# 线性代数在OI中的应用与题目讲解

北京大学
李超

# 线性方程组

$$\begin{cases} 3x_0 + 2x_1 - 5x_2 = -1 \\ 2x_0 - 4x_1 + x_2 = 1 \\ -x_0 - 5x_1 + x_2 = -6 \end{cases}$$

- 如何解n元线性方程组？
- 高斯消元法

# 矩阵

- 由n×m个数排成n行，m列的一张表称为一个n×m矩阵。
- 矩阵中的每一个数称为这个矩阵的一个元素。
- 矩阵的第i行与第j列交叉位置的元素称为矩阵的(i,j)元。
- 若一个矩阵的行与列相等，则称它为方阵。

$$\begin{cases} 3x_0 + 2x_1 - 5x_2 = -1 \\ 2x_0 - 4x_1 + x_2 = 1 \\ -x_0 - 5x_1 + x_2 = -6 \end{cases} \rightarrow \begin{bmatrix} 3 & 2 & -5 & -1 \\ 2 & -4 & 1 & 1 \\ -1 & -5 & 1 & -6 \end{bmatrix}$$

# 矩阵的加法与数量乘法

- 设A和B均为n×m矩阵，且$\forall i \forall j \; C_{i,j} = A_{i,j} + B_{i,j}$
- 则称C为A与B的和，记作C=A+B

- 设A为n×m矩阵，且$\forall i \forall j \; C_{i,j} = k \times A_{i,j}$
- 则称C为k与A的数量乘积，记作C=kA

# 矩阵乘法

- 设A为s×n矩阵，B为n×m矩阵，令C为s×m矩阵，且

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} \times b_{k,j}$$

- 则称C为A与B的乘积，记作C=AB
  - 只有左矩阵的列数与右矩阵的行数相同时才能相乘
  - 乘积矩阵的(i,j)元等于左矩阵的第i行与右矩阵的第j列对应元素乘积之和
  - 乘积矩阵的行数等于左矩阵的行数，列数等于右矩阵的列数

# 矩阵乘法

- 为什么我们要这样定义矩阵乘法？
- 在二维平面上考虑一个点A(x,y)
- 将线段OA逆时针旋转θ，则A点所对应的A'点的坐标是什么？

- 令 $r=\sqrt{x^2+y^2}$ ,α为∠AOx，则有
- x=rcosα , y=rsinα
- x'=rcos(α+θ) , y'=rsin(α+θ)
- x'=rcosαcosθ-rsinαsinθ=xcosθ-ysinθ
- y'=rsinαcosθ+rcosαsinθ=xsinθ+ycosθ

# 矩阵乘法

- x'=xcosθ-ysinθ
- y'=xsinθ+ycosθ
- 将方程的系数拿出来排成一张表
- $$A=\begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}$$
- 则该矩阵表示了转角为θ的旋转
- $$B=\begin{bmatrix}\cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi\end{bmatrix}$$
- 则该矩阵表示了转角为ϕ的旋转

# 矩阵乘法

- 考虑转动角度θ+ϕ

$$C=\begin{bmatrix}\cos(\theta+\varphi) & -\sin(\theta+\varphi)\\ \sin(\theta+\varphi) & \cos(\theta+\varphi)\end{bmatrix}$$

$$A=\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}\qquad B=\begin{bmatrix}\cos\varphi & -\sin\varphi\\ \sin\varphi & \cos\varphi\end{bmatrix}$$

$$c_{1,1}=a_{1,1}\times b_{1,1}+a_{1,2}\times b_{2,1}$$

$$c_{1,2}=a_{1,1}\times b_{1,2}+a_{1,2}\times b_{2,2}$$

$$c_{2,1}=a_{2,1}\times b_{1,1}+a_{2,2}\times b_{2,1}$$

$$c_{2,2}=a_{2,1}\times b_{1,2}+a_{2,2}\times b_{2,2}$$

# 矩阵乘法的性质

- 矩阵的乘法不适用交换律
  - 交换后的矩阵甚至不能相乘！即使能够相乘也不一定满足
- 矩阵的乘法适用结合律
  - 直接应用定义展开即可得到
- 矩阵的乘法适用左分配律与右分配律
  - $A(B+C)=AB+AC$
  - $(B+C)D=BD+CD$

- 矩阵乘法的结合律是许多题目解题的关键！

# Problem zero

- 幽鬼是游戏Dotb的一个英雄，有个被动技能“折射”，可以将受到的伤害的一定百分比平均反弹给周围的其他英雄
- 现在有N个幽鬼，每个幽鬼有不同折射率。给出每个幽鬼可以折射到的英雄与幽鬼的折射率，并在一开始对第K个幽鬼造成P点伤害，求最后每个幽鬼受到的伤害。
- 例：若1号幽鬼能反弹到2号幽鬼，折射率为60%，2号幽鬼不能反弹到1号幽鬼，且在最初给予1号幽鬼400点伤害，则最后1号幽鬼受到160点伤害，2号幽鬼受到240点伤害
- 折射进行无数次
- N<=100

# Problem zero Solution

- 设第i个幽鬼受到的总伤害为Xi
- 则对于每个幽鬼可以列出一个关于伤害的方程
- 共N个方程
- 解方程组即可
- O(N^3)

# Problem one

- 给定A[0]，A[1]
- A[i]=3×A[i-1]-2×A[i-2] (i>1)
- 求A[N]
- 答案模1000000007输出
- N<=10^9

# Problem one Solution

$$[a_i \ \ a_{i+1}]\times\begin{bmatrix}0 & -2\\ 1 & 3\end{bmatrix}=[a_{i+1} \ \ -2a_i+3a_{i+1}]=[a_{i+1} \ a_{i+2}]$$

- 即有 $[a_n \ a_{n+1}]=[a_0 \ a_1]\times\begin{bmatrix}0 & -2\\ 1 & 3\end{bmatrix}^n$

- 应用快速幂进行计算
- 快速幂可以在 $O(c^3\times\log N)$ 的时间复杂度内计算出一个c阶矩阵的N次幂

# 快速幂

```
// solve a^b
T power(T a,int b)
{
    T ret=1;
    while(b>0)
    {
        if (b%2==1)
            ret=ret*a;
        a=a*a;
        b/=2;
    }
    return ret;
}
```

# Problem one Solution

- 形如F[i]=A[1]*F[i-1]+A[2]*F[i-2]+...+A[c]*F[i-c]的方程均可以利用矩阵乘法和快速幂进行优化
- 直接求解的时间复杂度为$O(cN)$
- 使用矩阵乘法的时间复杂度为$O(c^3 \log N)$
- 适用c较小而N较大的情况

# Problem two

- 在一个古老的国家对数字有这样的规定：这个国家适用0,1,2,3,4,5,6,7,8,9这10个数字，但规定有P个整数对（xi,yi）(0<=xi,yi<=9)不能连续出现。
- 一个整数对(xi,yi)表示在一个数中，xi的后面不能紧跟着yi
- 现在我们要知道在这个规定下，第K小的正整数的前50位是多少。
- K<=2*10^9

# Problem two Solution

- 不妨设f[i,j]表示i位数，第1位为j的数的个数
- 则 $$f[i,j]=\sum_{k=0..9,\text{k可以在j后}} f[i-1,k]$$
- 注意到位数会很大
- 二分答案的位数t，从依次计算f[t]，f[t-1],...,f[t-50]，以确定每一位的值
- 时间复杂度 $O(c^3 \log^2 K)$,其中c在这里为10

# Problem three

- 给一张N个点的有向图
- 每条边的权值是1~5之间的整数
- 求从S走到T长度为K的路径条数
- 答案模1000003输出
- N<=10 K<=10^9

# Problem three Solution

- 有如下重要结论：
- 考虑一张无权的邻接矩阵A
- 设B=A^2，则Bij表示从i点到j点，走过两条边的路径条数
- 稍加扩展即可得到：
- 设C=A^k，则Cij表示从i点到j点，走过k条边的路径条数
- 这是路径长度为1的情况

- 由于路径长度为1~5之间的整数，可以拆点后再使用上述结论
- 时间复杂度 $O((5N)^3 \log K)$

# Problem four

- 给定N，R，Q，S
- 有N个关卡，初始有Q条命，且任意时刻最多只能有Q条命
- 每通过一个关卡，会得到u分和1条命，其中u=min（最近一次连续通过的关数,R）
- 若没有通过这个关卡，将失去一条命，并进入下一个关卡
- 若没有生命或N个关卡均已挑战过一次时，游戏结束，得到的分数为每关得到的分数的总和
- 每条命通过每个关卡的概率为p(0<=p<=1），原先最高分纪录为S
- 求当p至少为多少时，期望获得的总分能够超过最高分。
- 1<=N<=10^8 1<=R<=20 1<=Q<=5，输出保留6位小数

# Problem four

- 例如，当N=12，R=3，Q=2时
- 第一关：未通过 u=0 获得分数0 总分为0 剩余生命1
- 第二关：通过 获得分数1 总分为1 剩余生命2
- 第三关：通过 获得分数2 总分为3 剩余生命2
- 第四关：通过 获得分数3 总分为6 剩余生命2
- 第五关：通过 获得分数3 总分为9 剩余生命2
- 第六关：未通过 获得分数0 总分为9 剩余生命1
- 第七关：通过 获得分数1 总分为10 剩余生命2
- 第八关：未通过 获得分数0 总分为10 剩余生命1
- 第九关：未通过 获得分数0 总分为10 剩余生命0
- 游戏结束 总分为10

# Problem four Solution

- 容易想到二分答案p
- 令g[i，j，k]表示在第i关，u值为j，还剩k条生命的概率
- g[i+1，min(j+1,R)，min(k+1,Q)]+=g[i，j，k]×p
- g[i+1，0，k-1]+=g[i，j，k]×(1-p)
- Answer+=g[i，j，k]×p×(j+1)
- 将后两维展开为一维数组进行矩阵乘法
- 复杂度为 $O(T(RQ)^3 \log N)$，其中T为二分次数
- 这个复杂度仍然过大，需要继续优化

# Problem four Solution

- 我们发现，当u>=Q-1时，生命个数一定是Q
- 大量状态无用
- 去掉无用的状态后，矩阵的边长由PQ级别降为P+Q级别
- 再进行矩阵乘法，时间复杂度为
  $O(T(R+Q)^3 \log N)$，其中T为二分次数
- 这样可以通过所有的测试数据

- ☼即使使用时间复杂度较大的算法，加入常数优化也可以通过所有测试数据。

# Problem five

- 给定N个点的无权有向图
- 询问长度<K的环有几个
- 答案模M输出

- N<=35 K<=1000000

# Problem five Solution

- 令f[k，i，j]表示长度为k，从i走到j的方案数
- 则答案为 $\sum_{i=0}^{n-1}\sum_{j=2}^{k-1} f[j,i,i]$
- 暴力计算？$O(KN^3)$，难以接受
- 将矩阵直接展开进行优化？$O(N^6 \log K)$，同样不是优秀的算法

# Problem five Solution

- 将K按二进制位分为若干个子问题
- 容易算出2^i的方案数的和
- 容易算出一个长度k时的方案数
- 将其相乘，再将子问题的答案求和即可
- 时间复杂度$O(N^3 \log K)$

# Problem six

- 给定A[0],A[1]
- A[i]=3×A[i-1]-2×A[i-2] (i>1)
- 求A[N]
- 答案模10^100输出
- N<=10^9

# Problem six Solution

- 与problem one的区别在于模数为高精度数
- 直接矩阵乘法将涉及高精度数的相乘
- 实际上，我们可以改写矩阵

$$\begin{bmatrix} 0 & -2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} -2 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} -1 & -1 \\ 1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -2 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & -1 \\ 1 & 2 \end{bmatrix}$$

- 幂变得容易求得

# Problem six Solution

- 构造这样的矩阵，可以利用矩阵的特征值来完成
- 需要用到线性代数的一些知识

# 行列式

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} = \sum_{j_1 j_2 \cdots j_n} (-1)^{r(j_1 j_2 \cdots j_n)} a_{1 j_1} a_{2 j_2} \cdots a_{n j_n},$$

其中$j_1 j_2 \cdots j_n$是$n$元排列，

$r(j_1 j_2 \cdots j_n)$表示其中的逆序对个数

## N=2,3

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \times a_{22} - a_{12} \times a_{21}$$

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32}$$

$$- a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32}$$

# 上三角行列式

- 主对角线下方全为0的n阶行列式称为上三角行列式
- 上三角行列式的值为对角线上N个元素的乘积

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{vmatrix} = a_{11}a_{22}\cdots a_{nn}$$

# 行列式的计算

- 直接使用定义　　O(n!)
- 状态压缩DP　　O(2^n)
- 我们需要更好的算法！

# 行列式的性质

- 性质一
- 交换行列式的行与列，行列式值不变

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{vmatrix}$$

# 行列式的性质

- 性质二
- 行列式一行的因子可以提出

$$\begin{vmatrix} a_{11} & a_{12} & \cdots a_{1n} \\ ka_{21} & ka_{22} & \cdots ka_{2n} \\ \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots a_{nn} \end{vmatrix} = k \begin{vmatrix} a_{11} & a_{21} & \cdots a_{n1} \\ a_{12} & a_{22} & \cdots a_{n2} \\ \vdots & \vdots & \vdots \\ a_{1n} & a_{2n} & \cdots a_{nn} \end{vmatrix}$$

# 行列式的性质

- 性质三
- 两行互换，行列式反号

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} = \begin{vmatrix} a_{21} & a_{22} & \cdots & a_{2n} \\ a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{vmatrix}$$

# 行列式的性质

- 性质四
- 将一行的倍数加到另一行上，行列式的值不变

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21}+ka_{11} & a_{22}+ka_{12} & \cdots & a_{2n}+ka_{1n} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{vmatrix}$$

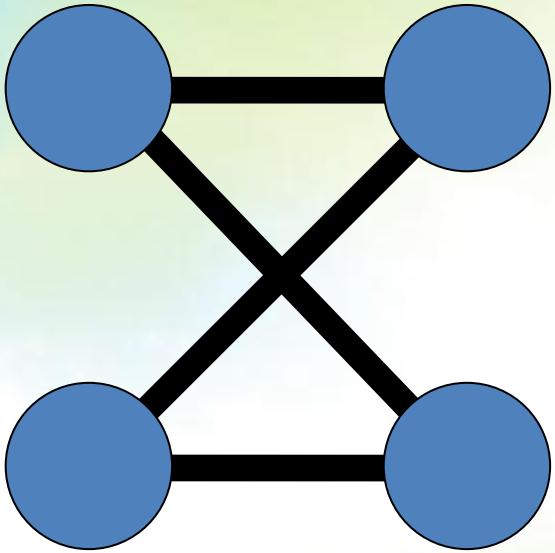
# 行列式的计算

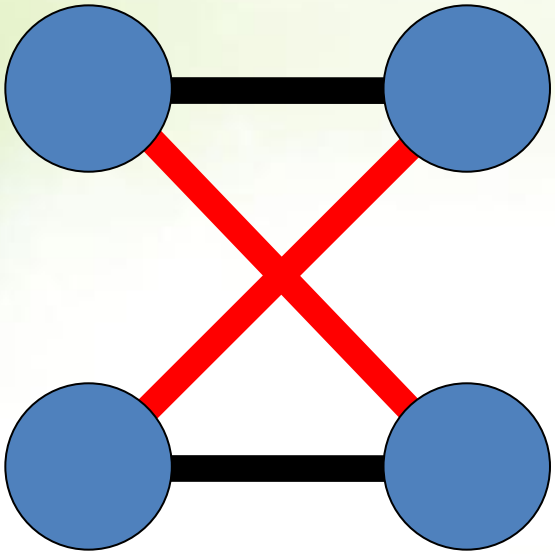
- 利用性质三和性质四
- 我们可以用类似高斯消元的方法
- 将一个普通的行列式通过初等行变换转化为一个上三角行列式进行求值
- O(N^3)

# Problem seven

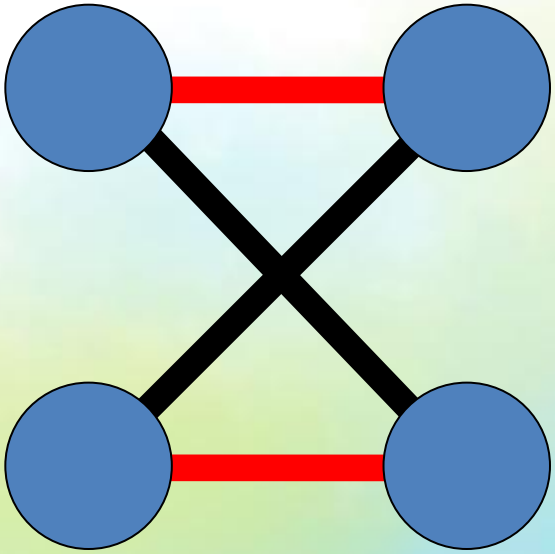
- 给定一张N个点M条边的有向无环图
- 保证有恰好K个点入度为0，我们称之为源；K个点出度为0，我们称之为汇，将源与汇的K个点分别标号为1..K
- 从图中选出K条没有公共点的路径，每条路径经过一个源与一个汇。不妨设从源i出发的边到达了汇Ti。
- 对于任意i,j，若有i<j且Ti>Tj，我们称之为一个逆序对。
- 如果所有路径中的逆序对个数是偶数，答案增加1，否则答案减少1
- 求答案模一个素数p后的值
- N<=600，M<=10^5，2<=p<=1000000007

# Problem seven





-1



1

# Problem seven Solution

- 注意到相交的路径不影响答案
- e.g.
- 1->2->3
- 4->2->5
- 与
- 1->2->5
- 4->2->3
- 成对出现，且一个答案为1，一个答案为-1
- 因此不需要考虑路径相交的情况

# Problem seven Solution

- 令A[i,j]表示从第i个源到第j个汇的路径数
- 简单dp即可得到A
- 易发现答案即为A的行列式值
- O(N^3+NM)

# 积和式

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} = \sum_{j_1 j_2 \cdots j_n} a_{1 j_1} a_{2 j_2} \cdots a_{n j_n},$$

其中$j_1 j_2 \cdots j_n$是$n$元排列

- 行列式的无系数版本
- 不容易计算
- 对同一个方阵，积和式的奇偶性与行列式相同

# 余子式

- 将行列式A划去第i行与第j列，得到的N-1阶行列式称为余子式，通常记作Mij

- 令 $A_{ij} = (-1)^{i+j} M_{ij}$ ，我们称Aij为A的(i,j)元的代数余子式

# 行列式按一行展开

$$|A| = \sum_{j=1}^{n} a_{ij} A_{ij}$$

- 思考：fibonacci数列的第N项如何用一个元素仅含0,1,-1的N阶行列式表示？

# 矩阵的逆

- 若 $A \times B = I$，则我们称B是A的逆，记作$B = A^{-1}$
- 行列式值不为0的n阶矩阵必有逆矩阵
- 如何求逆矩阵？
- 将A通过初等行变换变为单位矩阵，再将单位矩阵按同样顺序做相同的初等行变换，则得到的矩阵即为$A^{-1}$

# 矩阵的相似

- 设A，B为n级矩阵
- 若存在矩阵P，使$A = P^{-1}BP$，则称A与B相似

- 为什么要研究相似？
- 可以优化一部分矩阵的计算
- $A^m = (P^{-1}BP)^m = P^{-1}B^mP$

# 矩阵的特征值与特征向量

- 设A是n级矩阵
- 若有非零列向量α，使Aα=λα，且λ∈K
- 则称λ是A的一个特征值，α是A的属于特征值λ的一个特征向量

- λ为A的特征值的充要条件：$|\lambda I - A| = 0$

# Problem eight

- 一个培养皿分为N个区域
- 一开始第1个区域有一个细胞
- 在每个单位时间，第i个区域的细胞，每个细胞会分裂成为K个细胞，且进入第j个区域的细胞有Aij个。
- 细胞只会进入区域编号大于等于当前区域编号的区域，即 $\forall i, \forall j < i, A_{i,j} = 0$
- 数据保证$\forall i \forall j\ A_{i,i} \neq A_{j,j}$
- 求T个时刻后细胞的数目，答案模p输出
- N<=100

# Problem eight Solution

- 直接矩阵乘法 O(N^3logT)
- 注意到是上三角矩阵，且对角元两两不同
- 特征值即为对角线上的N个元素
- 我们可以在O(N^3)时间内求出所有特征值对应的特征向量
- 利用矩阵的相似将A写成三个矩阵相乘的形式，只需要对对角的元素使用快速幂
- 时间复杂度O(N^3)
- 需要高精度时时间复杂度大大降低

# Scrivener

- 维护一个字符序列（初始为空），有三种操作
  - 在当前序列末尾加入一个字符
  - 撤销之前的Pi个操作，撤销操作也可以被撤销
  - 询问当前序列的第Qi个字符
- 要求在线
- 操作数量<=1000000

# Scrivener Solution

- 所有操作形成一棵树结构
- 记录每个操作结束后的终止节点位置
  - 操作一：在当前节点下新增一个节点
  - 操作二：跳跃到撤销操作后对应的节点位置
  - 操作三：在操作一时维护每个节点的2^k的祖先节点，由于知道当前序列的长度，可以知道要求输出的字符是当前序列的倒数第几个字符，利用维护的倍增数组即可
- 操作一、三时间复杂度为O(logN)
- 操作二时间复杂度为O(1)
- 空间复杂度为O(NlogN)

# Tournament

- N名选手参加一个竞赛，每名选手有一个唯一的不与其他选手相同的能力值
- N名选手开始时从左到右站成一排，进行C场比赛
- 每场比赛选择目前存活的第Li到Ri个选手进行竞赛，能力值最高的选手获胜，其他选手则退出比赛。数据保证C场比赛之后只剩下1名选手。
- 现在有一名能力值为R的选手迟到，其他选手已经安排好了位置。要求将这名选手插进某个位置，使得这名选手获胜的场数最多。若有多解，选择最靠左的位置。
- N<=100000

# Tournament Solution

- 利用线段树维护Li,Ri在原先中序列的位置
  - 区间修改
  - 查询第K个存在的元素
- 所有的比赛形成了一棵树结构
- 枚举新骑士插入的位置，则任务变为寻找该点向上的深度最小的最大值不超过新骑士值的节点
- 树上倍增+区间查询最大值
- O(NlogN)

# Supper

- 有一个大小为K的调色板，共N(N<=10^5)种颜色。初始时调色板上有0~K-1的颜色。
- 现在要向调色板上加入N次颜色。每次加入时：
  - 若该颜色调色板上已有，则忽略这次操作
  - 否则删去调色板上的一种颜色，并加入这种新颜色
- 需要写两个程序，第一个程序可以知道K,N以及N次加入的颜色分别是什么。
- 第一个程序需要向第二个程序传输一个长度不超过M的二进制串，第二个程序只知道K，N和二进制串。之后每读入一个颜色，就要立刻返回当前的操作。
- 第二个程序所进行的删去操作不能比最优方案的删去操作数量多。满分算法需要保证二进制串长M<=2N。

# Supper Solution

- 最优方案可以通过O(NlogK)时间内求得
- 想法一：将数据直接压缩传输
- 想法二：记录下每次操作，传输操作序列
- 想法三：记录下最优操作的等价操作，传输操作序列

# Supper Solution

- 首先得到一个最优方案
- 我们不妨在序列中添加K个元素，令颜色0~K-1在序列的最前端出现一次
- 则对于N+K次操作，我们找到每个颜色曾被删除过的位置，并将该位置之前的最晚一次出现的该颜色进行标记
- 在求答案时，若需要更新颜色，则在之前标记的位置中任意取一个对应的颜色，然后删去该标记即可
- O(NlogN)
- 最优方案正确性证明
- 操作方案等价性证明

# 谢谢大家