数学 03

- 乘法逆元
- 快速幂求乘法逆元
- 扩欧求乘法逆元
- 题目选讲

乘法逆元

给定模数 m，对于 $a \in \{0,\ 1,\ 2,\ ..,\ m-1\}$，若存在 $x \in \{0,\ 1,\ 2,\ ..,\ m-1\}$，使得

$$ax \equiv 1\ (mod\ m)$$

则称 x 为 a 在模 m 意义下的乘法逆元

为何需要乘法逆元

在 OI 中，计算结果常常过大，以致于爆出精度范围，常见的办法是对计算结果进行模 m 处理

在模 m 意义下，加减乘都可以照常进行，唯独除法不再成立，因为模 m 意义下的值均为整数 0，1，2，$..$，$m-1$，所以需要引入模意义下的除法，即乘法逆元

乘法逆元存在条件

当且仅当 $(a,\ m) = 1$ 时，即 a 和 m 互质时，a 在模 m 意义下的乘法逆元存在

0 没有乘法逆元

乘法逆元的求法

基于费马小定理和扩展欧几里得算法，时间复杂度均为 $O(log)$ 级别

前者局限于模数 m 只能是质数，后者没有限制

除此之外，还可以根据欧拉定理进行计算，但是效率不如上述两种方式

基于费马小定理与快速幂

考虑非零整数 a 和质数 m，根据费马小定理 $a^m \equiv a \ (mod\ m)$

所以 $a(a^{m-1} - 1) \equiv 0 \ (mod\ m)$，即 $a^{m-1} \equiv 1 \ (mod\ m)$，即 $a \cdot a^{m-2} \equiv 1 \ (mod\ m)$

所以 a 在模 m 意义下的逆元为 a^{m-2}，根据快速幂算法可以在 $O(logm)$ 时间内计算出乘法逆元

代码

```
int qpow(int a, int b, int m)
{
    int ans = 1;
    while(b) {
        if(b & 1) ans *= a, ans %= m;
        a *= a, a %= m, b >>= 1;
    }
    return ans;
}
int getInv(int a, int m)
{
    return qpow(a, m - 2, m);
}
```

基于扩展欧几里得算法

此方法对模数 m 没有限制

由 $ax \equiv 1 \ (mod\ m)$，得到 $ax + my = 1$

调用扩欧计算出模 m 意义下的最小正整数 x 即可，时间复杂度为 $O(logm)$

代码如下

```
int exgcd(int a, int b, int &x, int &y)
{
    if(!b) {x = 1, y = 0; return a;}
    int r = exgcd(b, a % b, y, x); // y的值被修改为x', x的值被修改为y'
    y -= (a / b) * x;
    return r;
}

int getInv(int a, int m)
{
    int x, y;
    int r = exgcd(a, m, x, y);
    if(r != 1) return -1; // 不存在逆元
    while(x < 0) x += m;
    return x % m;
}
```

2546. 你真的会乘法逆元吗?

给定 n，p，计算 $1 \sim n$ 在模 p 意义下逆元

数据保证 $1 \le n \le 3 \times 10^6$，$n < p < 20000528$

题解

对于 $1 \le i \le n$，对 p 做待余除法，得到 $q = \left\lfloor \frac{p}{i} \right\rfloor$，$j = p \bmod i$，
所以 $p = qi + j,\ 0 \le j < i$

根据 $p = qi + j \equiv 0 \ (mod\ p)$

得到 $i^{-1} = -qj^{-1} \ (mod\ p)$

用数组维护 j^{-1}，便可以计算出 i^{-1}，时间复杂度为 $O(n)$

2548. 你真的真的会乘法逆元吗?

给定 n 个正整数 a_i，质数 p，正整数 k，计算

$$\sum_{i=1}^{n} \frac{k^i}{a_i} \pmod p$$

数据保证 $1 \le n \le 5 \times 10^6$，$2 \le k < p < 10^9$，$1 \le a_i < p$

题解

考虑快速计算 a_i 的逆元

计算前缀积 $m_j = \prod_{i=1}^{j} a_i$，计算 $m_n^{-1} = (a_1 a_2 \cdots a_n)^{-1}$，于是

$$a_i^{-1} = m_i^{-1} \cdot m_{i-1}$$

其中

$$m_i^{-1} = m_n^{-1} \cdot \prod_{j=i+1}^{n} a_j$$

只需要从后往前递推，便可以在 $O(n + logp)$ 时间内处理出所有的乘法逆元

接下来循环递推即可，总的时间复杂度为 $O(n + logp)$