数学专题选讲 (同余, 线性代数基础)

2025 CEIT 算法集训队暑期集训 #2

Orange

沈阳师范大学，人工智能学院，CEIT LAB

2025/07/28

目录

定义

a | b：b 是 a 的倍数/a 是 b 的因数
a % b：记为 a $\bmod$ b，即 a 对 b 的余数
(a,b)：记为 $\gcd(a,b)$，即 ab 的最大公约数
[a,b]：记为 $\mathrm{lcm}(a,b)$，即 ab 的最小公倍数
ab 互质：即 a 和 b 没有除了 1 之外的公约数，互质的充要条件是 $\gcd(a,b)=1$。

欧几里得算法

```
int gcd(int a,int b)
    return b ? gcd(b, a % b) : a;
```

欧几里得算法，也叫做辗转相除法，可以在 $O(\log \min(a, b))$ 的复杂度内求出 (a,b)。实现原理不再赘述 (允许 x 和 y 为负数)。此外，我们这里再引入一个重要的定理：

裴蜀定理

对于 a,b，一定存在解 x,y 满足如下不定方程：

$$ax + by = (a, b)$$

即 a 和 b 的若干倍之和一定能凑出其最大公约数。

那么，应该如何求解出上述不定方程的一组解呢？

拓展欧几里得算法

我们不妨将目光放回欧几里得算法。欧几里得算法是一个经典的递归结构。
让我们将目光放到欧几里得算法的递归终点：当 b=0 时，此时，易知 (a,b)=a，所以求解 ax+by=(a,b) 是容易的，x=1，y=0 显然是一组解。
此时，我们再来考虑 $b \neq 0$ 的一般情况，由欧几里得算法可知，(a,b)=(b,a%b)。因此，我们考虑在 (b,a%b) 回溯时求解，即此时我们已经得到了一组解 x',y'，使得 bx'+(a%b)y'=(b,a%b)。
我们不妨代入 (a,b)=(b,a%b)，此时有：

$$bx' + (a\%b)y' = (a, b)$$

而此时又有 $a\%b = a - \lfloor \frac{a}{b} \rfloor b$，我们将其带入上式：

$$bx' + (a - \lfloor \frac{a}{b} \rfloor b)y' = (a, b)$$

拓展欧几里得算法

展开上式：

$$bx' + ay' - \lfloor \frac{a}{b} \rfloor by' = (a, b)$$

整理后可得：

$$ay' + b(x' - \lfloor \frac{a}{b} \rfloor y') = (a, b)$$

很显然，$x = y', y = x' - \lfloor \frac{a}{b} \rfloor y'$ 是对于当前 a 和 b 的一组解。至此，我们得出了递归的关系，因此我们可以利用欧几里得算法求出 ax+by=(a,b) 的**一组特解**。
我们把上述求解裴蜀定理不定方程的算法称之为**拓展欧几里得算法**。

```
int exgcd(int a,int b, int &x, int &y){
    if(b == 0) return x = 1, y = 0, a;
    int r = exgcd(b, a % b, x, y);
    tie(x, y) = make_tuple(y, x - (a/b)*y);
    return r;
}
```

一个题目

青蛙的约会

在一个 0-index，长度为 L 的环上，a 和 b 分别位于坐标 x 和 y 处，他们每次回同时向前跳长度为 m 和 n 的单位，求他们第一次位于同一个坐标点的时候跳了多少次。无解输出 Impossible 。
数据范围:$x, y, m, n \leq 2e9, L \leq 2.1e9$

样例输入: 1 2 3 4 5，样例输出： 4

思路

由于是一个环，因此假设 a 和 b 之间的距离相差刚好是 L 的倍数，则认为他们重合。
不妨假设 k 次之后，他们完成重合，那么 a 的距离就是 x+km，b 的距离就是 y+kn，此时他们相差 $p \times L$，可以列出下式：

$$x + km - y - kn = pL$$

此时 k 和 p 为待求未知数，整理上式：

$$k(m - n) - pL = y - x$$

这很似乎就是我们裴蜀定理方程的格式，方程有解等价于 y-x 是 (m-n,L) 的倍数。但是需要注意的是，使用 exgcd 求出的是方程等于 (m-n,L) 的解而不一定是 y-x，因此还需要对方程两边同时放大 $\frac{y-x}{(m-n,L)}$ 倍。而且求出的解 k 并不一定满足条件。

裴蜀定理方程的解

事实上，裴蜀定理本质上是阐述了：对于不定方程 ax+by=R(其中 x 和 y 待求)，只有当 R 是 (a,b) 的倍数时，方程才可能有解。而 exgcd 则是刚好求出 ax+by=(a,b) 的一组解，此时如果要求对 R 的解，还需要将求出的 x 和 y 等比放大。

裴蜀定理方程的解的形式

假设，我们现在得到了一组特解 x 和 y，满足 ax+by=R，不妨思考其他解应该是何种形式？以下直接给出其通解的结构：

$$\begin{cases} x = x_0 + \dfrac{b}{(a,b)}t, \\ y = y_0 - \dfrac{a}{(a,b)}t, \end{cases} \qquad t = 0, \pm 1, \pm 2, \cdots.$$

回到上面的题目，我们容易求出其一个解 k_0，但是其很可能不满足题目要求，因为题目要求最小化其跳跃次数，而且跳跃次数显然非负，因此需要利用特解的形式将 k_0 转化成一个最小的非负解。至此，上题结束。

同余的意义

同余

假设存在 3 个整数 a，b，m，其中 m>0，若 a%m=b%m 成立，我们称 a 和 b 对 m 同余。记作：

$$a \equiv b \pmod{m}$$

在同余的前提下，我们认为**所有对 m 余数相同的数都相等**。例如在 m=4 的情况下，我们可以认为 1，5，9...1+km... 均相等。因此，可以得出，我们最多只会有 m 种本质不同的数，即 0,1,2...m-1。同余有很多非常方便的性质：

1. $a + b \equiv c$, 即可以认为，任何与 a 等价的数 a' 和任何与 b 等价的数 b'，其和 a'+b' 一定与 c 等价。
2. $a \times b \equiv c$, 同理。

用上述性质，我们可以将一些原本很大的数表示到很小的范围内，这也是算法竞赛中常用的表示大数的技巧。

逆元

定义

在同余 m 的前提下，对于一个整数 a，若存在一个整数 b，满足 $ab \equiv 1$，我们就称 b 为 a 的逆元，记为 a^{-1}。

逆元一定存在吗？如何求解？

我们观察逆元的定义式：$ab \equiv 1 \pmod{m}$，上一页对同余的定义，我们知道对于一个数 p，所有与他等价的数我们可以写成 p + km，其中 k 为任意整数。考虑对 1 做出变形，即 1+km，那么上面的同余式可以改写成下面的等式：

$$ab = 1 + km$$

考虑做出变形：

$$ab - km = 1$$

这个形式不就是我们上面的裴蜀定理方程吗？其中 b 和 k 是待求的未知数，回忆裴蜀定理，不难发现，只有当 (a,m)=1，即两数互质，方程才有解，且其解可以使用拓展欧几里得算法求解。

逆元

因此，我们得出了逆元存在的条件：**在模 m 意义下，a 存在逆元 a^{-1}，当且仅当 a 和 m 互质**。

不妨考虑特殊情况：假设 m 为质数，此时对于所有小于 m 的数，其在模 m 意义下均存在逆元。

事实上，对于 m 为质数，我们有更多的数学性质可以使用。

欧拉函数

在介绍更加深入的同余性质之前，我们有必要引入欧拉函数。

定义

欧拉函数 $\phi(x)$ 等于 1 到 x-1 的所有数中，与 x 互质的数的个数。

由于这里仅仅只是引入而不是详细介绍，我们直接给出欧拉函数的计算方法，感兴趣的可以自行查阅相关资料。
钦定 $x = p_1^{c_1} p_2^{c_2} ... p_n^{c_n}$，则根据容斥原理变形后可得：

$$\begin{aligned}\phi(x) &= n(1-\frac{1}{p_1})(1-\frac{1}{p_2})...(1-\frac{1}{p_n}) \\ &= (p_1-1)p_1^{c_1-1} \times (p_2-1)p_2^{c_2-1} \times ... \times (p_n-1)p_n^{c_n-1}\end{aligned}$$

观察上述形式，我们求解欧拉函数可以直接使用质因数分解完成。

逆元

不扯远了，我们回到原来的话题。引入欧拉函数后，我们接着引入欧拉定理：

欧拉定理

若 (a,m)=1，则 $a^{\phi(m)} \equiv 1 \pmod{m}$。

对于欧拉定理 $a^{\phi(m)} \equiv 1 \pmod{m}$，我们展开可得：
$a \times a^{\phi(m)-1} \equiv 1 \pmod{m}$，即可以得出 $a^{\phi(m)-1}$ 就是 a 的一个逆元。
此时，考虑如果 m 是质数，那么很显然 $\phi(m) = m - 1$，此时就有：

$$a^{m-1} \equiv 1 \pmod{m}$$

这就是我们的**费马小定理**，将其变形之后可以得出：$a^{-1} \equiv a^{m=2} \pmod{m}$。也就是一个数 x 的 m-2 次方在模 m 意义下一定就是 x 的逆元。上述过程可以用**快速幂**求出。

求逆元

求解线性同余方程

求解关于 x 的同余方程 $ax \equiv 1 \pmod{b}$ 的最小正整数解，若无解输出-1。
数据范围:$a, b \leq 2e9$

根据同余的定义，我们不妨将同余式写成等式：ax=1+kb，k 为任意正整数。此时再作变形，得到 ax-kb=1，很显然，这又变回了我们裴蜀定理方程的形式，直接使用 exgcd 求解即可，再根据方程解的结构，对其做等值变形，即可得到最终答案。当 (a,b)>1 时无解。
此外，上面的题目也可以使用欧拉定理求出，当 (a,m)=1 时，$a^{\phi(m)-1}$ 就是一个答案。

线性逆元

整数的逆元 II

给定整数 n，求出 1 到 n 所有数在 mod p 意义下的逆元。
数据范围：$n \leq 1e7, n < p < 2e9$，保证 p 为质数。

线性逆元

很显然，当 n 大到无法容忍 $O(n \log n)$ 的算法时，无论是快速幂还是拓展欧几里得算法都无法完成逆元的求解。
不过好在，我们可以从递推的性质上做文章。不妨考虑待求数 i，对于模数 p，显然满足：

$$\lfloor \frac{p}{i} \rfloor i + p\%i = p$$

观察等式的左边，其显然满足在模 p 意义下同余 0，因此可以将等式改写成：

$$\lfloor \frac{p}{i} \rfloor i + p\%i \equiv 0 \pmod{p}$$

移项后可得：

$$\lfloor \frac{p}{i} \rfloor i \equiv -p\%i \pmod{p}$$

线性逆元

我们此时左右两边均乘上 i^{-1} 和 $(p\%i)^{-1}$：

$$\lfloor \frac{p}{i} \rfloor (p\%i)^{-1} \equiv -i^{-1} \pmod{p}$$

最后，简单整理可得：

$$i^{-1} \equiv -\lfloor \frac{p}{i} \rfloor (p\%i)^{-1} \pmod{p}$$

此时，我们就得到了 i 的逆元递推式，通过 p%i 的逆元，就可以计算出 i 的逆元。(因为我们从小往大进行地推时，p%i $<$ i，因此满足拓扑关系)。此时考虑边界情况，显然有 $1^{-1} \equiv 1 \pmod{p}$。至此，我们只需要从 2 开始，即可在线性的复杂度内，利用递推式在 O(n) 复杂度内求解 1 到 n 的所有逆元。

求解同余方程组

求解同余方程组

求如下方程组的解:

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \quad \cdots \\ x \equiv a_k \pmod{m_k} \end{cases}$$

求解同余方程组

不妨考虑对其中的前两项进行讨论：很显然，我们可以改写同余式为等式，此时有：$x = a_1 + y_1 m_1 = a_2 + y_2 m_2$。我们重点讨论等式右边的 $a_1 + y_1 m_1 = a_2 + y_2 m_2$，将其改写后可得：

$$-y_1 m_1 + y_2 m_2 = a_1 - a_2$$

这个方程正是裴蜀方程，其有解当且仅当 $(m_1, m_2) | a_1 - a_2$，同时它的解形式是已知的。钦定其有解，那么我们假设用 exgcd 求出一个 y_1 的特解 y_1^*，那么其所有解的形式为：

$$y_1 = y_1^* + k\frac{m_2}{(m_1, m_2)}$$

将其代入 $x = y_1 m_1 + a_1$：

$$x = m_1(y_1^* + k\frac{m_2}{(m_1, m_2)}) + a_1$$

求解同余方程组

展开上式：

$$x = m_1 y_1^* + k\frac{m_1 m_2}{(m_1, m_2)} + a_1$$

很显然，$m_1 y_1^* + a_1$ 这部分是固定的常数，我们将其记为 a'，而 $\frac{m_1 m_2}{(m_1, m_2)} = [m_1 m_2]$ 也是一个常数，将其记为 m'，此时回到原式：

$$x = km' + a'$$

当前形式的 x 一定满足为上面两个方程的解，且通过观察，这个形式不就是我们的同余式的等式形式吗？

$$x \equiv a' \pmod{m'}$$

这就是说，我们通过上面联立求解再代入的方法，可以成功将两个方程合并一个方程，且这种方法任然可以对我们新得出的方程与剩下的方程使用，直到剩下最后一个方程时，方程右边的常数就是我们的最终解。最终，通过 k-1 次 exgcd，我们顺利求出了 k 个线性同余方程组的解，时间复杂度 $O(n \log n)$。

一个例题

Biorhythms

人有三个周期，身体周期 23 天，情绪周期 28 天，智力周期 33 天，每个周期都有一个峰值。现在给定一个人的三个周期的峰值日期 p,e,i，求其在 d 天后的第一个三重峰。

思路

很显然，题目就是要求同时满足如下方程的解：

$$x \equiv p \pmod{2}3$$

$$x \equiv e \pmod{2}8$$

$$x \equiv i \pmod{3}3$$

直接使用上述提到的同余方程组的解法即可，注意，最后我们求出一个解后，需要其满足大于 d，加上 k 倍的 [23,28,33] 即可。

中国剩余定理

很显然，虽然我们上面不断代入再消元的方法时间复杂度已经非常优秀，但实际上，在应用时，其很容易出错，并且编码较为复杂。

早在南北朝时期，中国的数学家孙子就在《孙子算经》中给出了线性同余方程组的构造解，这种方法被称为孙子定理，也叫做中国剩余定理。

中国剩余定理能够在同样的时间复杂度内构造出一组关于给定线性同余方程组的解，且编码简单，易于实现，常用于算法竞赛中。

中国剩余定理

假设 $m_1, m_2 \ldots m_k$ 两两互质，对于线性同余方程组：

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \quad \ldots \\ x \equiv a_k \pmod{m_k} \end{cases}$$

其解为：

$$x \equiv \sum_{i=1}^{k} M_i' M_i a_i \pmod{M}$$

其中，$M = m_1 m_2 \ldots m_k$，$M_i = \frac{M}{m_i}$，$M_i' M_i \equiv 1 \pmod{m_i}$。

矩阵的定义

定义

矩阵是一个由数字组成的矩形数组，通常用大写字母表示。一个 $m \times n$ 矩阵 A 可以写成：

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

- a_{ij} 表示第 i 行第 j 列的元素
- 在算法竞赛中，矩阵通常用二维数组表示
- 方阵：行数等于列数的矩阵（$n \times n$）
- 矩阵不强制要求是方形，三行四列矩阵是合法的 xD

矩阵运算

矩阵加法

两个同型矩阵对应元素相加：$(A+B)_{ij} = A_{ij} + B_{ij}$

矩阵乘法

设 A 为 $m \times p$ 矩阵，B 为 $p \times n$ 矩阵，则 $C = AB$ 为 $m \times n$ 矩阵：

$$C_{ij} = \sum_{k=1}^{p} A_{ik} \cdot B_{kj}$$

注意

- 矩阵乘法不满足交换律：$AB \neq BA$
- 矩阵乘法满足结合律：$(AB)C = A(BC)$
- 矩阵乘法满足左分配律和右分配律:$(A+B)C = AC + AB$。

行列式

定义

n 阶方阵 A 的行列式定义为：

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}$$

重要性质

- 交换两行，行列式变号
- 某行乘以常数 k，行列式乘以 k
- 某行加上另一行的倍数，行列式不变
- 上三角矩阵的行列式等于主对角线元素的乘积

求解线性方程组

例

求解方程组：

$$x + 2y + 3z = 6 \tag{1}$$
$$2x + 3y + z = 7 \tag{2}$$
$$3x + y + 2z = 8 \tag{3}$$

线性代数为我们提供了一种很方便的解方程组的方法，根据矩阵乘法，我们很容易可以将其改写成矩阵形式。

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ 7 \\ 8 \end{pmatrix}$$

矩阵的初等行变换

矩阵的初等行变换的核心目标是通过特定操作简化矩阵（如上三角/行最简形），用于求解方程组、求秩、求行列式等。
三种基本操作:

- 行交换: 将矩阵的第 i 行和第 j 行互换。
- 行数乘: 用一个**非零常数** k 乘以矩阵的第 i 行。
- 行倍加: 将第 j 行的 k 倍加到第 i 行上（k 无限制）。

以上三种行变换，不改变矩阵所表示线性方程组的解集，这是高斯消元求解方程组的基础。

高斯消元

增广矩阵

增广矩阵就是系数矩阵和数值矩阵”拼”起来的矩阵。

高斯消元

高斯消元本质上是利用矩阵的初等行变换，将增广矩阵化为行阶梯形式，求解线性方程组或者行列式。

实现步骤

1. 找到绝对值最大的一行 (保证精度)
2. 将该行与第一行交换
3. 利用数乘，将该行第一个数变为 1
4. 将最上面一行下面所有行的第 c 列变成 0

线性方程组的解

高斯消元一般会让我们得到一个阶梯型矩阵，此时我们还需要求出每个未知数的具体解，以及判断解是否存在。

矩阵的秩

我们一般用 r(A) 表示 A 矩阵的秩，简单理解，矩阵的秩就是矩阵 A 通过初等行变换变成阶梯型矩阵后，非全 0 行的数量。

假设我们通过高斯消元求出的矩阵满足：

- r(A)=r(A|C)= 未知数个数: 说明 A 矩阵代表的线性方程组存在唯一解。
- r(A)=r(A|C)< 未知数个数: 说明 A 矩阵代表的线性方程组存在无穷多解。
- r(A) < r(A|C): 说明 A 矩阵代表的线性方程组无解。

求解异或方程组

Extended Lights Out

有一个 5*6 的开关阵列和灯，如果按下 (i,j) 个开关，则它对应灯和它四周相邻的灯的状态都会反转（亮变暗，暗变亮）。给出灯的初始状态，问是否可能使得所有灯都熄灭，如果可能给出一个方案。

异或方程组

异或运算性质

- $a \oplus a = 0$
- $a \oplus 0 = a$
- 异或满足交换律和结合律

例

求解异或方程组：

$$x_1 \oplus x_2 \oplus x_3 = 1 \tag{1}$$
$$x_1 \oplus x_3 = 0 \tag{2}$$
$$x_2 \oplus x_3 = 1 \tag{3}$$

关键思想

将异或运算看作模 2 意义下的加法，用高斯消元求解

矩阵快速幂

问题

计算矩阵 A 的 n 次幂：A^n

朴素算法

连续乘 $n-1$ 次，时间复杂度 $O(n \cdot k^3)$，其中 k 是矩阵维数

快速幂思想

利用二进制表示：

$$A^n = A^{b_k \cdot 2^k + b_{k-1} \cdot 2^{k-1} + \cdots + b_1 \cdot 2 + b_0}$$

$$= A^{b_k \cdot 2^k} \cdot A^{b_{k-1} \cdot 2^{k-1}} \cdots A^{b_1 \cdot 2} \cdot A^{b_0}$$

因此，利用快速幂的思想，我们可以将计算矩阵的 n 次幂优化到 $O(\log n \cdot k^3)$。

斐波那契数列

递推关系

$$F_n = F_{n-1} + F_{n-2}, \quad F_0 = 0, F_1 = 1$$

矩阵表示

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix}$$

推广

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$$

复杂度优化

从 $O(n)$ 优化到 $O(\log n)$

一般线性递推

问题

给定递推关系：

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

求 a_n 的值

矩阵构造

$$\begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_{n-k+1} \end{pmatrix} = \begin{pmatrix} c_1 & c_2 & \cdots & c_k \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} a_{n-1} \\ a_{n-2} \\ \vdots \\ a_{n-k} \end{pmatrix}$$

一个题目

Magic Gems

求长度为 N 的 01 串的数量，满足所有极长的全 0 子串长度都是 M 的倍数。

数据范围: $1 \leq N \leq 10^{18}, 2 \leq M \leq 100$

思路

首先不考虑数据范围，钦定 $f(n)$ 表示长度为 n 的满足条件的序列数量。由于其中所有极长的 0 子串都为 m 的倍数，事实上我们每次考虑往后填数都只有两种转移：

- 从 f[i-1]，放一个 1
- 从 f[i-m]，放 m 个 0

因此，转移方程就是 f[n]=f[n-1]+f[n-m]。很显然，n 非常大，不允许我们在线性的复杂度内通过本题。考虑构造矩阵来加速递推。

$$\begin{pmatrix} f(n) \\ f(n-1) \\ \vdots \\ f(n-m+1) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \begin{pmatrix} f(n-1) \\ f(n-2) \\ \vdots \\ f(n-m) \end{pmatrix}$$

另一个题目

Magic Gems

在 n*m 的方格上，给每个格子染成黑白两种颜色。要求左右相邻两格不能同为白色，且相邻两列不能全为黑色。求方案数
数据范围: $1 \leq N \leq 10^{18}, 1 \leq M \leq 5$。

思路

首先，对于每一列来说，方案数非常少，最多只有 2^m 种方案。考虑枚举所有方案之间的二元关系 (笛卡尔积)，假设状态 j 能够放到状态 i 的右侧，我们称 i 可以转移到 j，此时我们可以求出所有的转移关系。设 f(n,i) 表示第 i 列状态为 i 的方案数，其转移方程为：

$$f(n,i) = \sum_{(i,j)\in st_i \times st_j} f(n-1,j)$$

构建一个 $2^m \times 2^m$ 大小的系数矩阵进行递推即可。

一些常见的递推形式

1. 带系数的递推式

$$\begin{aligned} f[n] &= \sum_{i=1}^{k} a_i f[n-i] \\ &= a_1 f[n-1] + a_2 f[n-2] + \cdots + a_k f[n-k] \end{aligned}$$

考虑构造如下矩阵：

$$\begin{pmatrix} f[n] \\ f[n-1] \\ \vdots \\ f[n-k+1] \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_3 & \cdots & a_k \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \begin{pmatrix} f[n-1] \\ f[n-2] \\ f[n-3] \\ \vdots \\ f[n-k] \end{pmatrix}$$

一些常见的递推形式

2. 带系数以及常数项的递推式

$$\begin{aligned} f[n] &= \sum_{i=1}^{k} a_i f[n-i] + c \\ &= a_1 f[n-1] + a_2 f[n-2] + \cdots + a_k f[n-k] + c \end{aligned}$$

考虑构造如下矩阵：

$$\begin{pmatrix} f[n] \\ f[n-1] \\ \vdots \\ f[n-k+1] \\ c \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & \cdots & a_k & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} f[n-1] \\ f[n-2] \\ \vdots \\ f[n-k] \\ c \end{pmatrix}$$

一些常见的递推形式

3. 多项一起递推/递推需要借助中间辅助量转移的

$$\begin{cases} f[n] = a_{11}f(n-1) + a_{12}g(n-1) \\ g[n] = a_{21}f(n-1) + a_{22}g(n-1) \end{cases}$$

考虑构造如下矩阵：

$$\begin{pmatrix} f[n] \\ g[n] \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} f[n-1] \\ g[n-1] \end{pmatrix}$$

注意：$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ 必须是常数矩阵。

引入

斐波那契询问

维护一个序列 a_i，支持两种操作：

- 给定一个区间 [l,r] 和一个数 x，使得 a[l...r]+x。
- 给定一个区间 [l,r]，询问 $\sum_{i=1}^{r} f(a_i) \pmod{10^9+7}$，其中 f(i) 表示斐波那契数列的第 i 项

数据范围：$n, m \leq 10^5, a_i, x \leq 10^9$

矩阵乘法带来的一些启发

首先，对于求解斐波那契数列的求解，$f(a_i)$ 显然是好求的。钦定 $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, F(1) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$，很显然:

$$f(a_i) = A^{a_i} F(1)$$

考虑带修:

$$f(a_i + x) = A^{a_i + x} F(1) = A^x F(a_i)$$

不难发现，对单个位置的带修也是很好操作的。
再考虑区间修改:

$$\sum_{i=l}^{r} F(a_i + x) = \sum_{i=l}^{r} A^x F(a_i) = A^x \sum_{i=l}^{r} F(a_i)$$

到这一步，相信你已经完全知道了接下来要发生什么。

线段树维护带修的动态规划 (DDP)

不妨考虑，对序列的每个位置都维护一个 $F(a_i)$，修改部分，可以直接采用线段树来维护，将修改的 x 视为 A^x，并作为懒标记下放到各个节点，每个节点维护一个区间内所有 $F(a_i)$ 之和 (根据矩阵的加法)，即可完成本题。

另一个简单题

区间 sin 和

维护一个序列 a_i，支持两种操作：

- 给定一个区间 [l,r] 和一个数 x，使得 a[l...r]+x。
- 给定一个区间 [l,r]，询问 $\sum_{i=1}^{r} \sin a_i$

数据范围：$n, m, a_i, x \leq 2 \times 10^5$

思路

回忆高中的数学知识：

$$\sin(a+x) = \sin a_i \cos x + \sin x \cos a$$

那么，我们显然可以构造一个中间量 $\cos x$ 来辅助转移，而又有：

$$\cos(a+x) = -\sin a_i \sin x + \cos x \cos a$$

那么转移矩阵也就显而易见了：

$$\begin{pmatrix} \sin(a_i+x) \\ \cos(a_i+x) \end{pmatrix} = \begin{pmatrix} \cos x & \sin x \\ -\sin x & \cos x \end{pmatrix} \begin{pmatrix} \sin a_i \\ \cos a_i \end{pmatrix}$$

使用线段树维护即可。

总结

带修动态规划是数据结构与数学的一个较难的考点，但是其模型非常经典，不能不会，本质上就是用高级数据结构维护矩阵乘法。其次，我们不难发现，矩阵乘法只能维护常递推结构，即对于 F(n)，其转移的来的项目是固定的 (结构固定)，例如斐波那契数列或者杨辉三角之类的结构。而对于 LIS 这类问题，是 DDP 无法解决的。注意使用 DDP 的场景，以及递推转移矩阵的构建，即可轻松解决这类问题。

Thank You!

Reference

1 xxxxxxxxxxxxxxx

2 xxxxxxxxxxxxxxx