easy字符串专题

Author：hly

Modifier：Cold_Chair

声明——

此ppt大量借鉴前人的表达及资料，“站在巨人的肩膀上”，仅做少量整理

Catalog

- SA 讲解
- SAM讲解
- Palindrome Tree讲解
- 省选及以下难度字符串杂（luo）题选讲

后缀数组 SA

- ~~这是某个，em，非常常用的算法，且较好理解，实现简单~~

- SA，是一种在字符串处理中非常优秀的数据结构，是一种**处理字符串问题的有力工具，**我们应该掌握好后缀数组这种数据结构，并且能在不同类型的题目中灵活、高效的运用

双关键字桶排

- 为了更好的理解SA的倍增建法，我调整了讲解顺序。
- 先看一道题：
  给出n个数a[1-n]，把它们从小到大排序。
- ~~不要问我怎么读入，问就交互。~~

- $1 <= n \leq 5e7$
- $0 <= a[i] < 2^{31}$

双关键字拆分

- 如果a[i]<=1e6，那么便是桶排裸题。
- 由这个受到启发，对于一个较大的数，可以把它拆成两个比较小的数：
- 设$m = \sqrt{maxa}$
- a1$[i] = \left\lfloor \frac{a[i]}{m} \right\rfloor, a2[i] = a[i] - a1[i] * m$
  即$a[i] = a1[i] * m + a2[i]$
- 这样的话有$a1[i], a2[i] <= m$
- 对a的排序可以看作以a1为第一关键字，a2为第二关键字的排序。

正常桶排

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N = 1e7 + 5;

const int m = 1 << 16;

int n, a[N], b[N];
int c[m];

int main() {
    scanf("%d", &n);
    for(int i = 1; i <= n; i ++) {
        scanf("%d", &a[i]);
        c[a[i]] ++;
    }
    for(int i = 1; i < m; i ++) c[i] += c[i - 1];
    for(int i = 1; i <= n; i ++) b[c[a[i]] --] = a[i];
    for(int i = 1; i <= n; i ++) printf("%d ", b[i]);
}
```

双关键字的

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N = 1e7 + 5;

const int m = 1 << 16;

int n, a[N], b[N];
int c1[m], c2[m];

int main() {
    scanf("%d", &n);
    for(int i = 1; i <= n; i ++) {
        scanf("%d", &a[i]);
        c1[a[i] % m] ++, c2[a[i] / m] ++;
    }
    for(int i = 1; i < m; i ++) c1[i] += c1[i - 1], c2[i] += c2[i - 1];
    for(int i = n; i >= 1; i --) b[c1[a[i] % m] --] = a[i]; //注意循环顺序是倒着的
    for(int i = n; i >= 1; i --) a[c2[b[i] / m] --] = b[i];
    for(int i = 1; i <= n; i ++) printf("%d ", a[i]);
}
```

发现就是做个前缀和，然后倒着扫一遍就好了。

……

- 上面的写法并不是常数最小的，只是为了方便看懂。
- N关键字的做法也是一样的。
- 有兴趣卡常的同学可以去做WC2017挑战。

- 顾名思义，SA就是一个对后缀搞事情的数组
- 现在要给S的|S|个后缀排序。

- 定义：
- suf(i)：表示当前字符串S的子串s[i..|S|]，即S以i开始的后缀。
- 两个字符串的比较(……)
- 数组SA[]：它是|S|个后缀的一个排列，其中 suf(SA[1])<suf(SA[2])<...<suf(SA[|S|])
- 数组rank[]：rank[i]表示suf(i)的排名

- 简单来说，SA[i]表示“第i大的是谁”，rank[i]即“suf(i)排第几”

- 问题在于如何在合理的时间内给所有的后缀排序
- 有两种算法：倍增和DC3
- 只讲倍增（我也只会倍增）
- 以字符串“aabaaaab”为例。

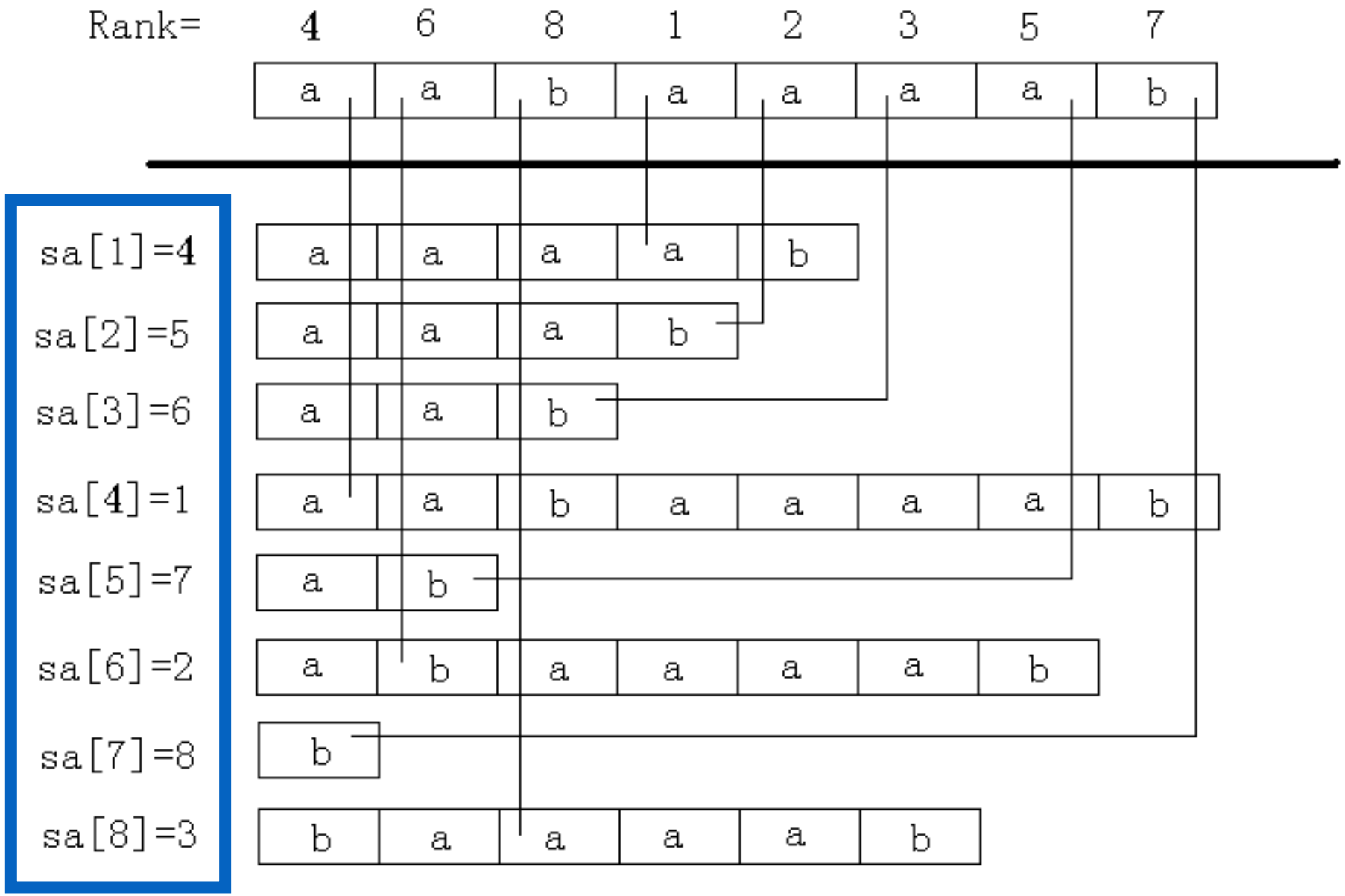

Rank= 4 6 8 1 2 3 5 7
a a b a a a a b
sa[1]=4
a a a a b
sa[2]=5
a a a b
sa[3]=6
a a b
sa[4]=1
a a b a a a a b
sa[5]=7
a b
sa[6]=2
a b a a a a b
sa[7]=8
b
sa[8]=3
b a a a a b

图1

- 倍增算法的主要思路：
- 用倍增的方法对每个字符开始的**长度为 2^k** 的子字符串进行排序，求出排名，即 rank 值。当2^k大于 n 以后，每个字符开始的长度为 2^k 的子字符串便相当于所有的后缀。并且这些子字符串都一定已经比较出大小，即 rank 值中没有相同的值，那么此时的 rank 值就是最后的结果。
- 每一次排序都利用上次长度为 2^{k-1}的字符串的 rank 值，那么长度为 2^k 的字符串就可以用两个长度为 2^{k-1}的字符串的排名作为关键字表示，然后进行基数排序，便得出了长度为 2^k的字符串的 rank 值。

以字符串“aabaaaab”为例，整个过程如图 2 所示。
其中 x、y 是表示长度为 2k的字符串的两个关键字 。

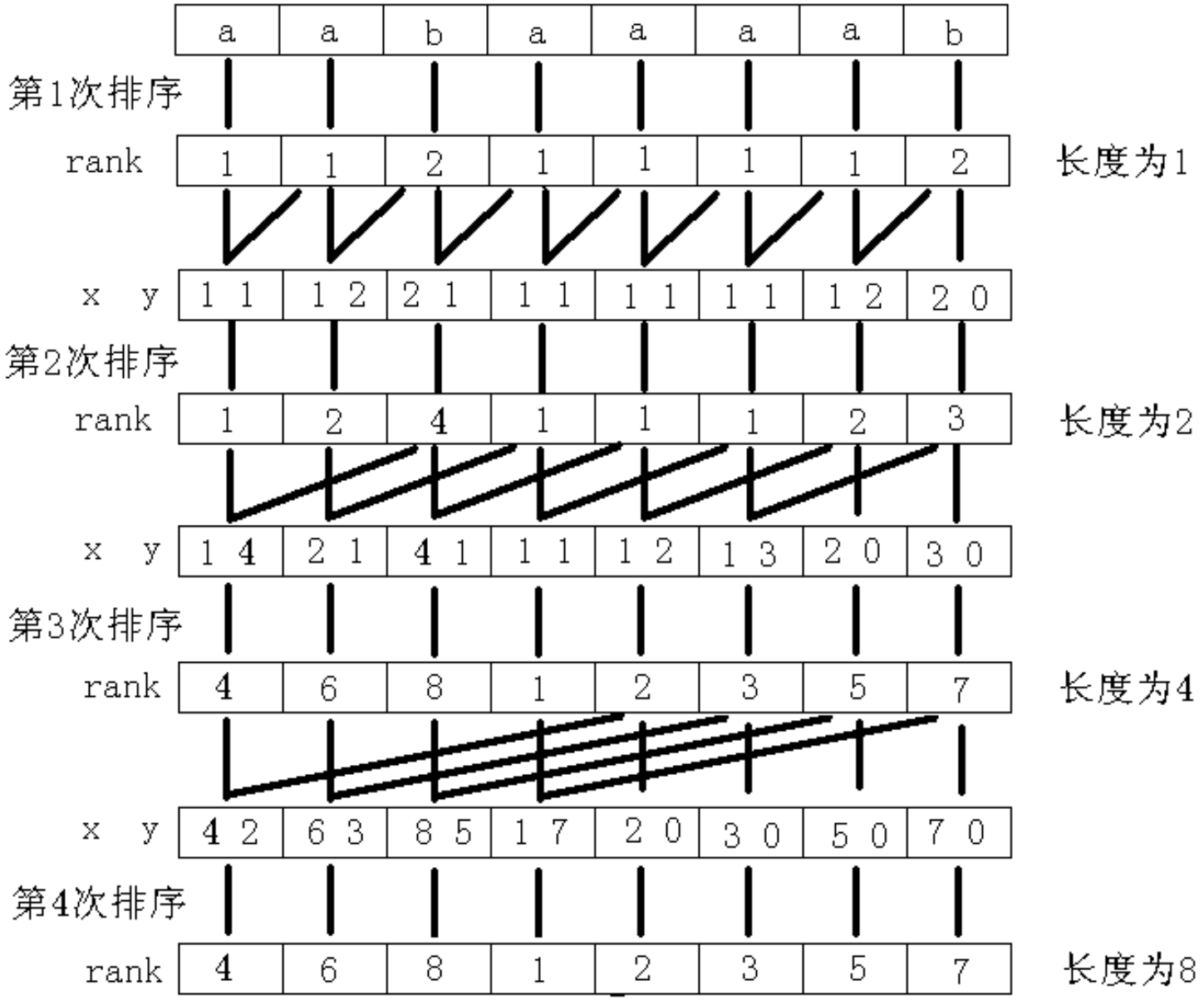


图2

- 具体来说就是先按第二关键字排序，如果没有第二关键字的（第二关键字超界的）就排在最前面

- 然后用一个桶排（排第一关键字），按照第二关键字的顺序从桶排的数组中取出，就可以得到按照两个关键字排序的结果了

```
11  //排序部分
12  void rsort(){
13      fo(i,0,m) sm[i]=0;
14      fo(i,1,n) sm[rank[i]]++;
15      fo(i,1,m) sm[i]+=sm[i-1];
16      fd(i,n,1) SA[sm[rank[sec[i]]]--]=sec[i];
17  }
18
```

height数组

- 定义height[i]为suf(SA[i-1])与suf(SA[i])的LCP长度。（height[1]=0）

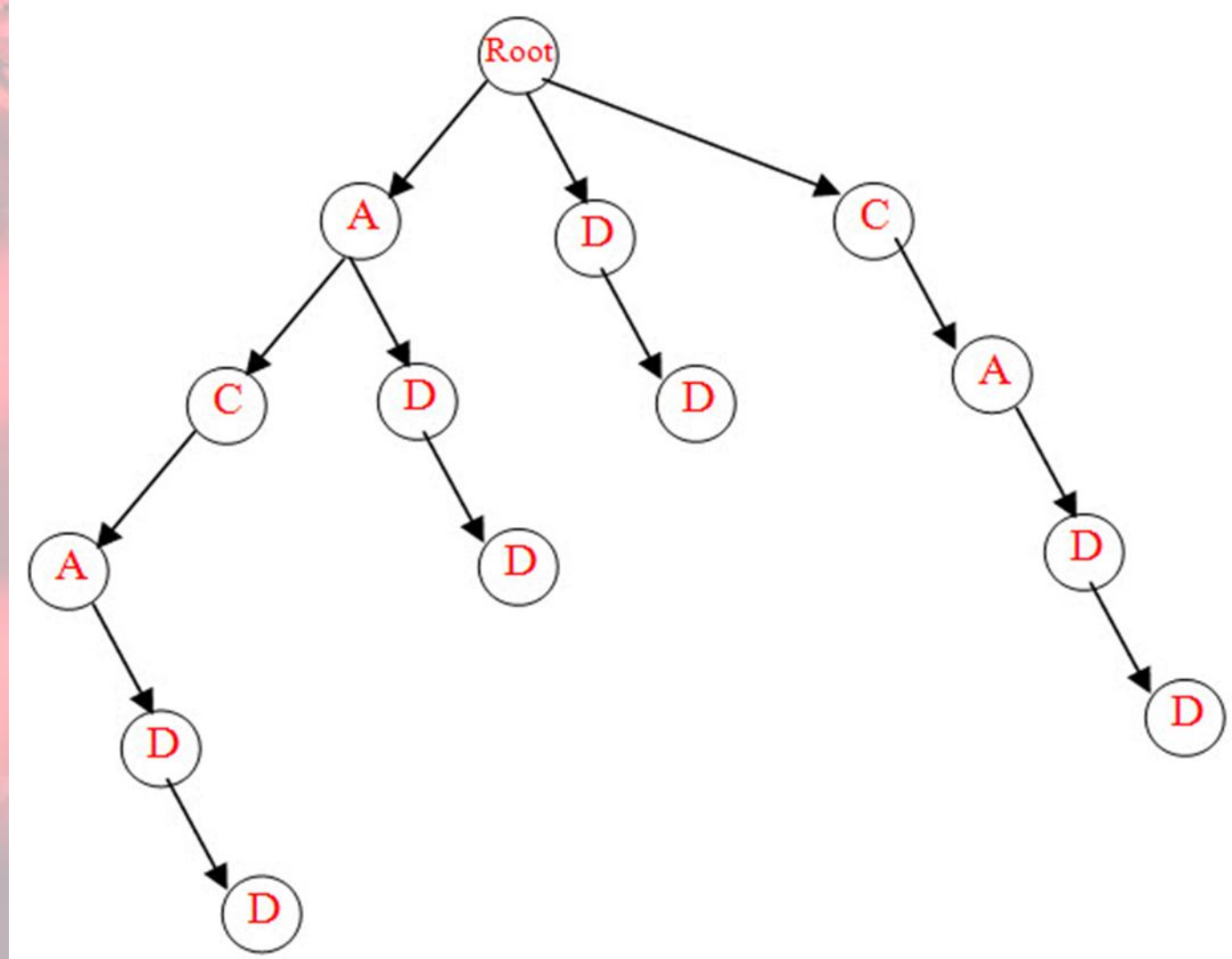
- 则有性质——
- suf(i) 和 suf(j) 的 LCP 为min(height[rank[k]]), i+1≤ k ≤ j

- 这里要利用一个结论：height[rank[i]]≥height[rank[i-1]]-1
  这样才能O(n)去预处理height数组。
- 下一页是证明

- 假设suf(i-1)在SA[]中前一个是suf(j)，它们的LCP长度height[rank[i-1]]=len，那么S[j..j+len-1]==S[i-1..i+len-2]
- 那么一定有S[j+1..j+len-1]==S[i..i+len-2]
- 这一段的长度len-1=height[rank[i-1]]-1
- 所以height[rank[i]]≥height[rank[i-1]]-1

后缀自动机SAM

- 后缀自动机实质上是用后缀搞事情, 以字符串ACADD为例:



- 这样很浪费空间和时间(实际上都是O(n^2)).但是,注意:这棵字母树的结点虽然多,但大部分结点都只有一个儿子,而且有很多段是一样的.那么,利用公共部分,就可以对空间进行压缩
- 大致做法:假设当前已经建好了s的某个前缀的后缀自动机t,那么就要通过某种算法,添加一个字符x,得到s另一前缀tx的后缀自动机,这样每次插入一个字符,最后把s的所有字符按顺序插入完毕就得到了s的后缀自动机.
- 这样的话,建造后缀自动机的过程是在线的,就是说,可以任意时刻询问s的某些信息,也可以任意时刻在s的结尾插入一些字符,变成新的字符串.不过,删除是不支持的.

- 定义:
- son[26]:该结点对应的子串加上某个字符后生成的合法子串在后缀自动机中所对应的位置
- pre:注意这不是返回它的父结点(因为某个点有可能成为多个结点的儿子),而是返回上一个可以接收后缀的结点(如果当前结点可以接收新的后缀,那么pre指向的结点也一定可以接收后缀).
- step:返回的是从根结点走到该结点,最多需要多少步.

- 提出三个后缀自动机的性质:

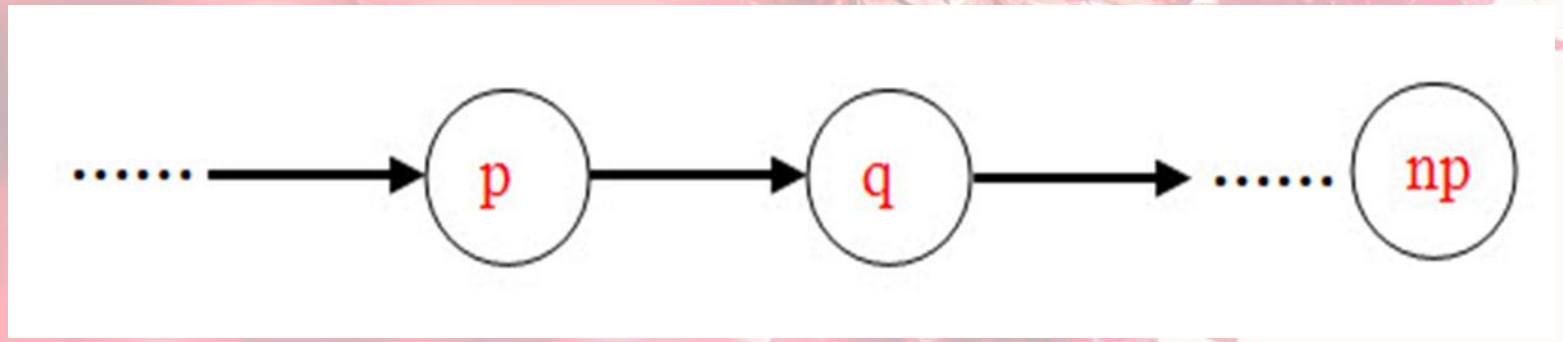
①从root到任意结点p的每条路径上的字符组成的字符串,都是当前串t的子串.

②因为满足性质一,所以如果当前结点p是可以接收新后缀的结点,那么从root到任意结点p的每条路径上的字符组成的字符串,都是必定是当前串t的后缀.

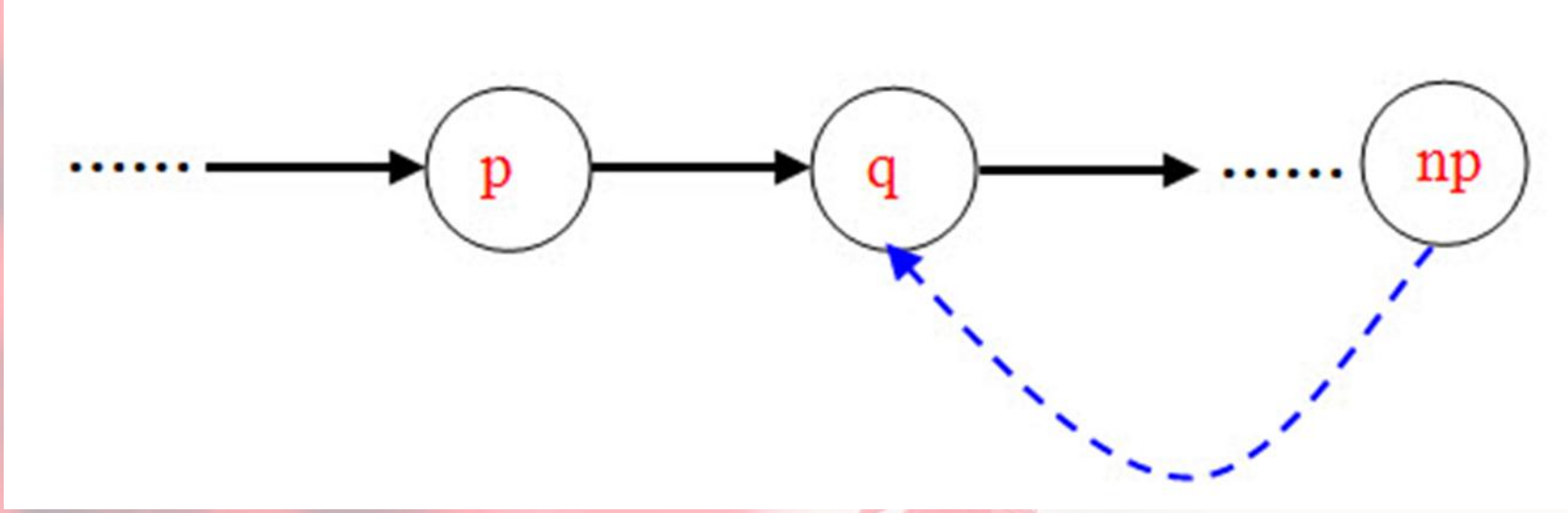
③如果结点p可以接收新的后缀,那么p的pre指向的结点也可以接收后缀，反过来不行

- 设插入字符x——
- 首先建立储存当前字符x的结点np,找到之前最后一个建立的结点(因为它一定满足性质②),然后就不断按pre指针跳(直到跳到有x儿子的结点为止).
- 假设当前跳到p结点,如果p没有x儿子,那么它一定可以接收新来的字符,然后就把p的x儿子赋值为np(注意这时,p接收了后缀字符x,目前已经不可以接收新的后缀字符了).然后,就要处理有x儿子的结点了,假设p的x儿子是q.

- 只有2种情况:
- ①step[q]=step[p]+1.
- 因为我们要后缀自动机的结点尽量少,所以要尽量共用一些信息.



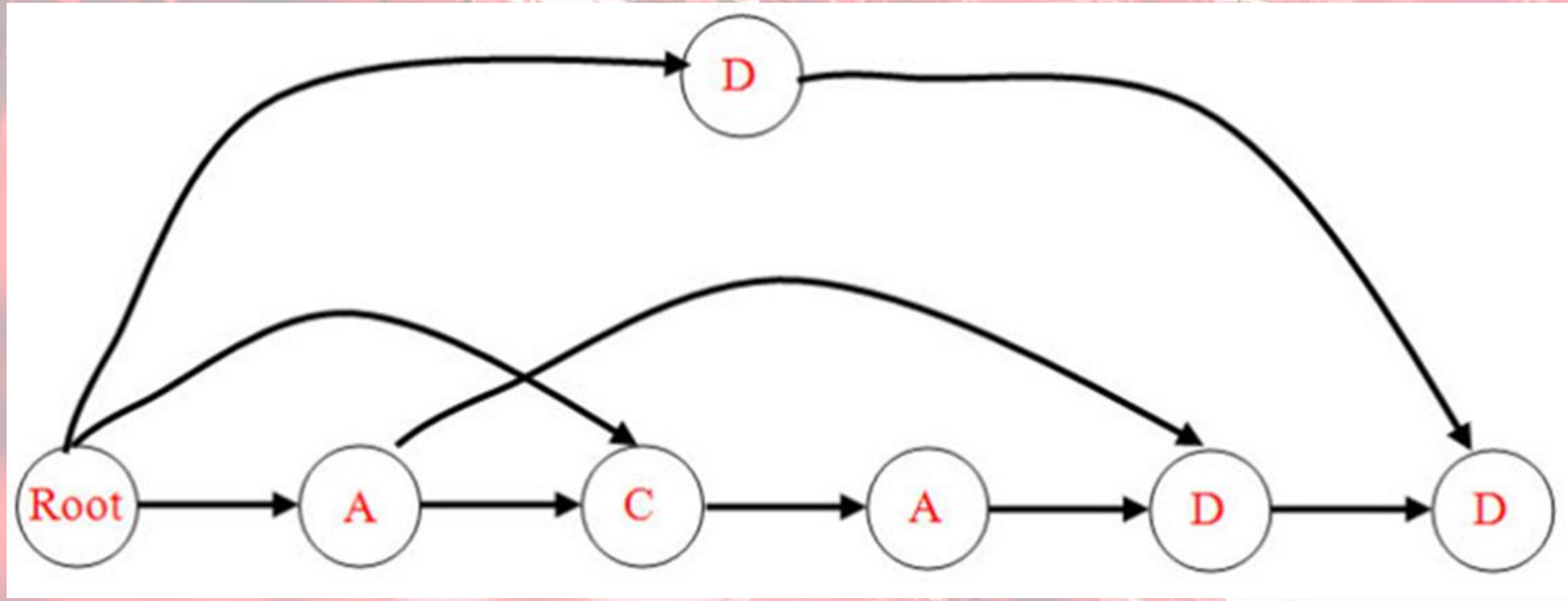
- step[q]=step[p]+1,保证了——
- q原本是从p的路径上来的,而且p和q之间不会夹杂其它字符.虽然q本来不一定可以接收新的后缀,但p可以接收后缀x,如果当前经过p来到q,就可以视为是在t的某个后缀后面插入了x(现在q就是那个x),并且在下一次插入的时候,q也可以接收后缀(因为它现在可以被视为x的结点了),所以就把np的pre指向q.



- ②step[q]>step[p]+1
- 这和上一种情况一样,也面临着q点是否可以当成x结点的问题.
- 在上一种情况的描述中,我们可以知道, step[q]=step[p]+1可以保证q原本是从p的路径上来的,而且p和q之间不会夹杂其它字符,所以可以直接使用q结点.
- 那么反过来, step[q]>step[p]+1,就说明p和q之间有可能会夹杂其它字符,这就不能保证利用q结点以后,到q的路径都是tx的后缀了,于是我们不能采取和前一种情况一样的做法.
- 但是,我们可以模仿前一种情况的做法.

- 其实很简单，就是把q拆成两个点。
- 新建一个结点nq来代替q的功能,同时使step[nq]=step[p]+1就相当于第一种情况了,这时,只要把q的son边和pre边都copy到nq上即可.但是别忘了把nq的pre改为p,再把nq和np的pre都改为nq.
- 因为现在nq代替了q,所以np的pre是nq.由性质③可知nq的pre只能是p.同样的,q和nq也满足性质③,所以q的pre只能是nq.
- 最后还要再按p的pre指针往上跳,把son[x]=q的p结点改为son[x]=nq(因为nq代替了原来的q).

最后变成这个样子



```
12      //如果需要就看一看
13      int push(int x){dep[++tot]=x;return tot;}
14      void ins(int c){
15          int np=push(dep[last]+1),p=last;
16          for(;p && !t[p][c];p=fa[p]) t[p][c]=np;
17          if(!p) fa[np]=1;
18          else{
19              int q=t[p][c];
20              if(dep[q]==dep[p]+1) fa[np]=q;
21              else{
22                  int nq=push(dep[p]+1);
23                  fa[nq]=fa[q],fa[q]=fa[np]=nq;
24                  memcpy(t[nq],t[q],sizeof t[q]);
25                  for(;t[p][c]==q;p=fa[p]) t[p][c]=nq;
26              }
27          }
28          last=np;
29      }
30
```

jzoj4072.【TJOI2015】弦论(string)

- 给定参数t, k和一个长度为n的字符串，对于给定的字符串，求其第k小子串
- 若t为0，则表示不同位置的相同子串算一个，t为1则表示不同位置的相同子串算多个

CF Compress String

- 给出长度为n的目标字符串s，现有两种生成方式：
- 在末尾加入一个字符，代价a
- 在末尾加入已输入字符串的一个子串，代价b

- n, a, b $\leq$ 5000

Jzoj4039 诸神眷顾的幻想乡

- 有一棵有n个节点的树，每个节点上有一个数字。
  求所有树的路径中，所形成的字符串中不同的有多少个。

- 1<=n<=10^5
- 叶子节点数小于20

Palindrome Tree

- 一个串s的回文树为一个森林，由两棵树组成。

- 设两棵树的根分别为even,odd。树上每个节点对应着一个字符串，除根外与s的回文子串一一对应。树上每条边都有对应的一个字符，且满足一个点的所有出边对应的字符各不相同。
- 对于树上一个点i，令fa_i为其父亲，则i对应的字符串s_i为fa_i 对应的字符串s_{fa_i}在两端加上连接i与fa_i的边对应的字符c， 即$s_i = cs_{fa_i}c$。

- 特别的，对于根even，令其对应的字符串为空串，根odd 对应的字符串为一 长度为-1 的实际并不存在的字符串，odd的儿子对应的字符串为连出边对应的字符。
- 此外，对于回文树上一点i，定义其失配指针$fail_i$，指向i的最长回文后缀在回文树上对 应的节点。特别的，定义$fail_{even}$与$fail_{odd}$均为odd。
- 由fail指针可以定义一个字符串s的fail树为以fail指针为连边所生成的树。对于fail树 上一点i，定义其fail链为一个集合，等于$fail_i$的fail链并上i。特别的，定义odd 的fail链为其本身

定理: 对于一个字符串s，不同的回文子串个数最多只有|s|个。

- 证明: 使用数学归纳法。
- 当|s| = 1时，只有s[1..1]一个子串，并且他是回文的，所以结论成立。
- 当|s| > 1时，设s = s'c，其中c为s的最后一个字符，并且结论对s'成立。考虑以末尾字符c为结尾的回文子串，假设他们的左端点从左到右依次为$l_1, l_2, \cdots, l_k$，那么由 于s[l_1..|s|] 为回文串，那么对于所有的位置$l_1 \le p \le |s|$，都会有s[p..|s|] = s[l_1.. l_1 +|s|− p]，所以对于回文子串s[l_i..|s|]，都会有 s[li..|s|] = s[l_1.. l_1 +|s|− l_i]，当i，1 时，总会有 l_1 +|s|− l_i <|s|，从而 s[l_i ..|s|]已经在s[1..|s|−1]中出现，因此每次不同的回文串最多新 增一个，即s[l_1..|s|]。因此结论对于s依然成立。

- 因此回文树的状态是O(|s|)级别的。考虑转移，一个点的转移数是O(Σ)的，但对于一个 状态，能转移到他的节点其实是唯一的，所以总转移数是O(|s|)的。一个节点只有一个失配 指针，因此回文树的总节点数与总转移数都是O(|s|)的。

- 然后就很简单啦
- 类似的，在s的最长回文后缀对应的fail链中找到长度最大的一个节点t， 满足s[|s|− len_t] = c，则sc的最长回文后缀就是ctc。
- 插入时，假如回文树上不存在一个节点代表ctc，则需要新建一个新的节点来代表ctc， 否则直接使用这个节点。假如新建了一个节点，还需要求出这个新节点的fail指针对应的节点，相当于在fail[t]对应的fail链中再找一个长度最长的节点v满足s[|s|−lenv] = c。
- 这里 需要注意假如$\mathrm{len}_t = -1$，那么ctc的fail指针应当指向长度为0的节点，否则ctc最长回文后缀必然在回文树上，直接将指针指向对应节点即可。

经典例(ban)题 JZOJ 3654
【APIO2014】回文串

- 考虑一个只包含小写字符的字符串s，定义s的一个子串t的出现值为t在s中的出现次数乘上t的长度。
- 求出s的所有回文子串中的最大出现值。

- $|s| \leq 300000$

BJOI2015 树的同构

- 对于两个树T1和T2，如果能够把树T1的所有点重新标号，使得树T1和树T2完全相同，那么这两个树是同构的。也就是说，它们具有相同的形态。
- 现在，给你M个无根树，请你把它们按同构关系分成若干个等价类。

「NOI2017」蚯蚓排队

Description

给出n个字符，初始每个字符单独成字符串。支持m次操作，每次为一下三种之一：

1 i j：将以i结尾的串和以j开头的串连到一起。

2 i：将i所在串从i位置和i下一个位置之间断开。

3 S k ：对于字符串S每个长度为k的子串，统计它在这n个字符组成所有字符串中出现的次数，求所有统计结果的乘积模998244353的结果。

$n \le 2e5$ $m \le 3e5$ $\Sigma|S| < 1e7$ $k \le 50$

2操作次数小于1e3

- 给出n个各不相同的字符串，给出k，要求选出k个字符串使选出的字符串之间两两的LCP之和最大，输出最大值

- n, len, k ≤ 2000

【NOI2014】动物园

- 对于字符串 s 的每一个前缀，求出有多少个子串既是它的后缀同时又是它的前缀，并且该后缀与该前缀不重叠

- $Q \leq 5$
- $Len \leq 1e6$

- 给出一个长度为n的01串，求这个串在随机01串中第一次出现的期望位置

- N ≤1e7

Nowcoder 131-D 回文

- 字符串 s 只包含小写英文字母。有四种操作，每次操作你可以选择其中一种：

- 删除字符串的第一个字母。
- 删除字符串的最后一个字母。
- 在字符串的头部添加任意一个你想要的字母。
- 在字符串的尾部添加任意一个你想要的字母。

- 删除一个第 i 种英文字母需要的花费是 A_i，添加一个第 i 种英文字母的花费是 B_i。

- 问将字符串 s 变成回文串需要的最小花费是多少？
- $1 \le |S| \le 1e5$, $1 \le A_i, B_i \le 1e9$.

「NOI2016」优秀的拆分

如果一个字符串可以被拆分为 AABB 的形式，其中 A 和 B 是任意**非空**字符串，则我们称该字符串的这种拆分是优秀的。
例如，对于字符串 aabaabaa ，如果令 $\mathrm{A} = \mathtt{aab}$，$\mathrm{B} = \mathtt{a}$，我们就找到了这个字符串拆分成 AABB 的一种方式。

一个字符串可能没有优秀的拆分，也可能存在不止一种优秀的拆分。
比如我们令 $\mathrm{A} = \mathtt{a}$，$\mathrm{B} = \mathtt{baa}$，也可以用 AABB 表示出上述字符串；但是，字符串 abaabaa 就没有优秀的拆分。

现在给出一个长度为 n 的字符串 S，我们需要求出，在它所有子串的所有拆分方式中，优秀拆分的总个数。这里的子串是指字符串中连续的一段。

以下事项需要注意：

1. 出现在不同位置的相同子串，我们认为是不同的子串，它们的优秀拆分均会被记入答案。
2. 在一个拆分中，允许出现 $\mathrm{A} = \mathrm{B}$。例如 cccc 存在拆分 $\mathrm{A} = \mathrm{B} = \mathtt{c}$。
3. 字符串本身也是它的一个子串。

对于全部的测试点，$1 \leq T \leq 10$，$n \leq 30000$。

Jzoj6042 Second

给定字符串 s[1..n]，请给 $k_1, k_2, .., k_n$ 赋值，满足 $0 \le k_i \le 1$，并且 $\sum_{i=1}^{n} k_i = 1$。使得 $\max_i\{\sum_j k_j * lcp(s[i..n], s[j..n])\}$ 最小，其中 lcp(s[i..n], s[j..n]) 表示字符串 s 从下标 i 开始的后缀与字符串 s 从下标 j 开始的后缀的最长公共前缀长度。请输出对应的最小值。

- |s| ≤ 1e6

CTSC2010 珠宝商

- 有一棵n个节点的树和一个长度为m的字符串S，树上每个节点有一个字符。对于每两个点组成的点对，它们之间路径上的点组成了一个字符串，求所有这样字符串出现次数的和
- n,m<=50000

子串

Bobo 有 n 个字符串 $S[1], S[2], \ldots, S[2]$.

他还有 q 个形如 (l_i, r_i, P_i) 的问题，代表他想知道 $S[l_i], S[l_i + 1], \ldots S[r_i]$ 中 P_i 母串的数量。

注：字符串 A 是 B 的母串，当且仅当 B 在 A 中（作为连续子串）出现。

- 30% 字符串总长T≤5000
  60% T≤200000
  100% T≤500000
  字符串只有字符集是{a,b}

JZOJ 4387

- 从一个空串开始，每次进行以下两种操作之一：
- 在串的头部或尾部插入一个字符
- 将整个串复制一遍，再反序接到原来串的后面
- 最少要几次操作才能获得目标串

- $Q \leq 10$， $|s| \leq 1e5$

参考（鸣谢）

- 各位dalao的各种论文和ppt

- 怎么说呢版权意识还是要有的