寒训（III）

主讲人：陈攒鑫

目录

（大）部分内容来源与 oi-wiki

数论基础

整除

- 定义：设 $a, b \in \mathbb{Z}$，$a \neq 0$。如果 $\exists q \in \mathbb{Z}$，使得 $b = aq$，那么就说 b 可被 a 整除, 记作$a \mid b$，反之记作 $a \nmid b$。
- 简单性质：
  - $a \mid b \Leftrightarrow -a \mid b \Leftrightarrow a \mid -b \Leftrightarrow |a| \mid |b|$
  - $a \mid b \wedge b \mid c \Longrightarrow a \mid c$
  - $a \mid b \wedge a \mid c \Leftrightarrow \forall x, y \in \mathbb{Z}, a \mid (xb + yc)$
  - $a \mid b \wedge b \mid a \Longrightarrow b = \pm a$
  - 设 $m \neq 0$，那么 $a \mid b \Leftrightarrow ma \mid mb$。
  - 设 $b \neq 0$，那么 $a \mid b \Longrightarrow |a| \leq |b|$。
  - 设 $a \neq 0, b = qa + c$，那么 $a \mid b \Leftrightarrow a \mid c$。

整除

- 约数（因数）、倍数
  - 0 是所有非 0 整数的倍数。
  - 对于整数 $b \neq 0$，b 的约数只有有限个。
- 平凡约数（平凡因数）
  - 对于整数 $b \neq 0$，± 1、$\pm b$ 是 b 的平凡约数。
  - 当 $b = \pm 1$ 时，b 只有两个平凡约数。
  - 对于整数 $b \neq 0$，b 的其他约数称为真约数（真因数、非平凡约数、非平凡因数）。

整除

- 约数的性质：
  - 设整数 $b \neq 0$，当 d 遍历 b 的全体约数的时候，$\frac{b}{d}$ 也遍历 b 的全体约数。
  - 设整数 $b > 0$，当 d 遍历 b 的全体正约数的时候，$\frac{b}{d}$ 也遍历 b 的全体正约数。
- 如果没有特别说明，约数总是指正约数。

带余数除法

- 余数：没有特别说明，总是指最小非负余数。
- 余数性质：
  - 任一整数被正整数 a 除后，余数一定是且仅是 0 到 $(a-1)$ 这 a 个数中的一个。
  - 相邻的 a 个整数被正整数 a 除后，恰好取到上述 a 个余数。特别地，一定有且仅有一个数被 a 整除。

最大公约数与最小公倍数

- 最大公约数(GCD)：
  - 公约数：一组整数的公约数，是指同时是这组数中每一个数的约数的数。
  - 最大公约数：所有公约数里面最大的一个。
- 最小公倍数(LCM)：
  - 所有正的公倍数里面，最小的一个数。
- 求GCD：欧几里得算法，复杂度为$O(\log n)$
- $a, b > 0$，则$LCM(a, b) = \frac{a \times b}{GCD(a,b)}$
- 裴蜀定理：设 a, b 是不全为零的整数，对任意整数 x, y，满足 $\gcd(a, b) \mid ax + by$，且存在整数 x, y, 使得 $ax + by = \gcd(a, b)$。

素数与合数

- 没有特别说明，素数总是指正的素数。
- 简单性质：
  - 大于 1 的整数 a 是合数，等价于 a 可以表示为整数 d 和 e（$1 < d, e < a$）的乘积。
  - 如果素数 p 有大于 1 的约数 d，那么 $d = p$。
  - 大于 1 的整数 a 一定可以表示为素数的乘积。
  - 对于合数 a，一定存在素数 $p \le \sqrt{a}$使得 $p \mid a$。
  - 素数有无穷多个。
  - 所有大于 3 的素数都可以表示为 $6n \pm 1$ 的形式。

算术基本定理

- 算术基本定理（唯一分解定理）：
  - 设正整数 a，那么必有表示：
$$a = p_1 p_2 \cdots p_s$$
  - 其中 $p_j (1 \leq j \leq s)$是素数。并且在不计次序的意义下，该表示唯一。
- 标准素因数分解式：
  - 将上述表示中，相同的素数合并，可得：
$$a = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_s^{\alpha_s}, p_1 < p_2 < \cdots < p_s$$
  - 称为正整数 a 的标准素因数分解式。

同余

- 定义：
  - 设整数 $m \neq 0$。若 $m \mid (a-b)$，称 m 为模数（模），a 同余于 b 模 m，b 是 a 对模 m 的剩余。记作 $a \equiv b \pmod{m}$。
  - 若 a 不同余于 b 模 m，b 不是 a 对模 m 的剩余。记作 $a \not\equiv b \pmod{m}$。
  - 式中的 b 是 a 对模 m 的剩余，这个概念与余数完全一致。通过限定 b 的范围，相应的有 a 对模 m 的最小非负剩余、绝对最小剩余、最小正剩余。
- 性质：
  - 线性运算：若 $a, b, c, d \in \mathbb{Z}, m \in \mathbb{N}^*, a \equiv b \pmod{m}, c \equiv d \pmod{m}$，则有：

$$a \pm c \equiv b \pm d \pmod{m}$$

$$a \times c \equiv b \times d \pmod{m}$$

同余

- 若 $a, b \in \mathbb{Z}, k, m \in \mathbb{N}^*, a \equiv b \pmod{m}$, 则 $ak \equiv bk \pmod{mk}$。
- 若 $a, b \in \mathbb{Z}, d, m \in \mathbb{N}^*, d \mid a, d \mid b, d \mid m$，则当 $a \equiv b \pmod{m}$ 成立时，有 $\frac{a}{d} \equiv \frac{b}{d} \left(\bmod \frac{m}{d}\right)$。
- 若 $a, b \in \mathbb{Z}, d, m \in \mathbb{N}^*, d \mid m$，则当 $a \equiv b \pmod{m}$ 成立时，有 $a \equiv b \pmod{d}$。
- 若$a, b \in \mathbb{Z}, d, m \in \mathbb{N}^*$，则当 $a \equiv b \pmod{m}$ 成立时，有 $\gcd(a, m) = \gcd(b, m)$。若 d 能整除 m 及 a, b 中的一个，则 d 必定能整除 a, b 中的另一个。

数论函数

- 数论函数：
  - 指定义域为正整数的函数。数论函数也可以视作一个数列。
- 积性函数
  - 定义：若函数 $f(n)$ 满足 $f(1) = 1$ 且 $\forall\, x, y \in \mathbb{N}^*, \gcd(x, y) = 1$ 都有 $f(xy) = f(x)f(y)$，则 $f(n)$ 为积性函数。
  - 若去除 $\gcd(x, y) = 1$ 条件后仍然满足 $f(xy) = f(x)f(y)$，则 $f(n)$ 为完全积性函数。
- 常见数论函数例子：
  - 单位函数：$\varepsilon(n) = [n = 1]$。（完全积性）
  - 恒等函数：$\mathrm{id}_{k(n)} = n^k$，$\mathrm{id}_1(n)$通常简记作 $\mathrm{id}(n)$。（完全积性）

数论函数

- 常数函数：$1(n) = 1$。（完全积性）
- 除数函数：$\sigma_k(n) = \sum_{d|n} d^k$。$\sigma_0(n)$通常简记作 $d(n)$ 或 $\tau(n)$，$\sigma_1(n)$通常简记作 $\sigma(n)$。
- 欧拉函数：$\varphi(n) = \sum_{i=1}^{n} [\gcd(i,n) = 1]$。（$1 \sim n$ 中与 n 互质数字个数）
- 莫比乌斯函数：$\mu(n) = \begin{cases} 1 & n = 1 \\ 0 & \exists d > 1, d^2 \mid n \\ (-1)^{\omega(n)} & \text{otherwise} \end{cases}$，其中 $\omega(n)$表示 n 的本质不同质因子个数，它是一个加性函数（与积性函数类似：a, b 互质时 $f(ab) = f(a) + f(b)$）。

筛法

引入

- 如何找出 $1 \sim n$ 中所有的素数？
  - 质数检验？
  - 对于 $1 \sim n$ 所有数字进行一次质数检验，时间复杂度为 $O(n\sqrt{n})$，当 n 较大时不能接受。
- 筛法：埃拉托斯特尼筛法（埃氏筛）、线性筛

埃氏筛

- 考虑 $1\sim n$ 中任意整数，它的 x 倍都是合数（$x > 1$）。
- 从小到大依此考虑 $1\sim n$ 中的每个数，将当前这个数的所有≥ 2的倍数记为合数。
- 事实上，根据唯一分解定理，只要考虑每个素数的倍数即可；
- 同时，对于素数 p，我们只要考虑其 $\geq p$ 的倍数即可，因为 $\leq p$ 的倍数在之前已经被筛去了。
- 时间复杂度为 $O(n \log\log n)$。
- 事实上其还有很多奇怪的优化，具体点击这里。

埃氏筛

- 参考代码

```cpp
// 程序段来源于oi-wiki
vector<int> prime;
bool is_prime[N];

void Eratosthenes(int n) {
    is_prime[0] = is_prime[1] = 0;
    for (int i = 2; i <= n; ++i) is_prime[i] = 1;
    for (int i = 2; i <= n; ++i) {
        if (is_prime[i]) {
            prime.push_back(i);
            for (int j = i * i; j <= n; j += i)
                // 因为从 2 到 i - 1 的倍数我们之前筛过了，这里直接从 i 的倍数开始，提高了运行速度
                is_prime[j] = 0;  // 是 i 的倍数的均不是素数
        }
    }
}
```

线性筛法

- 埃氏筛会将一个合数重复多次标记。
- 如何去除重复标记？
- 能否让每个合数只会被其最小的质因子筛去？
- 时间复杂度为严格 $O(n)$。

线性筛法

- 参考代码

```cpp
vector<int> pri;
bool not_prime[N];

void pre(int n) {
    for (int i = 2; i <= n; ++i) {
        if (!not_prime[i]) pri.push_back(i);
        for (int pri_j : pri) {
            if (i * pri_j > n) break;
            not_prime[i * pri_j] = 1;
            if (i % pri_j == 0) {
                // i % pri_j == 0 换言之，i 之前被 pri_j 筛过了
                // 由于 pri 里面质数是从小到大的，所以 i 乘上其他的质数的结果一定会被 pri_j 的倍数筛掉，
                // 就不需要在这里先筛一次，所以这里直接 break 掉就好了
                break;
            }
        }
    }
}
```

线性筛法

- 能用于求解一些积性函数；
- 能用于求约数个数；
- 结合链表能用于分解质因数。

模板题

- [P3383 【模板】线性筛素数 (luogu.com.cn)]
- 由于时限为 2s，所以两个筛法都能过。
- 测试效率：

线性筛	4.79s / 129.82MB / 787B C++20 **O2**
埃氏筛	7.69s / 130.11MB / 773B C++20 **O2**

扩展欧几里得

引入

- 回忆裴蜀定理，考虑不定方程 $ax + by = c\ (c \neq 0)$，其有解的充要条件为 $\gcd(\mathrm{a}, \mathrm{b}) \mid c$；
- 接下来我们只需考虑不定方程 $ax + by = \gcd(a, b)$。

推导过程

- 考虑欧几里得算法，递归到边界即 $b_0 = 0$ 时，$a_0 = \gcd(a, b)$，此时有 $x_0 = 1, y_0 = 1$，可满足 $a_0x_0 + b_0y_0 = \gcd(a, b)$。
- 考虑回溯状态，已知 $a_ix_i + b_iy_i = \gcd(a, b)$, $a_i = b_{i+1}, b_i = a_{i+1} \,\%\, b_{i+1}$，令 $a_{i+1} = kb_{i+1} + b_i$，则有
$$b_{i+1}x_i + (a_{i+1} - kb_{i+1})y_i = \gcd(a, b)$$
- 整理一下得
$$a_{i+1}y_i + b_{i+1}(x_i - ky_i) = \gcd(a, b)$$
- 故我们可以得到 $x_{i+1} = y_i$, $y_{i+1} = x_i - \left\lfloor\frac{a_{i+1}}{b_{i+1}}\right\rfloor y_i$。
- 用欧几里得算法即可求解。

参考代码

```
int exgcd(int a, int b, int &x, int &y){
    if (!b) {x=1, y=0; return a;}
    int gcd = exgcd(b, a%b, y, x);  // 交换实参 x,y
    y -= (a / b) * x;   //根据递推式更改当前y值
    return gcd;
}
```

模板题

- P5656 【模板】二元一次不定方程 (exgcd) (luogu.com.cn)
- 在 Exgcd 基础上增加了若干分类讨论。
- 了解如何求最小/大正整数解、最小/大整数解。

参考代码

```cpp
int exgcd(int a, int b, int &x, int &y){
    if (!b) {x = 1, y = 0; return a;}
    int gcd = exgcd(b, a%b, y, x);
    y -= (a / b) * x;
    return gcd;
}

void solve(){
    int a, b, c, x, y;
    cin >> a >> b >> c;
    int g = exgcd(a, b, x, y);
    if (c % g){     //无解
        cout << "-1\n";
        return;
    }
    a /= g, b /= g, c /= g; //将三个系数约分，方便后序讨论最值解
    x *= c, y *= c;
    int minx, maxy, miny, maxx, sum;
    minx = (x > 0 && x % b) ? x % b : x % b + b;    //将 x 变为最小正数
    maxy = (c - minx * a) / b;                      //根据 x 解出 y
    miny = (y > 0 && y % a) ? y % a : y % a + a;    //将 y 变成最小正数
    maxx = (c - miny * b) / a;                      //根据 y 解出 x
    sum = 0;
    if (maxx > 0) sum = (maxx - minx) / b + 1;      //求得正整数解个数
    if (!sum) cout << minx <<' '<< miny <<'\n';
    else cout << sum <<' '<< minx <<' '<< miny <<' '<< maxx <<' '<< maxy <<'\n';
}
```

乘法逆元

定义

- 模意义下乘法运算的逆元。
- 如果一个线性同余方程 $ax \equiv 1 \pmod b$，则 x 称为 $a \bmod b$ 的逆元，记作 a^{-1}。

费马小定理求逆元

- 要求模数 b 为素数，且 $\gcd(a, b) = 1$。
- 费马小定理：b 为素数且 $\gcd(a, b) = 1$ 时，有 $a^{b-1} \equiv 1 \pmod{b}$
- 根据定义 $a^{-1} = a^{b-2}$，使用快速幂求解即可。

扩展欧几里得求逆元

- 要求 $\gcd(a,b)=1$。
- 利用 exgcd 解线性方程 $ax+by=1$。
- 解得 x 显然满足 $ax\equiv 1 \pmod{b}$
- 根据定义 $a^{-1}=x$，考虑到 exgcd 的解可能为负数，故最终结果一般写成 $\big((x\%b)+b\big)\%b$

线性递推求逆元

- 用于求解 $1,2,\dots,n$ 中每个数关于 p 的逆元。
- 显然有 $1^{-1}=1 \pmod p$；
- 对于求 i^{-1}，令 $k=\left\lfloor\frac{p}{i}\right\rfloor, j=p \bmod i$，则有 $p=ki+j$，故能得到$ki+j\equiv 0 \pmod p$；
- 左右同乘 $i^{-1}\times j^{-1}$，得到 $kj^{-1}+i^{-1}\equiv 0 \pmod p$；
- 带入 $j=p \bmod i$，得到 $i^{-1}\equiv -\left\lfloor\frac{p}{i}\right\rfloor(p \bmod i)^{-1} \pmod p$；
- 由于 $p \bmod i<i$，故可通过递推求得 i^{-1}：

$$i^{-1}=\begin{cases}1 & i=1\\ -\left\lfloor\frac{p}{i}\right\rfloor(p \bmod i)^{-1} & \text{otherwise}\end{cases} \pmod p$$

线性递推求逆元

- 该算法时间复杂度为 $O(n)$；
- 根据上述递归式，也能递归求解单个数字的乘法逆元，时间复杂度上界为 $O(\sqrt[3]{n})$（搬运自 oi-wiki）相关讨论。
- 参考程序

```
inv[1] = 1;
for (int i = 2; i <= n; i++){
    inv[i] = (p - p / i) * inv[p % i] % p;
}
```

*线性求任意n个数的逆元

- 上述线性求逆元只能求 $1 \sim n$ 的逆元，现在要求任意给定的 n 个数 $a_1, \ldots, a_n$ 的逆元；
- 首先计算 n 个数的前缀积，记为 s_i，然后使用快速幂或扩展欧几里得法计算 s_n 的逆元，记为 sv_n；
- 对于 sv_n 乘上 a_n 会与 a_n 的逆元抵消，得到 $a_1 a_2 \ldots a_{n-1}$ 的逆元，记作 sv_{n-1}，同样的方式能求出任意 sv_i；
- 对于 a_i^{-1}，可以通过 $sv_i \times s_{i-1}$ 求得。

模板题

- [P3811 【模板】模意义下的乘法逆元 (luogu.com.cn)]()
- 直接使用上述的线性递推求逆元模板即可。
- [P5431 【模板】模意义下的乘法逆元2 (luogu.com.cn)]()
- 直接使用上述任意 n 个数的逆元递推累加即可得到答案；
- 事实上也可以将原式通分后利用前缀后缀和求解，只需求 $a_1a_2\dots a_n$ 逆元即可；
- 注意解绑后的 cin以及scanf 可能无法通过此题，使用快读后可读入。

参考代码

```cpp
inline ll read(){
    ll x = 0, tag = 1;
    char c = getchar();
    for (; !isdigit(c); c = getchar()) if (c == '-') tag = -1;
    for (; isdigit(c); c = getchar()) x = (x<<1) + (x<<3) + c - '0';
    return x * tag;
}

signed main(){
    ll n, p, k;
    n=read(),p=read(),k=read();
    vector<ll> a(n+1),s(n+1),sv(n+1);
    s[0]=1;
    for (ll i=1;i<=n;i++){
        a[i] = read();
        s[i] = (s[i - 1] * a[i]) % p;
    }
    ll _;
    exgcd(s[n], p, sv[n], _);
    sv[n]=(sv[n] % p + p) % p;
    for (ll i = n - 1; i; i--){
        sv[i] = sv[i + 1] * a[i + 1] % p;
    }
    ll nowk = k, ans = 0;
    for (ll i = 1; i <= n; i++){
        (ans += nowk * s[i - 1] % p * sv[i] % p) %= p;
        (nowk *= k) %= p;
    }
    cout << ans <<'\n';
    return 0;
}
```

谢谢大家^_^