



信息学

最小表示法与拓展KMP(Z函数)



补充知识：最小表示法



西南大学附属中学
High School Affiliated to Southwest University

- 给定字符串 $S[1 \sim n]$ ，如果不断把它的最后一个字符提到开头，最终会得到 n 个字符串，称这 n 个字符串是**循环同构**的。
- 这些字符串中**字典序最小**的一个，称为字符串 S 的最小表示。

abcd



最小表示

dabc

cdab

bcda

西 | 大 | 附 | 中 | 信 | 息 | 学 | 竞 | 赛 |
High School Affiliated to Southwest University



最小表示求解方法



西南大学附属中学
High School Affiliated to Southwest University

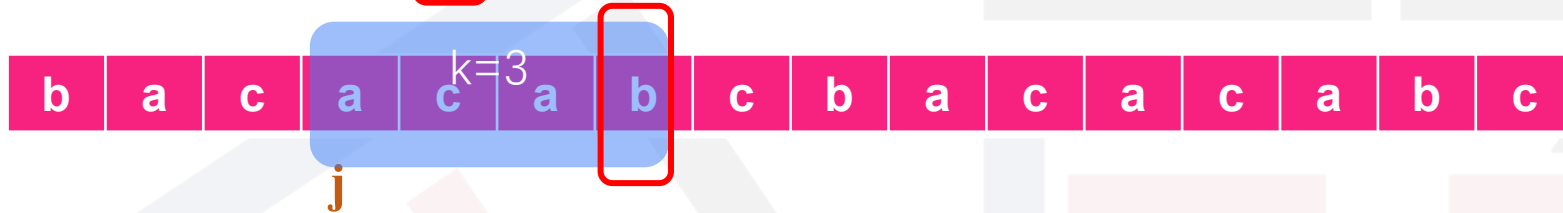
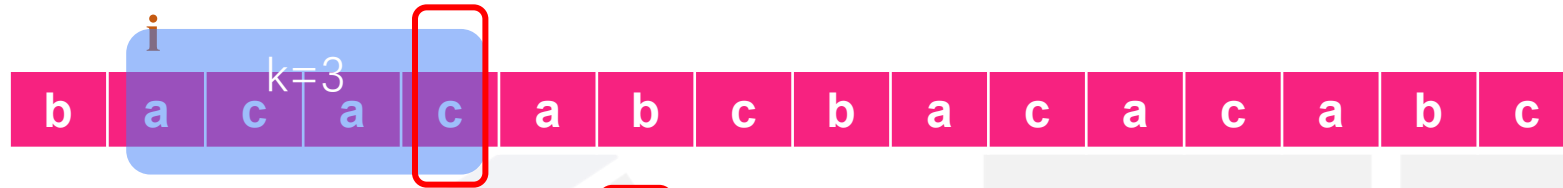
- 本质上来讲，是在串中找一个最小起点，依次输出。
- 如何做？思考一下。
- 考虑到串可能为 dcba 或为 caaaa 这种情况。
- 我们需要设置指针i j，i指向最小表示的位置。j用来移动比较。
- 具体算法如下
 - $i=0, j=1$
 - `if s[i] > s[j] : i=j, j++` //找到个更小的起点，赶快转移
 - `if s[i] < s[j] : j++` //无事发生
 - `if s[i] ==s[j] :` //竟然一样了，可能遇到 aab 和 aac的情况，向后寻找
 - `for k=1 ~ 有效范围:`
 - `if s[i+k]<s[j+k] : j++,break` //i的位置更好
 - `if s[i+k]>s[j+k] : i=j,j++,break` //j的位置更好
- 最后答案为i



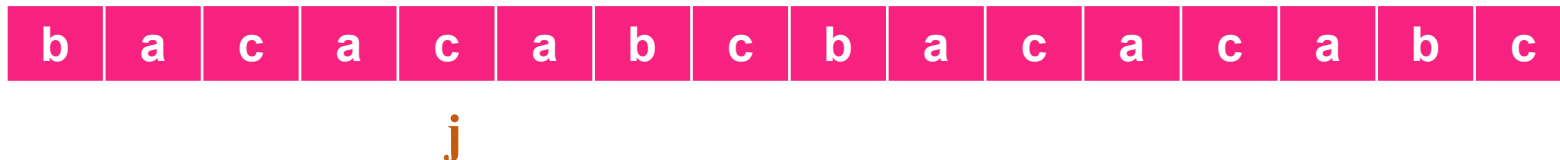
最小表示求解方法-演示过程



西南大学附属中学
High School Affiliated to Southwest University



$s[i+k] > s[j+k]$
 $i = j, j++$

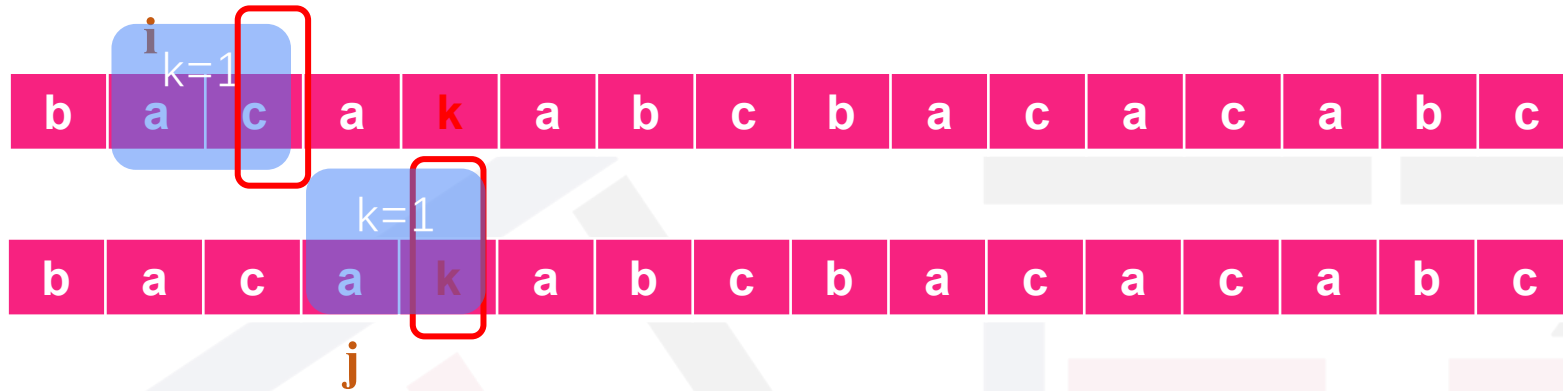




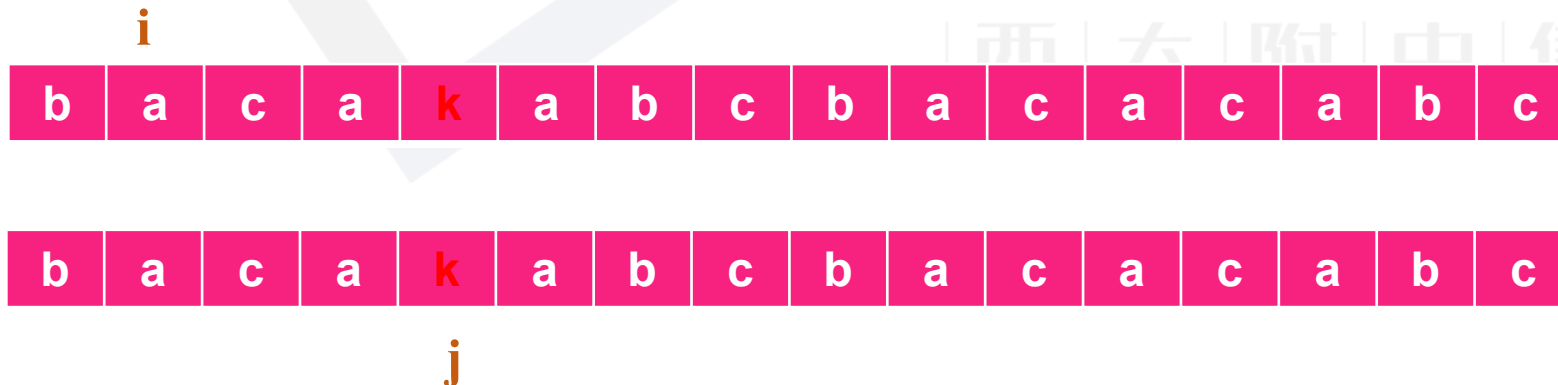
最小表示求解方法-演示过程



西南大学附属中学
High School Affiliated to Southwest University



$s[i+k] < s[j+k]$
 $j++$





最小表示求解方法-复杂度



西南大学附属中学
High School Affiliated to Southwest University

想一个边界一点的条件：

假设遇到 **aaaaaaaab**

Diagram illustrating the KMP algorithm's failure function. The string "aaaaaaaab" is shown. A green arrow labeled j points to the first 'a', and another green arrow labeled $j+k$ points to the last 'a'. A green arrow labeled i points to the first 'a', and another green arrow labeled $i+k$ points to the last 'a'. A green line connects the j and $j+k$ arrows, and another green line connects the i and $i+k$ arrows.

然后发现 i 更优秀，更新 $j++$
复杂度 $O(n^2)$

能不能再快一点？
回忆KMP的基本思路？
信息复用！



最小表示求解方法: $O(n)$



西南大学附属中学
High School Affiliated to Southwest University

考虑下列情况

cdac
↑
z

cdaa
↑
j

此时不再固定
谁表示最小起点
 i j 均有可能

当 $s[i+k] > s[j+k]$ 时

$i, i+1, i+2, \dots, i+k$ 都不可能是最小表示的起点, 直接 $i += k+1$

因为 $j+k$ 的位置小, 以 $s[j+k]$ 结尾的位置更优秀



最小表示求解方法: $O(n)$



西南大学附属中学
High School Affiliated to Southwest University

1. 拷贝一份, (abc 变为 $abcabc$)
2. 初始化 $i=0, j=1$
3. 通过向后扫描的方法, 比较 i 为起点和 j 为起点的两个循环同构串,
 - (1) 如果 $SS[i+k]=SS[j+k]$
 $k++$;
如果 $k=n$, 停止。
 - (2) 如果 $SS[i+k]>SS[j+k]$
 $i+=k+1$, 若 $i==j$, 则 $i++$
 - (3) 如果 $SS[i+k]<SS[j+k]$
 $j+=k+1$, 若 $j==i$, 则 $j++$
4. 最后 $\min(i, j)$ 就是答案



求最小表示代码 $O(n)$:



西南大学附属中学
High School Affiliated to Southwest University

```
int n=strlen(s+1);    //输入scanf('%s',s+1);
for(int i=1;i<=n;i++) s[n+i]=s[i];
int i=1,j=2,k;
while(i<=n&&j<=n){
    k=0;
    while(k<n&&s[i+k]==s[j+k]) k++;
    if(k==n) break;
    if(s[i+k]>s[j+k]){
        i=i+k+1;
        if(i==j) i++;
    }
    if(s[i+k]<s[j+k]){
        j=j+k+1;
        if(j==i) j++;
    }
}
ans=min(i,j);
```

为什么时间复杂度是 $O(n)$?

考虑指针 i 在数组上滑动，
不论是 j 动还是 k 动，
 i 总会向后移动 j 个单位或 k 个单位
所以循环次数不会超过 $2N$ 次



最小表示法能解决什么问题？



西南大学附属中学
High School Affiliated to Southwest University

您可能听说没有两个雪花是相似的。你的任务是编写一个程序来确定这是否真的如此。每个雪花都有六个分支，用六个整数代表，这六个整数是从任意一个分支开始，朝顺时针或逆时针方向遍历得到的。输入多个雪花，判断是否有形状一致的雪花存在。

之前不是很困恼顺序不好找么？

正着，反正跑一遍最小表示法，求出正着走的最小位置，反着走的最小位置。然后位权展开计算出hash_正，和hash_反
把min(hash_正, hash_反)存入set，利用set自带find完成查找。

High School Affiliated to Southwest University



雪花代码-最小表示法



西南大学附属中学
High School Affiliated to Southwest University

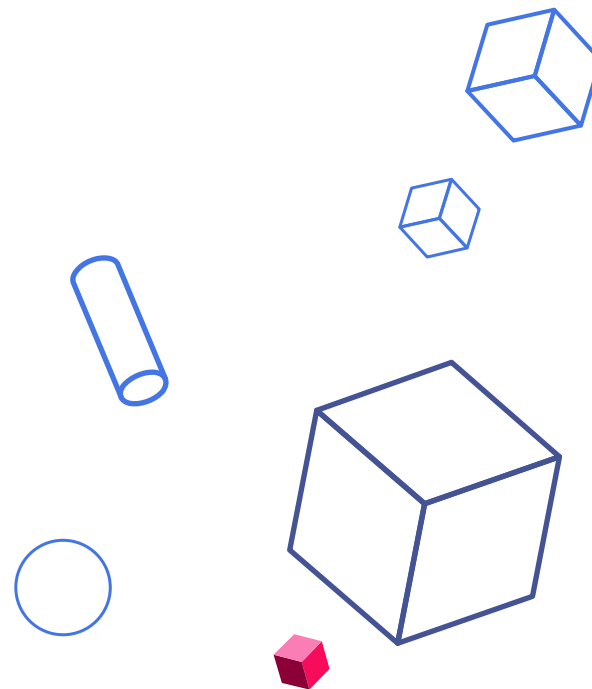
```
#include <bits/stdc++.h>
#define fastio ios::sync_with_stdio(false);cin.tie(0)
using namespace std;
#define int long long

const int maxn = 1e6 + 5;
const double pi = acos(-1.0);
const int mod = 991;
int s[300],ss[300];
int ge(int s[]) {
    int len = 6;
    for(int i=1; i<=len; i++) s[i+len]=s[i];
    int i=1,j=2,k;
    while(i<=len&&j<=len) {
        for(k=0; k<len&&s[i+k]==s[j+k]; k++);
        if(k==len) break;
        if(s[k+i]>s[j+k]) {
            i=i+1+k;
        } else {
            j=j+1+k;
        }
        if(j==i) j++;
    }
    int pos=min(i,j);
    return pos;
}
```

```
set<int> mp;

main() {
    int n;
    cin>>n;
    while(n--) {
        for(int i=1; i<=6; i++) {
            cin>>s[i];
        }
        for(int i=1; i<=6; i++) ss[7-i]=s[i];
        int hash = 0;
        int k1=ge(s);
        for(int i=0; i<6; i++) {
            hash=hash*mod+s[k1+i];
        }
        int k2=ge(ss);
        int hash2=0;
        for(int i=0; i<6; i++) {
            hash2=hash2*mod+ss[k2+i];
        }
        if(mp.find(min(hash,hash2))!=mp.end()) {
            cout<<"Twin snowflakes found."<<endl;
            return 0;
        }
        mp.insert(min(hash,hash2));
    }
    cout<<"No two snowflakes are alike."<<endl;
    return 0;
}
```

拓展KMP (z函数)





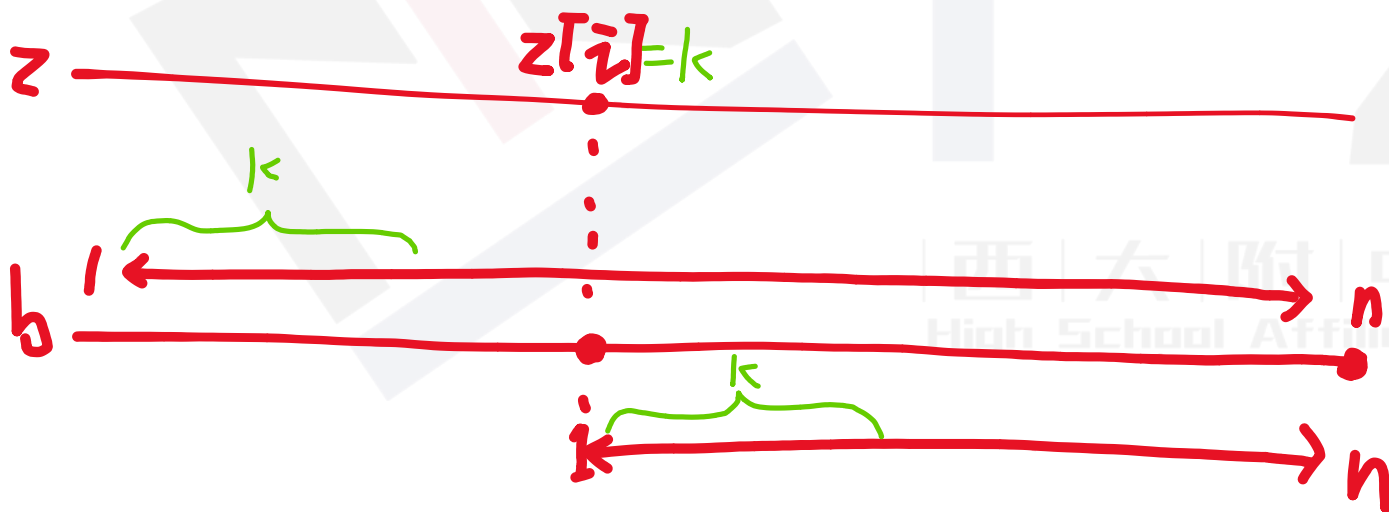
Z函数



西南大学附属中学
High School Affiliated to Southwest University

字符串 b 和数组 z

$z[i]$ 表示 $b[1 \sim n]$ 和后缀 $b[i \sim n]$ 的最长公共前缀。



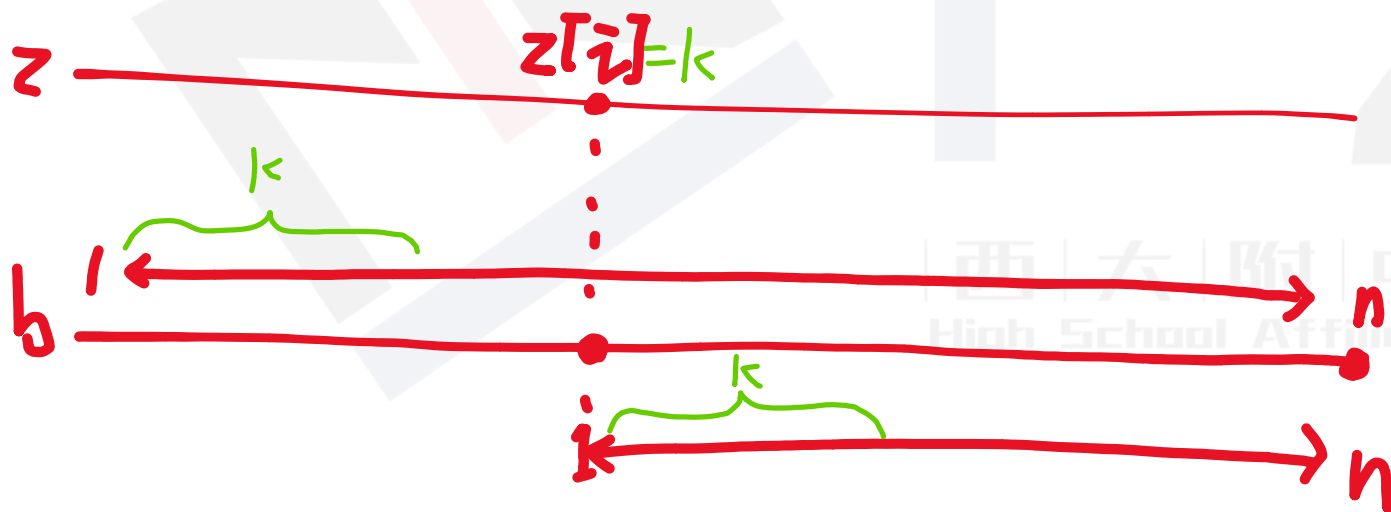



Z函数-暴力方法



西南大学附属中学
High School Affiliated to Southwest University

从 $0 \sim n-1$ 枚举 i ，根据枚举得出 i 枚举 k
时间复杂度 $O(n^2)$





Z函数-优雅方法



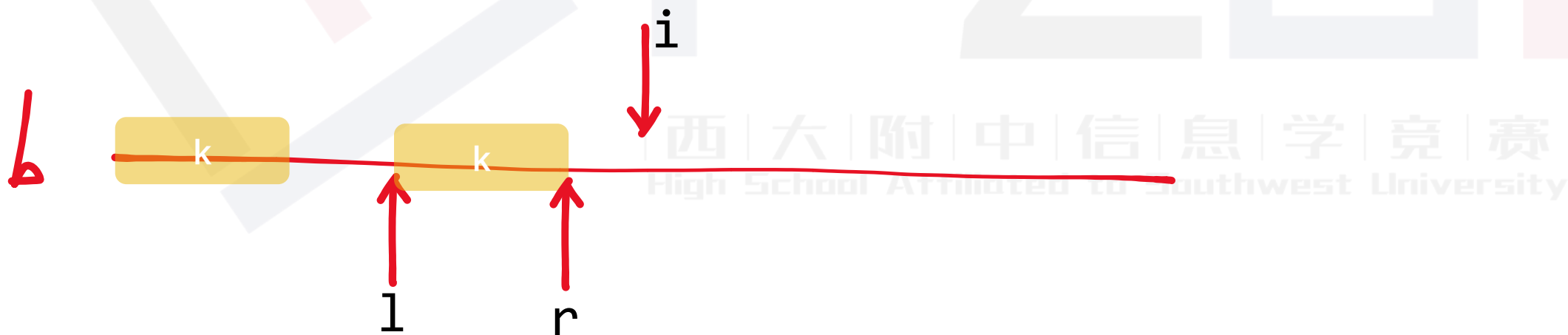
西南大学附属中学
High School Affiliated to Southwest University

根据之前的信息，减少不必要的计算！

寻找限制条件下的状态转移函数，使得可以借助之前的状态来加速计算新的状态。

已经求出 $Z[1] \sim Z[i-1]$,正在求 $Z[i]$

记 $\max(Z[1], Z[2], \dots, Z[i-1])$
在 $i=1$ 时拿到最大值，其右端点记为 r



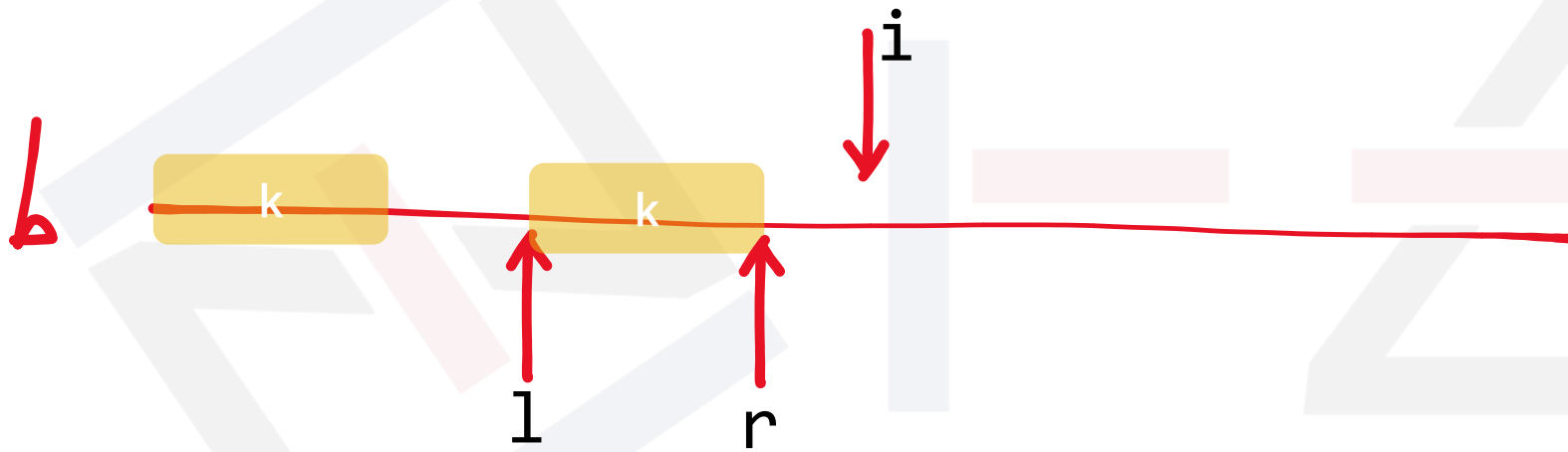


Z函数-优雅方法



西南大学附属中学
High School Affiliated to Southwest University

记 $\max(Z[1], Z[2], \dots, Z[i-1])$
在 $i=1$ 时拿到最大值，其右端点记为 r



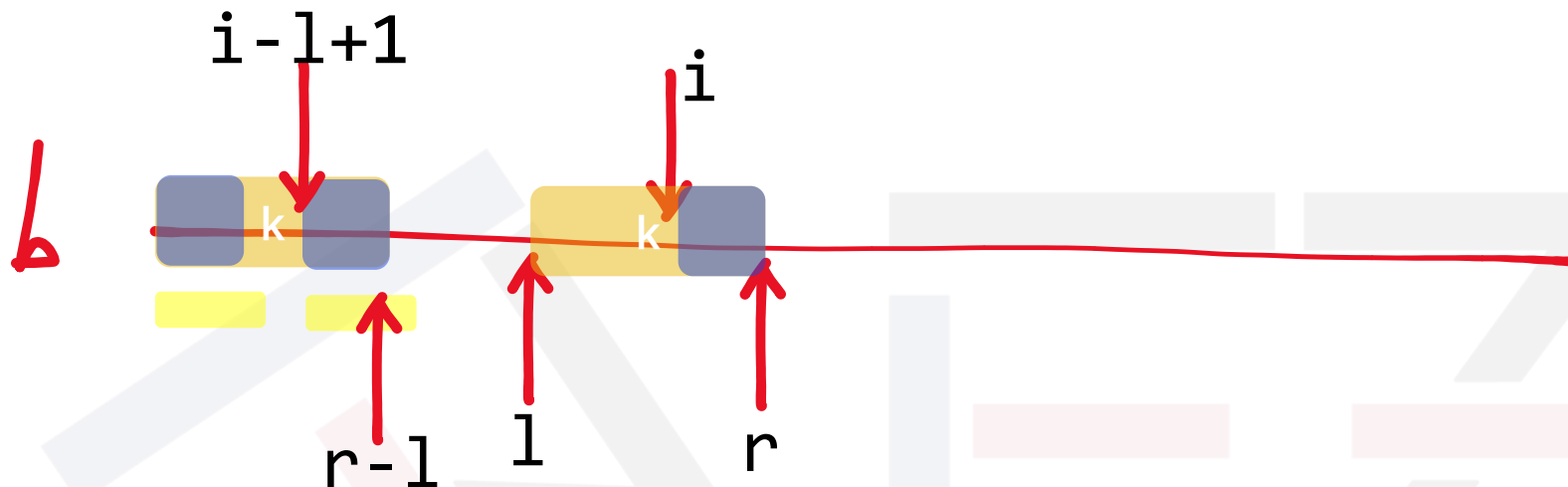
如果 $i > r$ 则 $z[i] = 0$ 暴力匹配即可：
然后反复比较 $b[i+z[i]]$ 与 $b[z[i]]$ 是否相同，相同则 $z[i]++$ ，重复执行。



Z函数-优雅方法



西南大学附属中学
High School Affiliated to Southwest University



如果 $i \leq r$, 因为蓝色部分相等,

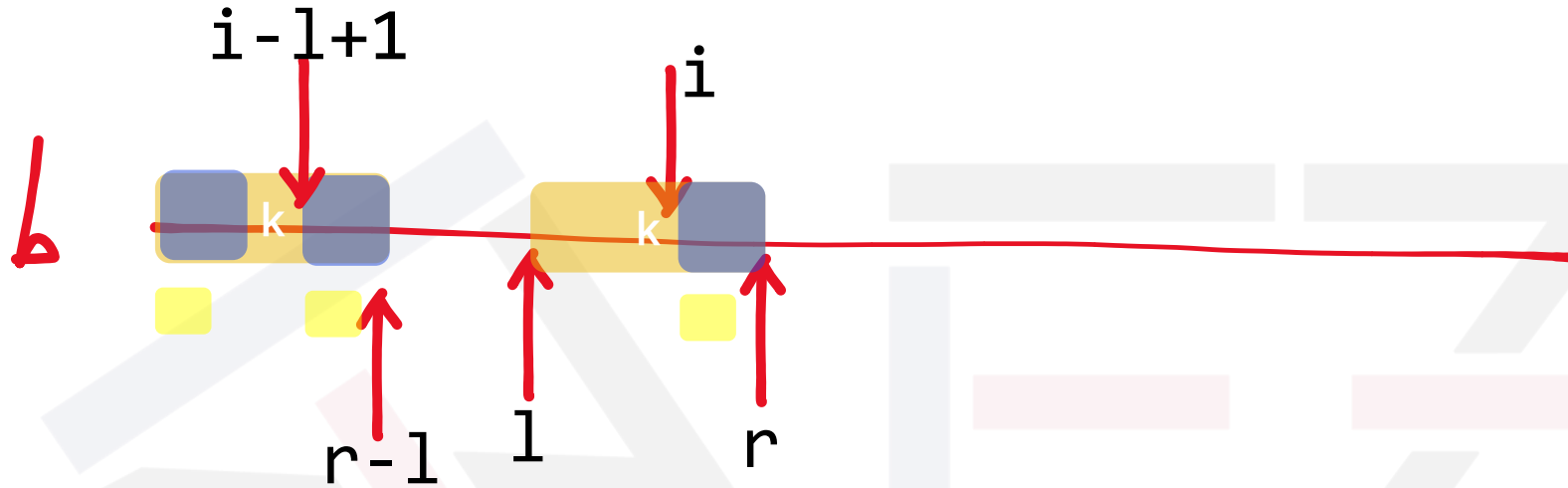
如果 $z[i-1+1]$ 黄色部分 $> (r-1) - (i-1) + 1 = r-i+1$, 则三处蓝色相等
 $z[i] = r-i+1$



Z函数-优雅方法



西南大学附属中学
High School Affiliated to Southwest University



如果 $i \leq r$, 因为蓝色部分相等,

如果 $z[i-1+1]$ 黄色部分 $> (r-1) - (i-1) + 1 = r-i+1$, 则三处蓝色相等

$z[i] = r-i+1$

如果 $i \leq r$, 因为蓝色部分相等,

如果 $z[i-1+1]$ 黄色部分 $< (r-1) - (i-1) + 1 = r-i+1$, 则三处黄色相等

$z[i] = z[i-1+1]$



Z函数-优雅方法



西南大学附属中学
High School Affiliated to Southwest University

1. 如果 $i \leq r$:

(1) 初始值: $z[i] = \min(z[i-1+1], r-i+1)$

(2) 继续比较 $b[i+z[i]]$ 和 $b[z[i]]$ 是否相同, 相同 $z[i]++$, 重复 (2); 否则停止。

(3) 同时更新最长的前缀 $z[k]$, 右端点 r

2. 如果 $i > r$: $z[i]$ 默认为0, 执行(2),(3)

```
void get_z() //z数组{
    z[1]=m;
    int l=0,r=0;
    for(int i=2;i<=m;i++){
        if(i<=r) z[i]=min(z[i-l+1],r-i+1);
        while(i+z[i]<=m&& b[i+z[i]]==b[z[i]+1])z[i]++;
        if(i+z[i]-1>r) r=i+z[i]-1,l=i;
    }
}
```

时间复杂度?



Z函数时间复杂度



西南大学附属中学
High School Affiliated to Southwest University

```
void get_z() //z数组{
    z[1]=m;
    int l=0,r=0;
    for(int i=2;i<=m;i++){
        if(i<=r) z[i]=min(z[i-l+1],r-i+1);
        while(i+z[i]<=m&& b[i+z[i]]==b[z[i]+1])z[i]++;
        if(i+z[i]-1>r) r=i+z[i]-1,l=i;
    }
}
```

因为 $O(1)$ 维护了

已知数据中的最右区间

i 在推进过程中严格递增

在内层while中产生的 $z[i]++$

最终会增长在 i 上

所以时间复杂度是线性的 $O(n)$

西 | 大 | 附 | 中 | 信 | 息 | 学 | 竞 | 赛 |
High School Affiliated to Southwest University



拓展KMP



西南大学附属中学
High School Affiliated to Southwest University

给定字符串 a, b ，求解 b 和 a 的每一个后缀的最长公共前缀数组 p 。 $|a|, |b| < 10^7$

例如： $a = \text{'aaaabaa'}$, $b = \text{'aaaaa'}$, $p = \{4, 3, 2, 1, 0, 2, 1\}$ 。

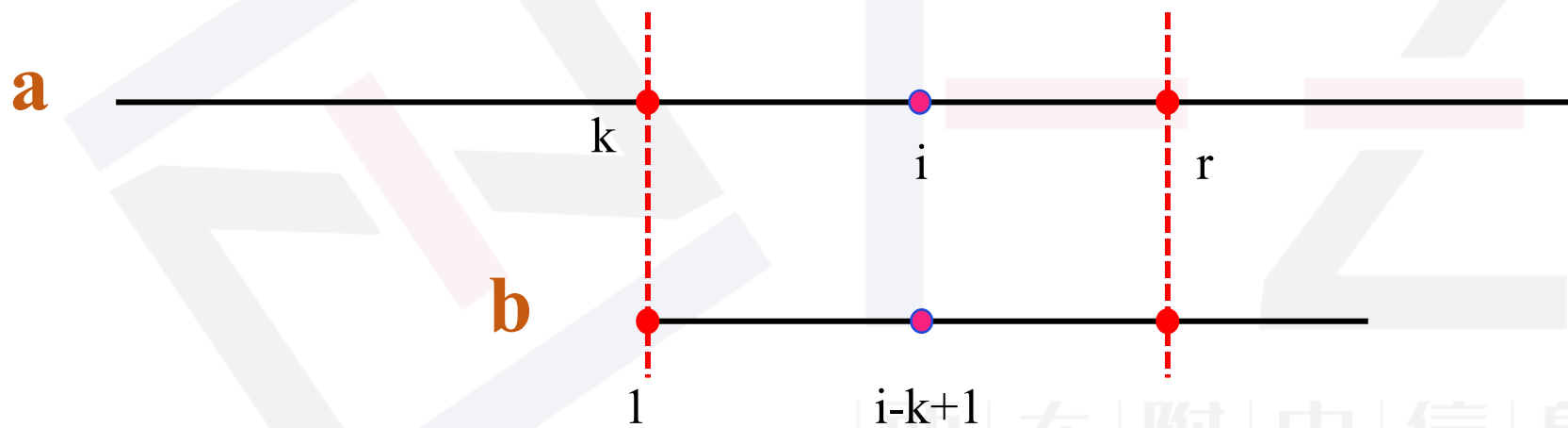


拓展KMP算法



西南大学附属中学
High School Affiliated to Southwest University

假设已经求解出 $P[1] \sim [i-1]$ ，求解 $P[i]$ 。
设当前最大的为 $P[k]$ ，对于右端点为 r 。



(1) 初始值:

$$P[i] = \min(z[i-k+1], r-i+1)$$

(2) 继续比较 $a[i+P[i]]$ 和 $b[P[i]+1]$

(3) 更新 $P[k]$ ，和 r



拓展KMP代码



西南大学附属中学
High School Affiliated to Southwest University

```
void exkmp()    //p数组
{
    int l=0,r=0;
    for(int i=1;i<=n;i++)
    {
        if(i<=r) p[i]=min(z[i-l+1],r-i+1);
        while(i+p[i]<=n&& a[i+p[i]]==b[p[i]+1]) p[i]++;
        if(i+p[i]-1>r) r=i+p[i]-1,l=i;
    }
}
```

西 | 大 | 附 | 中 | 信 | 息 | 学 | 竞 | 赛 |
High School Affiliated to Southwest University