

比赛报名

参见快快首页公告栏

→ 注册报名缴费: <http://rg.noi.cn>

填错**不能修改**! (手动呵呵)

错峰报名!

推荐J+S

→ 信息填报

与报名信息一致

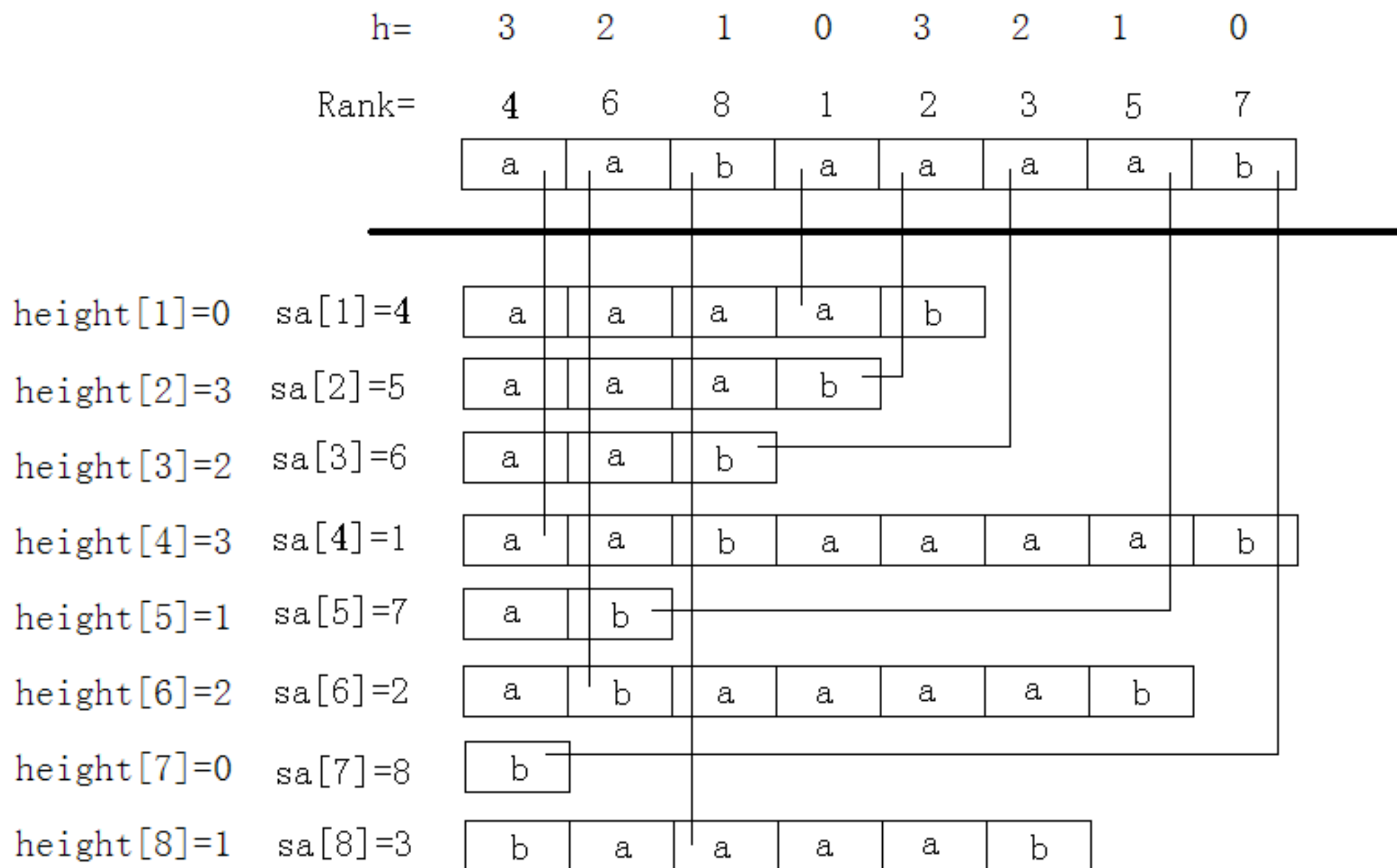
也需汇总给官方的



2019年 CSP 非专业级别
计算机能力认证 报名统计

后缀数组

Suffix Array



P692 最强大脑7

建模

→ 破题（模板题）

字符串的所有**后缀按字典序排序**

排序后的结果称为**后缀数组** (Suffix Array, SA)

→ 暴力算法（直接排序）

复杂度： $O(n^2 \log n)$

随机数据： $O(n(\log n)^2)$ (80分)

→ 前缀散列+LCP（自己写字符串比较）

复杂度： $O(n(\log n)^2)$



一些(dui)定义

→ $S_i = S$ 以 i 开始的后缀 (Suffix)

→ 字符串 a, b 长 k 的前缀 a 较小

称 **k -前缀意义下 a 小于 b** , 记 **$a <_k b$**

(如全长度不足 k , 以 $\backslash 0$ 补足)

→ $S_1..S_n$ 在 **k -前缀意义下**排序 (含并列)

结果称为 **k -前缀意义下的后缀数组**, 记为 **SA_k**

$k \geq n$, $SA_k = SA$

→ 求 **SA_1** 只需要将各字符排序

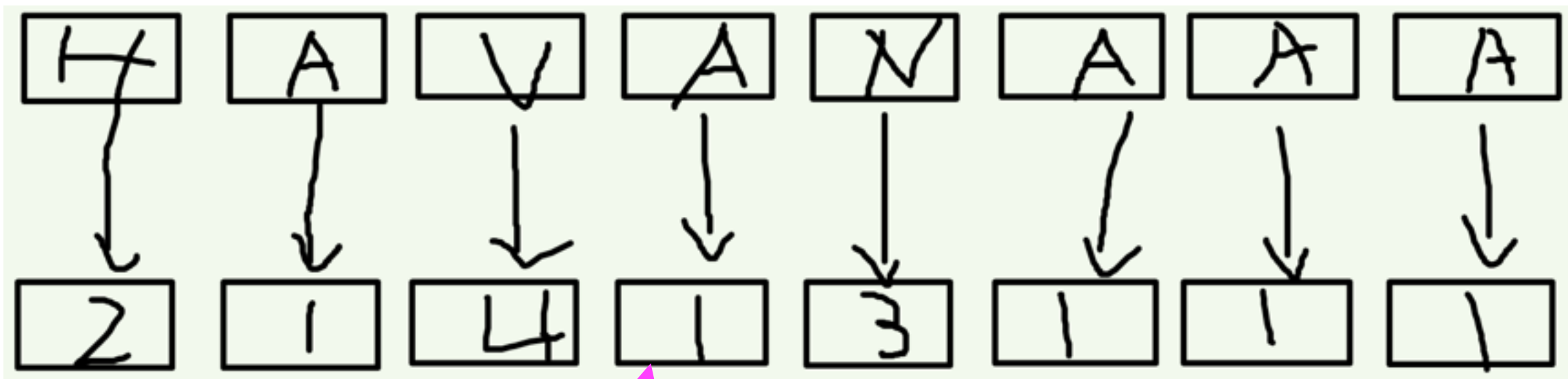
采用计数排序, 复杂度为 $O(n)$

→ Exp, $S = \text{aabdabc}$

$S_5 = \text{"abc"}$, $S_2 = \text{"abdabc"}$

$S_5 =_2 S_2$, $S_5 <_3 S_2$





→ 数字表示每个1-前缀的**排名 (含并列)**

可理解为离散化

这个称为rank (SA的**倒排**)，记为 rk_1

→ **排序=第i大是几；倒排=第i个是第几大**

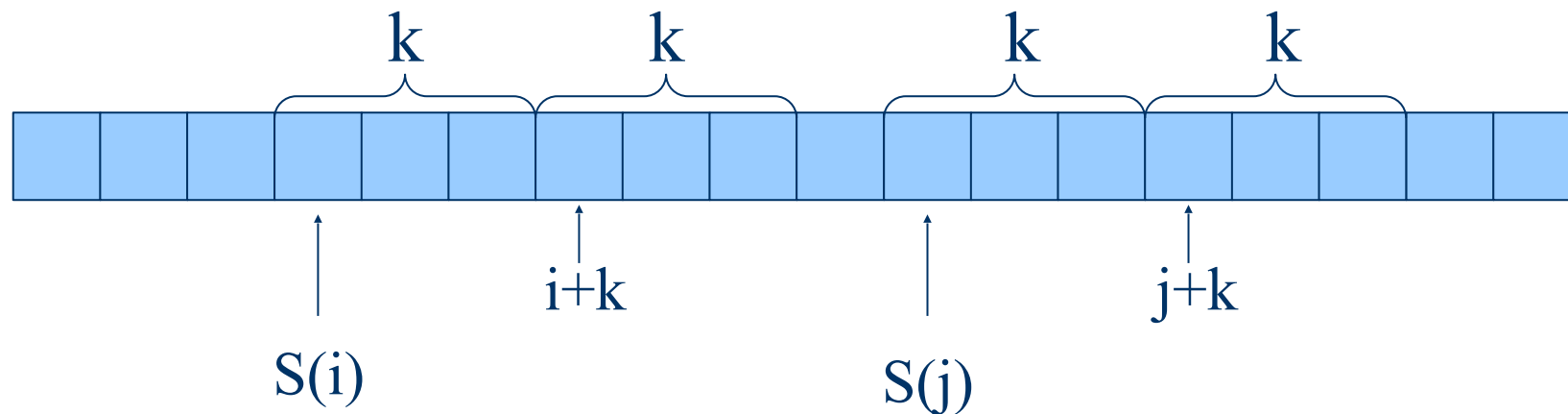
m=值域，初始为INF

```
void calcsa(int n, int m){
    for (int i=1; i<=n; i++) c[x[i]=a[i]]++;
    for (int i=1; i<=m; i++) c[i]+=c[i-1];
    for (int i=n; i; i--) sa[c[x[i]]--]=i;
```

计数器前缀和 (排名)

遍历计数排序求SA
排 $c[x[i]]$ 的是i

倍增



→ 已知 SA_k 求 SA_{2k}

前k位不同，则比前k位（第一关键字）

前k位相同，则比后k位（第二关键字）

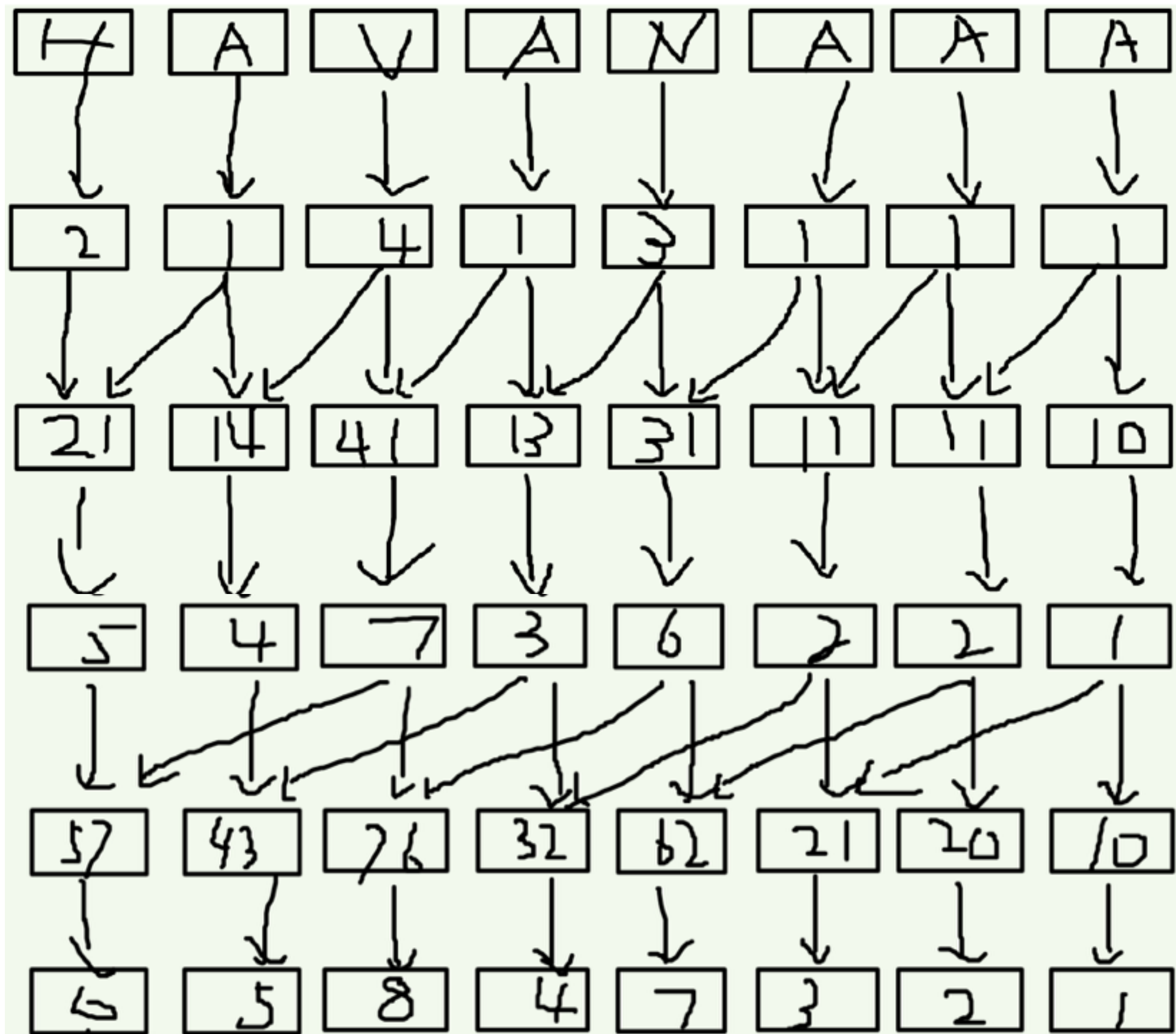
→ k位比较，用 SA_k 排序好的结果（排名）

SA_k 排的是所有后缀的前k位，即所有k位子串

→ 仍使用计数排序，求 SA_{2k} 复杂度为 $O(n)$

总复杂度： $O(n \log n)$





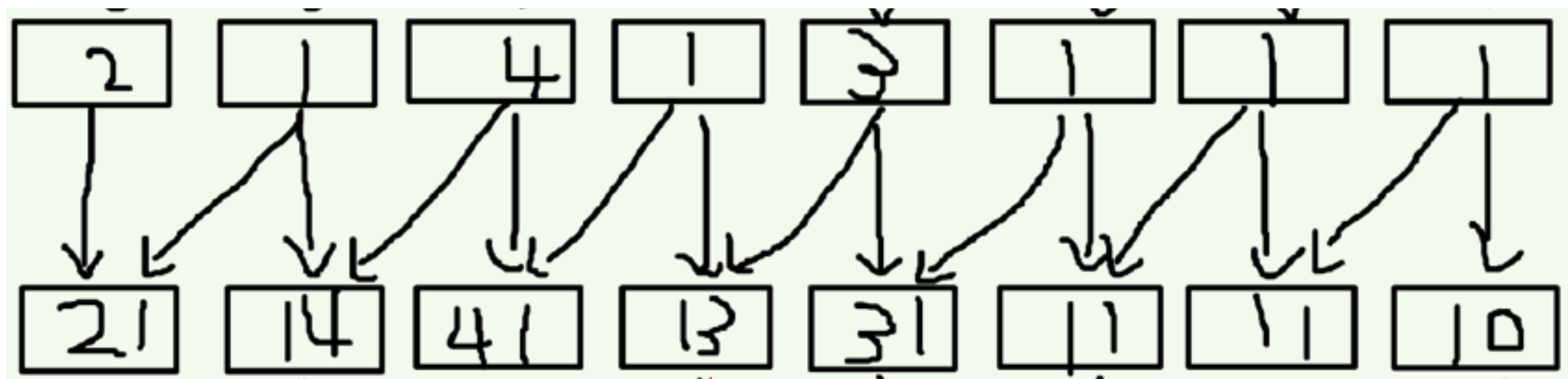
$y = SA_{2k}$ 中后 k 位 (第2关键词) 排序

$x = SA_k$ (第1关键词) 排名, 即 rk_k

```
for (int k=1, p, q=0, j; k<=n && q<=n; k<=<1) {  
  for (p=0, j=n-k+1; j<=n; j++) y[++p]=j;  
  for (j=1; j<=n; j++)  
    if (sa[j]>k) y[++p]=sa[j]-k;  
}
```

没有后 k 位

根据 SA 排名, 每个往前接 k 位, 成为 $2k$ 位



$x = \{2, 1, 4, 1, 3, 1, 1, 1\}$

$sa = \{2, 4, 6, 7, 8, 1, 5, 3\}$ (x 的倒排)

$y = \{8, 1, 3, 5, 6, 7, 4, 2\}$

```

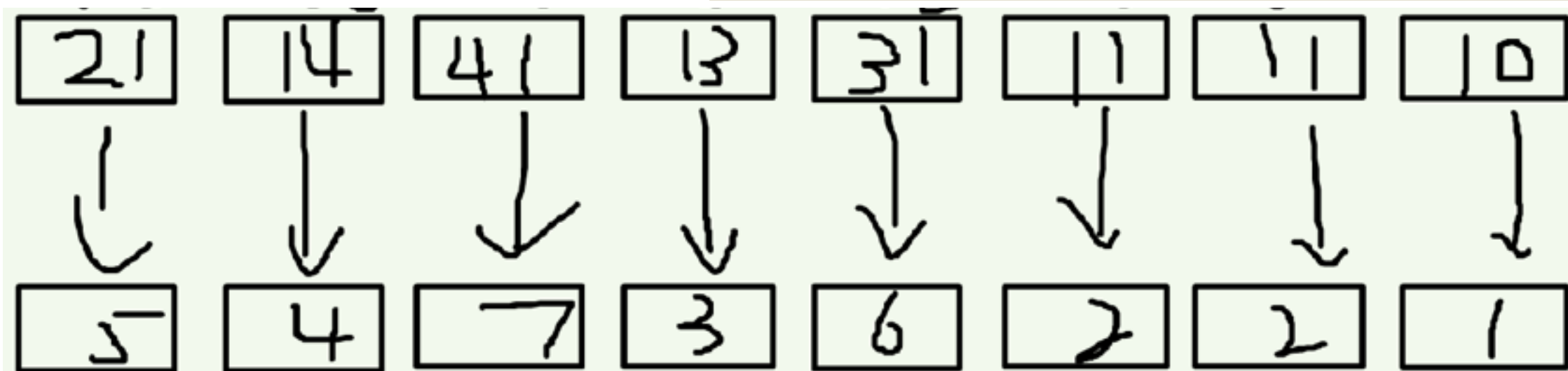
for (j=1; j<=m; j++) c[j]=0;
for (j=1; j<=n; j++) c[x[j]]++;
for (j=1; j<=m; j++) c[j]+=c[j-1];
for (j=n; j; j--) sa[c[x[y[j]]]--]=y[j];

```

第1关键词计数排序

第1关键词排名

x值相同的一组，在SA_{2k}排名相邻
组内按y排名（双关键字计数排序）



x= {2,1,4,1,3,1,1,1}

y= {8,1,3,5,6,7,4,2}

c= {5,1,1,1,0,0,0,0}

→ {5,6,7,8,8,8,8,8}

sa= {8,6,7,4,2,1,4,3}

Exp. j=8

sa[c[x[y[8]]--]=y[8]

sa[5]=2

```

swap(x, y);
for (x[sa[1]]=1, q=j=2; j<=n; j++)
    x[sa[j]]= y[sa[j]]==y[sa[j-1]]
        && y[sa[j]+k]==y[sa[j-1]+k] ? q-1 : q++;
m=q;

```

用y暂存SA_k, x更新为rk_{2k}

q=2k前缀意义下的排名

如并列, 则排名不变, 否则排名加1

```

scanf("%s", s+1);
int n=(int)strlen(s+1);
for (int i=1; i<=n; i++) a[i]=s[i]-' ';
calcsa(n, 10000);

```



不明真相喝水吃饼吃瓜吃鸡腿群众

完整代码（背背背）

```
9 void calcsa(int n,int m){
10     for (int i=1;i<=n;i++) c[x[i]=a[i]]++;
11     for (int i=1;i<=m;i++) c[i]+=c[i-1];
12     for (int i=n;i;i--) sa[c[x[i]]--]=i;
13     for (int k=1,p,q=0,j;k<=n && q<=n;k<=1){
14         for (p=0,j=n-k+1;j<=n;j++) y[++p]=j;
15         for (j=1;j<=n;j++)
16             if (sa[j]>k) y[++p]=sa[j]-k;
17         for (j=1;j<=m;j++) c[j]=0;
18         for (j=1;j<=n;j++) c[x[j]]++;
19         for (j=1;j<=m;j++) c[j]+=c[j-1];
20         for (j=n;j;j--) sa[c[x[y[j]]]--]=y[j];
21         swap(x,y);
22         for (x[sa[1]]=1,q=j=2;j<=n;j++)
23             x[sa[j]]= y[sa[j]]==y[sa[j-1]]
24                 && y[sa[j]+k]==y[sa[j-1]+k] ? q-1 : q++;
25         m=q;
26     }
27 }
```


后缀数组应用

→ 后缀数组三件套

SA数组

rk数组 (SA的倒排)

Height数组 (见后)

→ 求SA还有线性算法 (DC3算法)

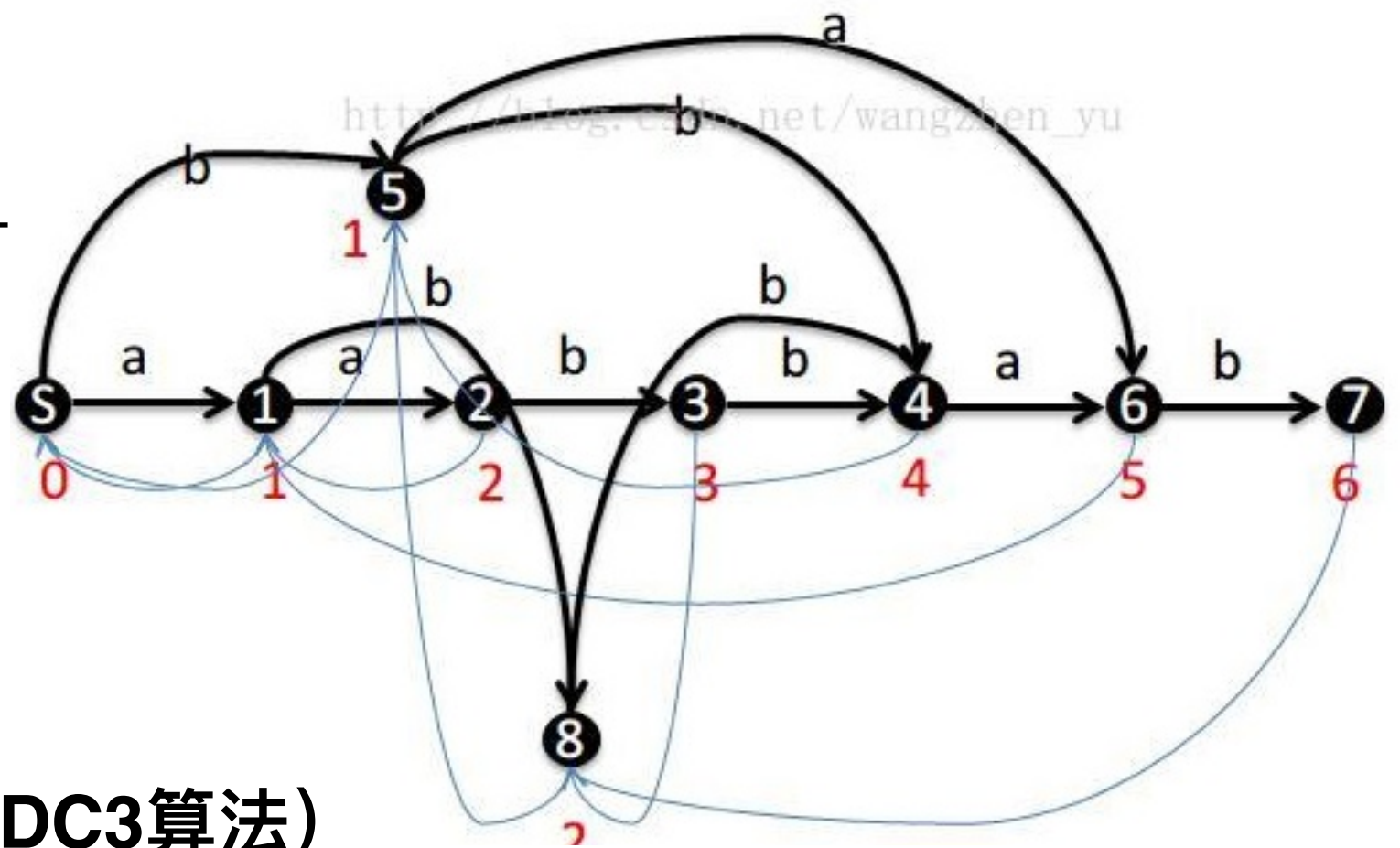
实现更复杂...

想用SA又不会写：暴力SA就是了

→ 后缀自动机 (Suffix Automation, SAM)

→ 应用场景

找回文串、最小循环表示、子串去重、K大子串...



P713 影分身



建模

→ 破题

求序列中**至少出现K次的最长子串**

→ 枚举+前缀散列+计数器

复杂度: $O(n^2+C)$

→ 二分答案

$OK(m)$ =是否存在出现 $\geq K$ 次, 长度 $\geq m$ 的子串

1.前缀散列+计数器: $O(n\log n+C)$ (可行自便)

2.后缀数组

```
for (int l=1, r=n, m; l<=r; ) {  
    m=(l+r)/2;  
    OK(m) ? ans=m, l=m+1 : r=m-1;  
}
```


子串=后缀的前缀

- 如一个子串出现了多次
则起对应后缀**在SA中连续**
- 只要找SA中
有否连续k个的 **$LCP \geq m$** 即可



S=abababbaba

SA=

a
aba
abababbaba
ababbaba
abbaba
ba
baba
bababbaba
babbaba
bbaba

SA中有否连续k个的 $LCP \geq m$

Height数组 (高度)

→ Height=SA中相邻项的LCP

$$\text{height}[i] = \text{LCP}(\text{Sr}[i-1], \text{Sr}[i-1])$$

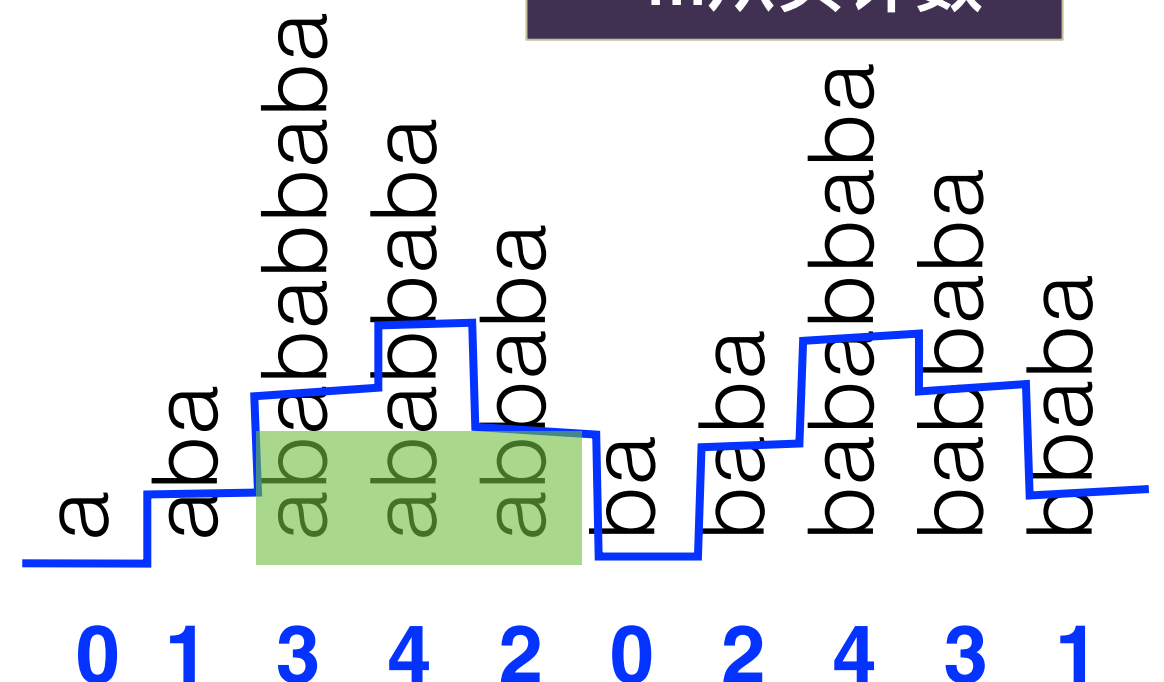
→ 找height连续 $\geq m$ 的最长的一段即可

num=连续height \geq m的长度

```
int OK(int m) {
    for (int i=1, num=0; i<=n; i++)
        if (ht[i]>=m) {
            if (++num==K-1) return 1;
        } else num=0;
    return 0;
}
```

满K就ok

<m从头计数



计算Height

$$\text{height}[i] = \text{LCP}(\text{Ssa}[i-1], \text{Ssa}[i])$$

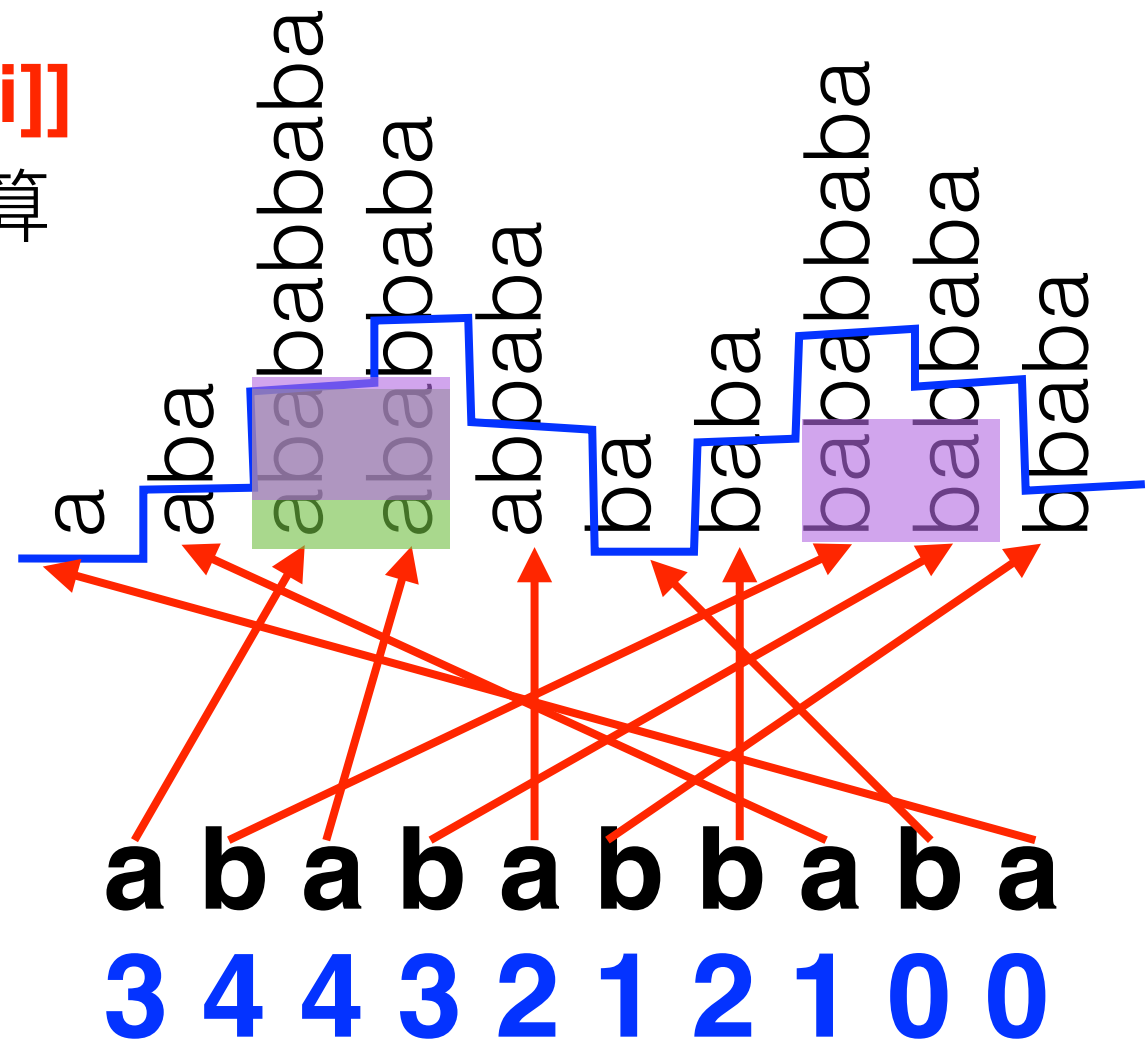
→ 将height以原串下标顺序排列

特征: $\text{height}[\text{rk}[i]] \geq \text{height}[\text{rk}[i-1]] - 1$ (降幅 ≤ 1)

证明: 如图所示

→ 以 $i=1..n$ 的顺序计算 $\text{height}[\text{rk}[i]]$

则只要从上一项减1起增量式计算



计算Height

以 $i=1..n$ 的顺序计算 $height[rk[i]]$

则只要从上一项减1起增量式计算

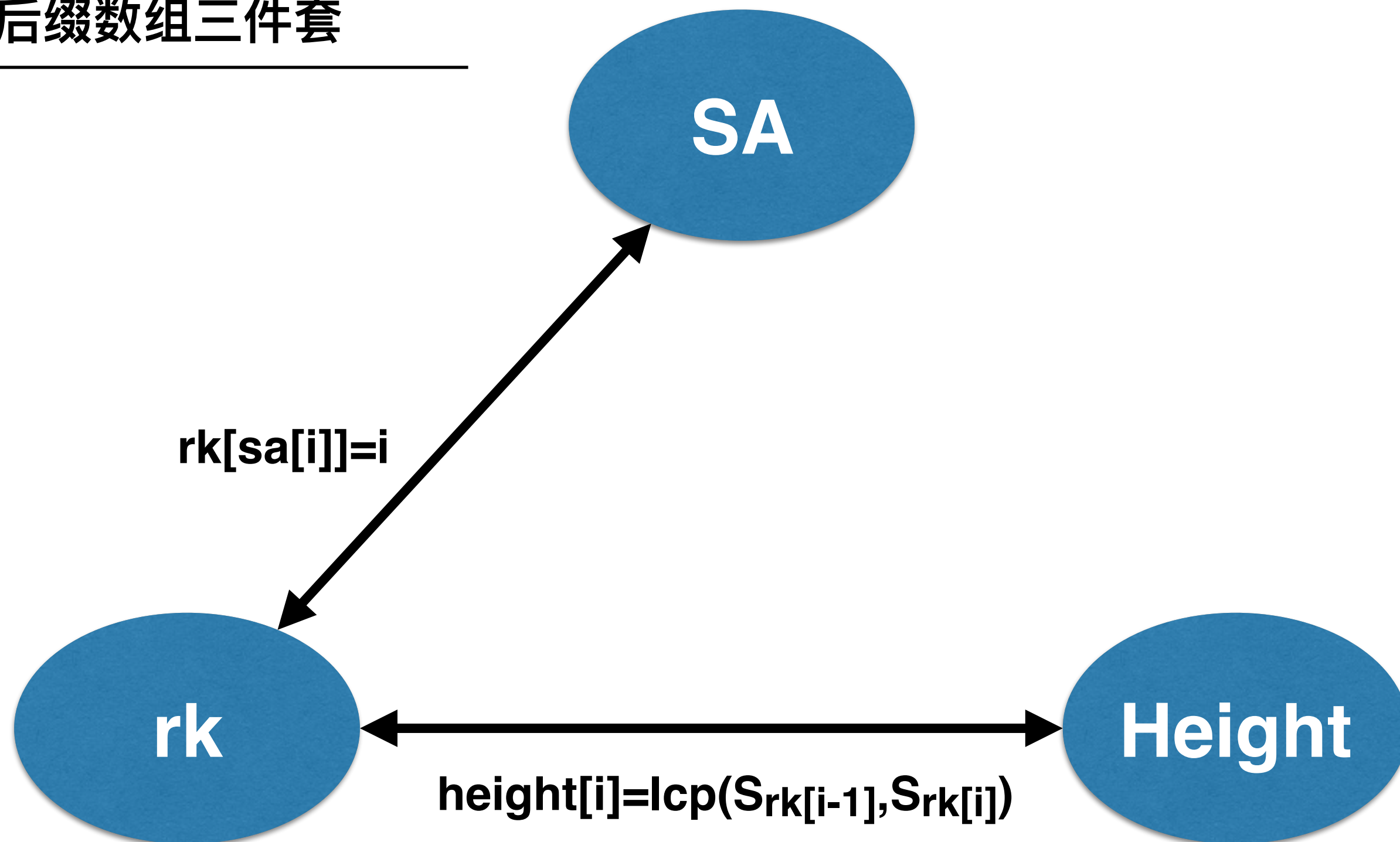
```
void calcHeight() {  
    for (int i=1; i<=n; i++) rk[sa[i]]=i;           计算rk  
    for (int i=1, k=0; i<=n; i++) {                以i递增顺序计算height[rk[i]]  
        if (rk[i]==1) continue;  
        for (k?k--:0;                               从上一个-1开始  
             a[i+k]==a[sa[rk[i]-1]+k]; k++);  
        ht[rk[i]]=k;  
    }  
}
```

→ 复杂度

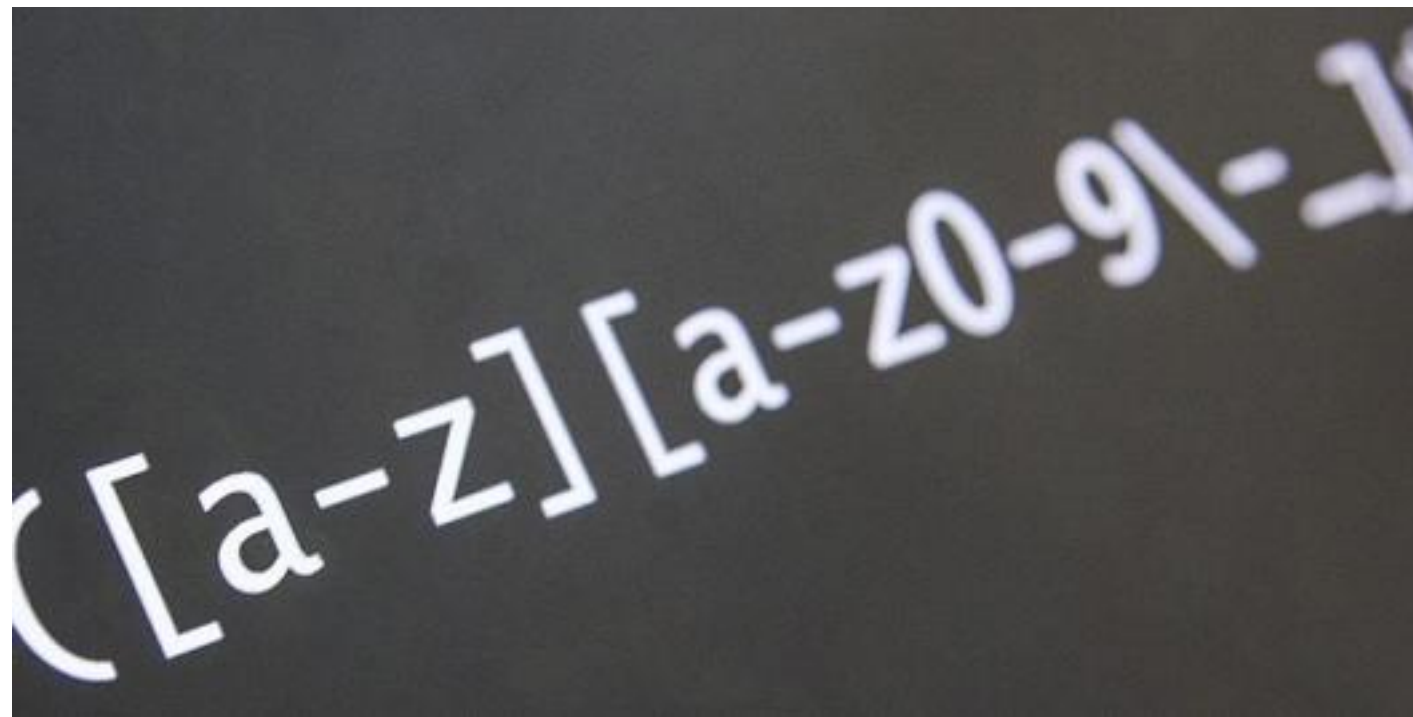
每次比较次数= $height[rk[i]]-height[rk[i-1]]+2$

对 i 求和得: $height[rk[n]]+2n=O(n)$

后缀数组三件套



P714 误差模式匹配



`([a-z][a-z0-9]{-1`

初步分析

→ 破题

求允许至多3个误差的模式匹配

→ 暴力方法

复杂度: $O(nm)$

→ 传统方法均依赖精确匹配, 无法使用

KMP、Rabin-Karp、Boyer-Moore

→ 先枚举开始匹配的位置

然后其实是子串后缀与母串后缀

求不超过3次LCP的过程

aabbababba
aa**a**bab**b**a

aabbababba
aa**a**babba
aa**a**bab**b**a
aa**a**bab**b**a

求子串与母串后缀的LCP

→ 先一个小技巧

aabbababba*aaababba

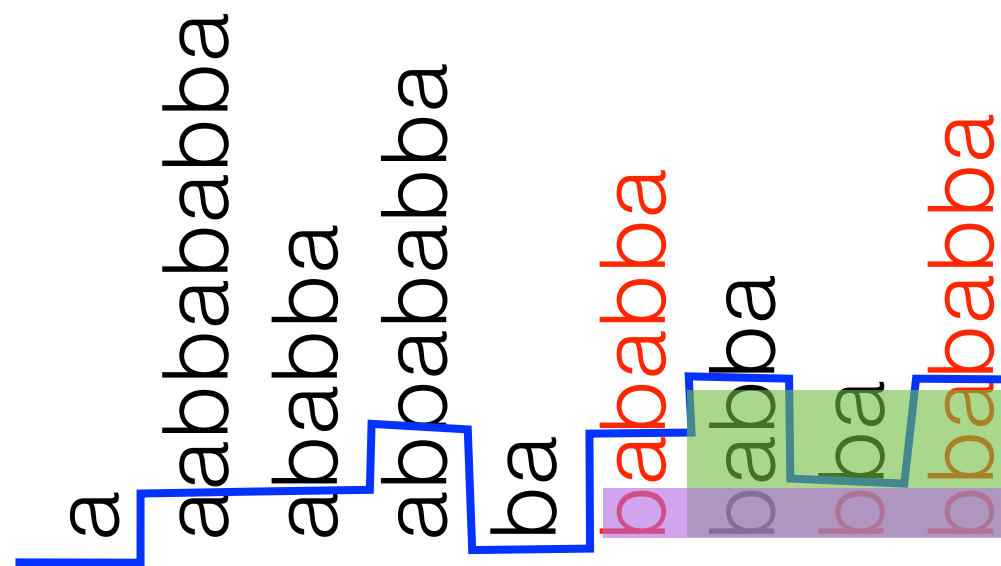
把两个串用特殊符号链接起来

这样就转化为**同一个串的2个后缀求LCP**

→ 该值等于**SA中两后缀之间height的最小值**（非人话）

查询height数组上的**区间最小值**（RMQ）

height数组上建立**ST表**




```

void calcsa(){
    int p=0,m=50;
    memset(c,0,sizeof(c));
    {...} 你懂的
    for(int i=1,k=0;i<=n;++i) {
        if (rk[i]==1) continue;
        for(k?k--:0;
            a[i+k]==a[sa[rk[i]-1]+k];k++);
        st[0][rk[i]]=k;
    }
    for(int j=1;j<=lg[n];++j)
        for(int i=1;i+(1<<j)-1<=n;++i)
            st[j][i]=min(st[j-1][i]
                        ,st[j-1][i+(1<<(j-1))]);
}

```

计算height

建ST表

```
int lcp(int i, int j) {
    i = rk[i], j = rk[j];
    if (i > j) swap(i, j);
    if (i == j) return 1e9;
    ++i;
    int h = lg[j - i + 1];
    return min(st[h][i], st[h][j - (1 << h) + 1]);
}
```

查询任意2个后缀的LCP

i=母串的起始匹配位置

```
for(int i=1, j, cnt; i <= n0 - m + 1; ++i) {
    for(j=1, cnt=0; j <= m && cnt <= 3; )
        if (a[i+j-1] != a[n0+1+j]) ++cnt, ++j;
        else j += lcp(i+j-1, n0+1+j);
    ans += cnt <= 3;
}
```

至多3次LCP

→ 复杂度: $O((n+m)\log(n+m))$

作业

- 1.最强大脑7 (P692)
- 2.影分身 (P713)
- 3.误差模式匹配 (P714)