical Themes

【题解】二分答案，转化为判定是否存在长度至少为 4 且不重叠出现至少两次的长为 k 的子串。

对排好序的后缀分组，使得每一个组内所有后缀的最长公共前缀都不小于 k，借助 height 数组即可实现这个分组。例如字符串 “aabaaaab” 在 $k = 2$ 时的后缀分组：



例题选讲：Musical Themes

显然，有希望成为最长公共前缀不小于 k 的两个后缀一定在同一组，因此为了让他们不重叠，求出每一组 sa 值的最大值和最小值，看相差是否不小于 k 即可。只要存在一组相差不小于 k 则可行，否则不可行。

例题选讲：[TJOI2016] 字符串

给一个长为 n 只有小写字母的串 s，有 m 组询问，每次给 a, b, c, d，问 $s[a...b]$ 的所有子串与 $s[c...d]$ 的最长公共前缀的最大值是多少。

例题选讲：[TJOI2016] 字符串

【题解】二分答案 k，考虑判断是否存在一个起始位置 $i \in [a, b]$，使得 $s[i...b]$ 与 $s[c...d]$ 的最长公共前缀至少为 k。

例题选讲：[TJOI2016] 字符串

【题解】二分答案 k，考虑判断是否存在一个起始位置 $i \in [a, b]$，使得 $s[i...b]$ 与 $s[c...d]$ 的最长公共前缀至少为 k。

根据 height 数组的性质，$s[i...n]$ 与 $s[c...n]$ 的最长公共前缀至少为 k，当且仅当`height[rk[i]...rk[c]]` 的值都至少为 k。

例题选讲：[TJOI2016] 字符串

【题解】二分答案 k，考虑判断是否存在一个起始位置 $i \in [a, b]$，使得 $s[i...b]$ 与 $s[c...d]$ 的最长公共前缀至少为 k。

根据 height 数组的性质，$s[i...n]$ 与 $s[c...n]$ 的最长公共前缀至少为 k，当且仅当`height[rk[i]...rk[c]]` 的值都至少为 k。

我们二分出最大的包含`rk[c]` 的区间`[l,r]`，使得`height[l+1...r]` 的值都至少为 k，二分的判定需要 RMQ。

例题选讲：[TJOI2016] 字符串

【题解】二分答案 k，考虑判断是否存在一个起始位置 $i \in [a, b]$，使得 $s[i...b]$ 与 $s[c...d]$ 的最长公共前缀至少为 k。

根据 height 数组的性质，$s[i...n]$ 与 $s[c...n]$ 的最长公共前缀至少为 k，当且仅当`height[rk[i]...rk[c]]` 的值都至少为 k。

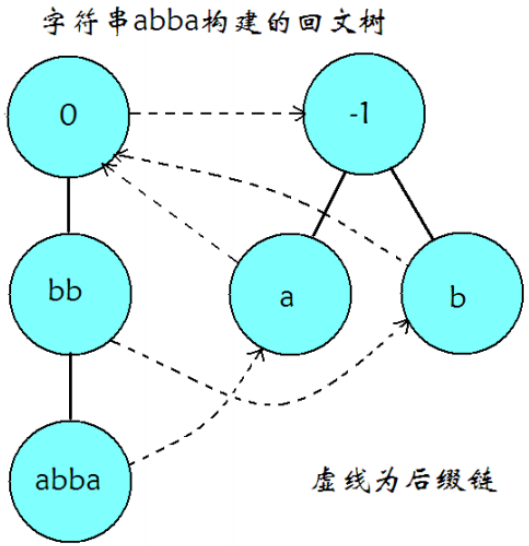
我们二分出最大的包含`rk[c]` 的区间`[l,r]`，使得`height[l+1...r]` 的值都至少为 k，二分的判定需要 RMQ。

然后判断是否存在 $i \in [l, r]$，满足 $sa[i] \in [a, b - k + 1]$ 即可，这一步需要可持久化线段树。

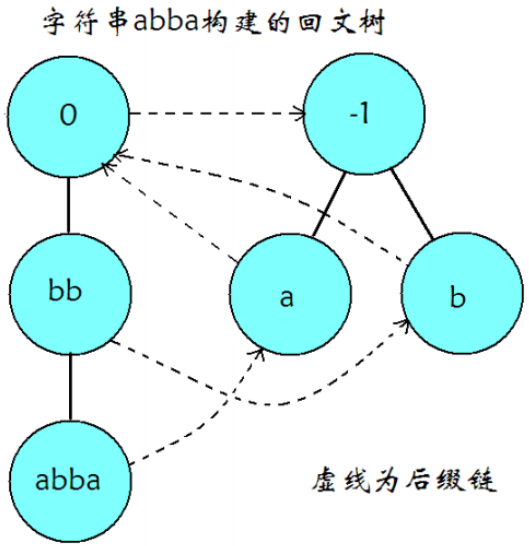
问题引入

给定一个长度不超过 10^5 的字符串，求本质不同的回文子串个数。

什么是回文自动机（回文树）

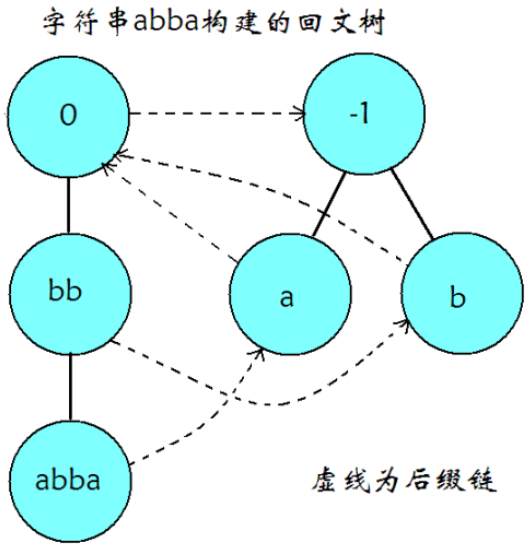


什么是回文自动机（回文树）



PAM 包含两棵树，一棵树中的节点对应的回文子串长度均为奇数，另一棵树中的节点对应的回文子串长度均为偶数。

什么是回文自动机（回文树）



PAM 包含两棵树，一棵树中的节点对应的回文子串长度均为奇数，另一棵树中的节点对应的回文子串长度均为偶数。
和其它的自动机一样，一个节点的 fail 指针指向的是这个节点所代表的回文串的最长回文后缀所对应的节点，但是转移边并非代表在原节点代表的回文串后加一个字符，而是表示在原节点代表的回文串前后各加一个相同的字符

建树

回文树有两个初始状态，分别代表长度为 -1,0 的回文串。我们可以称它们为奇根，偶根。它们不表示任何实际的字符串，仅作为初始状态存在。

建树

回文树有两个初始状态，分别代表长度为 -1,0 的回文串。我们可以称它们为奇根，偶根。它们不表示任何实际的字符串，仅作为初始状态存在。

偶根的 fail 指针指向奇根，而我们并不关心奇根的 fail 指针，因为奇根不可能失配（奇根转移出的下一个状态长度为 1，即单个字符。一定是回文子串）

建树

回文树有两个初始状态，分别代表长度为 -1,0 的回文串。我们可以称它们为奇根，偶根。它们不表示任何实际的字符串，仅作为初始状态存在。

偶根的 fail 指针指向奇根，而我们并不关心奇根的 fail 指针，因为奇根不可能失配（奇根转移出的下一个状态长度为 1，即单个字符。一定是回文子串）

类似 SAM，我们一个一个字符添加，现在向自动机中添加`S[k]`。

建树

回文树有两个初始状态，分别代表长度为 -1,0 的回文串。我们可以称它们为奇根，偶根。它们不表示任何实际的字符串，仅作为初始状态存在。

偶根的 fail 指针指向奇根，而我们并不关心奇根的 fail 指针，因为奇根不可能失配（奇根转移出的下一个状态长度为 1，即单个字符。一定是回文子串）

类似 SAM，我们一个一个字符添加，现在向自动机中添加`S[k]`。

我们从以上一个字符结尾的最长回文子串对应的节点开始，不断沿着 fail 指针走，直到找到一个节点`x` 满足 `S[k]==S[k-len[x]-1]`，即满足此节点所对应回文子串的上一个字符与待添加字符相同。

建树

现在，如果节点x 通过字母S[k] 的转移边已经被占用，我们无需做任何事情。否则我们要建一个新点q。

建树

现在，如果节点x 通过字母S[k] 的转移边已经被占用，我们无需做任何事情。否则我们要建一个新点q。

显然新点的长度是len[x]+2，它的 fail 指针？

建树

现在，如果节点x 通过字母S[k] 的转移边已经被占用，我们无需做任何事情。否则我们要建一个新点q。

显然新点的长度是len[x]+2，它的 fail 指针？

找到 x 在 fail 树上的某个尽量近的祖先 y，使这个祖先前后加上S[k] 后能构成新点的回文后缀，也就是满足 S[k-len[y]-1]==S[k]，那么令 fail[q]=ch[y][S[k]] 即可。

代码

```
1  int newnode(int x){len[++tot]=x;return tot;}
2  void pam_init(){tot=-1;newnode(0);newnode(-1);fail[0]=1;s[0]=-1;}
3  int find(int p){while(s[n]!=s[n-len[p]-1]) p=fail[p];return p;}
4  void insert(int c)
5  {
6      s[++n]=c;
7      int p=find(lst);
8      if(!ch[p][c])
9      {
10         int q=newnode(len[p]+2);
11         fail[q]=ch[find(fail[p])][c];
12         ch[p][c]=q;
13     }
14     lst=ch[p][c];
15 }
```

这是 PAM 的核心代码，很短。

例题选讲：洛谷模板题

给定一个字符串 S，保证每个字符为小写字母。对于 S 的每个位置，请求出以该位置结尾的回文子串个数。

例题选讲：洛谷模板题

给定一个字符串 S，保证每个字符为小写字母。对于 S 的每个位置，请求出以该位置结尾的回文子串个数。

【题解】建 PAM，维护节点深度，边建边输出当前点深度即可。

例题选讲：[APIO2014] 回文串

给你一个由小写拉丁字母组成的字符串 S。我们定义 S 的一个子串的存在值为这个子串在 S 中出现的次数乘以这个子串的长度。

对于给你的这个字符串 S，求所有回文子串中的最大存在值

例题选讲：[APIO2014] 回文串

给你一个由小写拉丁字母组成的字符串 S。我们定义 S 的一个子串的存在值为这个子串在 S 中出现的次数乘以这个子串的长度。

对于给你的这个字符串 S，求所有回文子串中的最大存在值

【题解】建 PAM，如何求 PAM 中的节点所代表的回文串出现的次数？

例题选讲：[APIO2014] 回文串

给你一个由小写拉丁字母组成的字符串 S。我们定义 S 的一个子串的存在值为这个子串在 S 中出现的次数乘以这个子串的长度。

对于给你的这个字符串 S，求所有回文子串中的最大存在值

【题解】建 PAM，如何求 PAM 中的节点所代表的回文串出现的次数？

每次插入完打一个 $+1$ 标记，然后在 fail 树上自底向上累加即可。

例题选讲：[APIO2014] 回文串

给你一个由小写拉丁字母组成的字符串 S。我们定义 S 的一个子串的存在值为这个子串在 S 中出现的次数乘以这个子串的长度。

对于给你的这个字符串 S，求所有回文子串中的最大存在值

【题解】建 PAM，如何求 PAM 中的节点所代表的回文串出现的次数？

每次插入完打一个 $+1$ 标记，然后在 fail 树上自底向上累加即可。

题外话：PAM 是 2015 年的科技，这是 2014 年的题，std 的做法是 manacher+SAM（或者 manacher+SA），有兴趣可以看看

例题选讲：CF835D

给定一个只包含小写字母的字符串 S，对 $k = 1, ..., |S|$，求有多少个 k 阶回文子串。一个串 T 是 k 阶回文串，需要满足：

- T 是回文串
- T 的左半部分是 $k-1$ 阶回文串（若 $|T|$ 为奇数，则左半部分不包括中间）

原题数据范围只有 5000，我们可以做的数据范围是 10^6

例题选讲：CF835D

【题解】在 PAM 上维护一个g[x]，指向x 的长度不超过len[x]/2 的最长回文后缀，维护方式类似 fail：

```
if(len[now]<=2) g[now]=fail[now];
else {
    int p=g[cur];
    while(s[n]!=s[n-len[p]-1]||(len[p]+2)*2>len[now]) p=fail[p];
    g[now]=ch[p][c];
}
```

例题选讲：CF835D

【题解】在 PAM 上维护一个g[x]，指向x 的长度不超过len[x]/2 的最长回文后缀，维护方式类似 fail：

```
if(len[now]<=2) g[now]=fail[now];
else {
    int p=g[cur];
    while(s[n]!=s[n-len[p]-1]||(len[p]+2)*2>len[now]) p=fail[p];
    g[now]=ch[p][c];
}
```

接下来按转移图拓扑序 dp 即可（其实按点从小到大的顺序就行），设f[x] 表示节点x 最多是几阶回文，转移：f[x]=f[g[x]]+1，然后统计答案输出即可。

例题选讲：[CERC2014] Virus synthesis

初始有一个空串，你每次可以在下面选一种操作执行：

- 串开头或末尾加一个字符
- 串开头或末尾加一个该串的逆串

求构造 S 所需的最小操作次数，$|S| \leq 10^5$，字符集为{A,T,C,G}

例题选讲：[CERC2014] Virus synthesis

【题解】建 PAM，设f[x]表示构造节点x 的串、且最后一步为复制，所需的最少操作数。（对于奇串无意义，因此为了方便，设奇串的f[x]=len[x]）

例题选讲：[CERC2014] Virus synthesis

【题解】建 PAM，设f[x] 表示构造节点x 的串、且最后一步为复制，所需的最少操作数。（对于奇串无意义，因此为了方便，设奇串的f[x]=len[x]）

若偶串z（非空）可以通过加一个字母转移到x，那么显然可以在z 复制之前添加一个字母，从而通过f[z]+1 次操作构造x。因此为了方便，空偶串的f 值设为 1

例题选讲：[CERC2014] Virus synthesis

【题解】建 PAM，设f[x] 表示构造节点x 的串、且最后一步为复制，所需的最少操作数。（对于奇串无意义，因此为了方便，设奇串的f[x]=len[x]）

若偶串z（非空）可以通过加一个字母转移到x，那么显然可以在z 复制之前添加一个字母，从而通过f[z]+1 次操作构造x。因此为了方便，空偶串的f 值设为 1

我们也可以找到长度不超过len[x]/2 的最长回文后缀，和上一题一样，就是g[x]，然后通过g[x] 先填到x 的一半，然后做一次复制，操作次数是f[g[x]]+(len[x]/2-len[g[x]])+1

例题选讲：[CERC2014] Virus synthesis

【题解】建 PAM，设f[x] 表示构造节点x 的串、且最后一步为复制，所需的最少操作数。（对于奇串无意义，因此为了方便，设奇串的f[x]=len[x]）

若偶串z（非空）可以通过加一个字母转移到x，那么显然可以在z 复制之前添加一个字母，从而通过f[z]+1 次操作构造x。因此为了方便，空偶串的f 值设为 1

我们也可以找到长度不超过len[x]/2 的最长回文后缀，和上一题一样，就是g[x]，然后通过g[x] 先填到x 的一半，然后做一次复制，操作次数是f[g[x]]+(len[x]/2-len[g[x]])+1

结合上面两种转移，答案就是n-len[x]+f[x] 的最小值。

[1] OIwiki, "后缀自动机."

[2] Pecco, "算法学习笔记 (85): 后缀自动机."

[3] cyendra, "后缀自动机（SAM）."

[4] OIwiki, "后缀数组."

[5] 罗穗骞, "后缀数组——处理字符串问题的有力工具," *IOI2009 国家集训队论文*, 2009.

[6] OIwiki, "回文树."

Thank You