提高数论选讲

洛谷夏令营-提高组

will7101

Aug. 2018

清华大学

Part 1

目录

倍数

- 对于自然数 a, b，如果存在自然数 k，使得 $ka = b$，那么 b 是 a 的倍数，称 a 整除 b，b 能被 a 整除，记做 $a|b$
- 一个数有无穷多个倍数
- 所有数都是自身和 1 的倍数
- 倍数具有传递性
- 在 $[1, n]$ 范围内的 x 的倍数有 $\left\lfloor \frac{n}{x} \right\rfloor$ 个

约数

- 对于自然数 a, b，如果 b 是 a 的倍数，那么 a 是 b 的约数，又称因数
- 所有数的约数都包含自身和（或）1
- 一个自然数只有有限个约数，约数的个数通常记为 $d(n)$
- 打个表看看

n	1	2	3	4	5	6	7	8	9	10
$d(n)$	1	2	2	3	2	4	2	4	3	4

怎么打表

```cpp
int d[MAXN];
// 计算出 [1, n] 中每个数的约数个数
// 复杂度 O(n log n)
void calc_divisors(int n) {
    for (int i = 1; i <= n; ++i) {
        // 枚举 i 的所有倍数
        for (int j = i; j <= n; j += i) {
            d[j]++;
        }
    }
}
```

目录

素数与合数

- 素数在数论中的地位十分重要，许多理论都围绕素数而建立
- 如果一个数的约数只有两个（1 和自身），那么称它为素数（或者质数）
- 如果一个数有非平凡（除了 1 和自身以外）的约数，就称它为合数
- 1 既不是素数也不是合数！

素性判定

```cpp
// 判断一个数是否为素数
bool is_prime(int x) {
    // 特判 1 的情况
    if (x == 1) return false;
    // 从 2 枚举到 √x，判断是否能整除
    for (int i = 2; i * i <= x; ++i) {
        // 如果能整除，则为合数
        if (x % i == 0) return false;
    }
    // 如果都不能整除，则为素数
    return true;
}
```

朴素筛法

```cpp
int vis[MAXN], p[MAXN], cnt;
// 筛出 [2, n] 之间所有的素数
// 复杂度: O(n log n)
void naive_sieve(int n) {
    for (int i = 2; i <= n; ++i) {
        if (!vis[i]) p[cnt++] = i;
        // 筛掉 i 除了本身以外的所有倍数
        for (int j = i + i; j <= n; j += i) {
            vis[j] = 1;
        }
    }
}
```

埃氏筛法

```cpp
int vis[MAXN], p[MAXN], cnt;
// 筛出 [2, n] 之间所有的素数
// 复杂度：O(n log log n)
void eratosthenes_sieve(int n) {
    for (int i = 2; i <= n; ++i) {
        if (!vis[i]) {
            p[cnt++] = i;
            // 筛掉 i 除了本身以外的所有倍数
            for (int j = i + i; j <= n; j += i) {
                vis[j] = 1;
            }
        }
    }
}
```

欧拉筛法（线性筛）

```cpp
int vis[MAXN], p[MAXN], cnt;
// 筛出 [2, n] 之间所有的素数
// 复杂度：O(n)
void euler_sieve(int n) {
    for (int i = 2; i <= n; ++i) {
        if (!vis[i]) p[cnt++] = i;
        for (int j = 0; i * p[j] <= n; ++j) {
            vis[i * p[j]] = 1;
            // 保证 p[j] 是 i*p[j] 最小的素因子
            if (i % p[j] == 0) break;
        }
    }
}
```

唯一分解定理（算术基本定理）

定理 (唯一分解定理)

每个大于 1 的自然数均可写为素数的积，而且这些素因子按大小排列之后，写法仅有一种方式。

- $105 = 3 \times 5 \times 7$
- $200 = 2 \times 2 \times 2 \times 5 \times 5 = 2^3 \times 5^2$
- 写成素因子的幂的积，$x = {p_1}^{k_1} {p_2}^{k_2} {p_3}^{k_3} \cdots {p_n}^{k_n}$
- 如果不含某个素因子，可以看作它的次数为 0
- 乘法就是对应素因子的次数相加，除法就是对应素因子的次数相减
- 如果一个数所有的素因子次数都大于等于另一个数对应的素因子次数，那么就可以被它整除

素因数分解

```cpp
// 将 x 分解素因数，结果从小到大储存在 p[] 中，返回素因子的个数
int factorize(int x, int p[]) {
    int cnt = 0;
    // 从 2 枚举到 √x，判断是否为约数
    for (int i = 2; i * i <= x; ++i) {
        // 如果找到一个约数，就不断从 x 中除去
        while (x % i == 0) {
            p[cnt++] = i;
            x /= i;
        }
    }
    // 如果 x > 1，则说明剩下的 x 是素数，也要放进数组
    if (x > 1) p[cnt++] = x;
    return cnt;
}
```

目录

最大公约数 (gcd)

定义 (最大公约数)

两个自然数所共有的约数中最大的一个，称为它们的最大公约数 (Greatest Common Divisor)

- 利用唯一分解定理及推论，可以发现 gcd 就是两个数对应的素因子次数取 min 后得到的数
- 实际应用中，通常使用辗转相除法来计算 gcd

辗转相除法（欧几里得算法）

引理（辗转相除原理）

定义对于任意自然数 a，$\gcd(0, a) = \gcd(a, 0) = a$。则对于任意自然数 a, b，满足 $\gcd(a, b) = \gcd(a, b \bmod a)$

- 利用这个性质，可以将 a, b 两者中较大的一个不断缩小，直到变为零
- 每次大的数对小的数取模，至少缩小一半，所以复杂度为 $O(\log \max(a, b))$

辗转相除法

```
// 循环实现
int gcd(int a, int b) {
    while (a > 0) {
        int t = b % a;
        b = a;
        a = t;
    }
    return b;
}

// 递归实现
int gcd(int a, int b) {
    if (a == 0) return b;
    return gcd(b % a, a);
}
```

最小公倍数 (lcm)

定义 (最小公倍数)

两个自然数所共有的倍数中最小的一个，称为它们的最小公倍数 (Least Common Multiple)

- 利用唯一分解定理及推论，可以发现 lcm 就是两个数对应的素因子次数取 max 后得到的数
- 不需要另外去求，只要用一个公式就能转化成 gcd 问题
- $\operatorname{lcm}(a, b) = \dfrac{ab}{\gcd(a, b)}$
- 可以用唯一分解定理来证明

扩展欧几里得算法

定理 (裴蜀定理)

对于任意整数 a, b，存在无穷多组整数对 (x, y) 满足不定方程 $ax + by = d$，其中 $d = \gcd(a, b)$

- 在求 $\gcd(a, b)$ 的同时，可以求出（关于 x, y 的）不定方程 $ax + by = d$ 的一组整数解
- 考虑递归计算：假设已经算出了 $(b\%a, a)$ 的一组解 (x_0, y_0) 满足 $(b\%a)x_0 + ay_0 = d$
- 可以得到 $(b - a\left\lfloor \frac{b}{a} \right\rfloor)x_0 + ay_0 = d$
- 整理得到 $a(y_0 - \left\lfloor \frac{b}{a} \right\rfloor x_0) + bx_0 = d$
- 为了方便，交换 x_0 和 y_0，得到 $a(x_0 - \left\lfloor \frac{b}{a} \right\rfloor y_0) + by_0 = d$

扩展欧几里得算法

```
// 返回 gcd(a, b) 和方程 ax + by = gcd(a, b) 的一组解
int gcd(int a, int b, int &x, int &y) {
    if (a == 0) {
        // gcd(0, b) = b，显然 0 · 0 + 1 · b = b 满足条件
        x = 0, y = 1;
        return b;
    }
    // 这里交换了 x_0 和 y_0
    int d = gcd(b % a, a, y, x);
    // x = x_0 - b / a \times y_0
    // y = y_0
    x -= b / a * y;
    return d;
}
```

目录

P1403 约数研究

设 $f(i)$ 为 i 的约数个数，求 $f(1)+f(2)+f(3)+\cdots+f(n)$。

$n \le 10^6$

- 直接暴力统计，最优能做到 $O(n\sqrt{n})$
- 使用我们上面的打表方法，可以做到 $O(n \log n)$
- 实际上，并不需要知道具体每个数的 f，只要统计总和，这等价于求所有数的范围内的倍数个数之和，于是可以直接对于每个数 i，将答案加上 $\left\lfloor \frac{n}{i} \right\rfloor$，复杂度 $O(n)$

P1072 Hankson 的趣味题

给出 a_0, a_1, b_0, b_1，求满足 $\begin{cases}\gcd(x, a_0) = a_1 \\ \operatorname{lcm}(x, b_0) = b_1\end{cases}$ 的 x 的个数。

题目有多组数据，至多 2000 组数据。

- 对于某个素数 p，假如在 a_0 中的次数为 k_{a0}，在 a_1 中的次数为 k_{a1}，那么在 x 中的次数 k 应该满足 $\min(k_{a0}, k) = k_{a1}$
- 分情况讨论，如果 $k_{a0} < k_{a1}$，那么就无解；如果 $k_{a0} = k_{a1}$，那么就只要 $k \geq k_{a0}$；如果 $k_{a0} > k_{a1}$，那么 k 就必须等于 k_{a_1}
- 对于 b 中的情况也是类似讨论，把 min 换成 max
- 最后根据乘法原理，将每个素因子的答案相乘，得到答案
- 可以 $O(\sqrt{n})$ 来枚举素因子，由于每个数至多只有一个大于 $\sqrt{n}$ 的素因子，特判即可
- 预处理，筛出小素数，可以进一步优化

P2520 [HAOI2011] 向量

输入 a, b, x, y，能否用
$(a, b), (a, -b), (-a, b), (-a, -b), (b, a), (b, -a), (-b, a), (-b, -a)$
拼出 (x, y)

- 首先，设 $d = \gcd(a, b)$，如果 x, y 不能整除 d，显然无解，否则就把 d 约掉
- 我们可以组合出这样的操作：横/纵坐标单独加/减 $2a$ 或 $2b$
- 然后根据裴蜀定理，可以做到横/纵坐标单独加/减 2
- 然后考虑奇偶性，分别枚举 $(0, 0), (a, b), (b, a), (a + b, a + b)$ 四种初始走法就行了

Part 2

目录

取模运算

- 对于自然数 a，正整数 m，$a \bmod m = a - m \left\lfloor \frac{a}{m} \right\rfloor$
- 也称取余，因为实际上就是整除得到的余数
- $a \bmod m \in [0, m)$
- 取模运算和除法一样慢
- 尽量不要负数取模

模意义下的数和运算

- 如果 $a \bmod m = b \bmod m$，可以记做 $a \equiv b \pmod m$
- 所有自然数都能用 $[0, m)$ 之间的整数来“代表”
- 可以把取模推广到负数，$a \bmod m = b (b \in [0, m)) \Leftrightarrow$ 存在整数 k，满足 $a = km + b$
- 我们可以建立一个新的数字运算体系，其中只有 $0 \sim m - 1$ 这 m 个数，然后建立四则运算等基本规则
- 两个数相加，如果超出了 $m - 1$，就从 0 开始再往上加，相乘与相加类似
- 两个数相减，如果小于 0，就从 $m - 1$ 再往下减
- 32 位无符号整数（unsigned int）的运算实际上是对 2^{32} 取模的

运算的性质

- $(a+b)\%m = (a\%m + b\%m)\%m$
- $(a-b)\%m = (a\%m - b\%m)\%m$
- $(a\times b)\%m = (a\%m)\times(b\%m)\%m$
- 假如题目要求最终答案对 m 取模，那么（为了防止数字过大）可以在每一步运算中都取模

目录

乘法逆元

- 加法、减法、乘法都定义好了，除法怎么做呢？
- 可以不断将被除数加上 m，直到可以除尽，但是这样做效率太低了
- 回想起以前学习分数时，除以一个数，可以转化成乘它的倒数
- 类似地，我们可以试着为除数找到一个乘法逆元，即满足 $xx^{-1} \equiv 1 \pmod{m}$ 的整数 x^{-1}
- 当且仅当 m 为素数时，每个数都有唯一的乘法逆元，因此大部分 OI 题中的模数都是素数

费马小定理

定理 (费马小定理)

对于任意质数 p 和正整数 $a < p$，有 $a^{p-1} \equiv 1 \pmod{p}$

证明.

- 易证，$a, 2a, 3a, \ldots, (p-1)a \pmod{p}$ 可以取遍 1 到 $p-1$ 之间的所有数
- 于是 $a \cdot 2a \cdot 3a \cdots (p-1)a \equiv (p-1)! \pmod{p}$
- 由于 p 是素数，和 $1 \sim p-1$ 之间的数都互素，所以等式两边可以同时除以 $(p-1)!$
- 就得到了 $a^{p-1} \equiv 1 \pmod{p}$

□

快速求逆元

- 有了费马小定理，就可以快速求出一个数的乘法逆元
- 由于 $a^{p-2} \cdot a \equiv a^{p-1} \equiv 1 \pmod{p}$，所以 $a^{p-2}\%p$ 就是逆元
- 可以通过快速幂来计算，复杂度 $O(\log p)$
- 另外，也可以用扩展欧几里得算法来求逆元，求出 $ax + py = 1$ 的一组解，$x\%p$ 就是逆元

目录

中国剩余定理 (CRT)

- 有若干个人列队，三个一排会多出两个人，五个一排多三个人，七个一排多两个人，求人数至少是多少？
- 本质上是解一元线性同余方程组 $\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \quad \vdots \\ x \equiv a_n \pmod{m_n} \end{cases}$
- 当 m 两两互素时，一定有解

中国剩余定理 (CRT)

- 我们来尝试构造出这个解
- 设 $M = m_1 \times m_2 \times \cdots \times m_n$，$M_i = \dfrac{M}{m_i}$
- M_i 只有在模 m_i 的时候不为零，模其他的 m 都为零
- 由于我们想在模 m_i 时得到 a_i，所以需要乘上 $a_i t_i$，其中 t_i 是 M_i 在模 m_i 意义下的乘法逆元
- 最终得到 $a_1 t_1 M_1 + a_2 t_2 M_2 + \cdots + a_n t_n M_n$，再对 M 取模，就得到了最小的自然数解，加上 kM 就是通解

目录

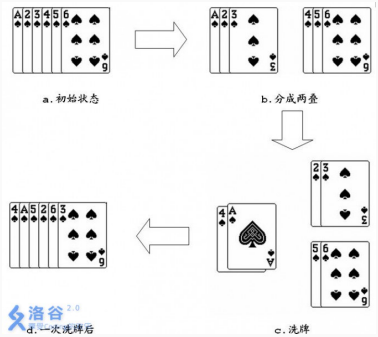
P4079 [SDOI2016] 齿轮

给出一个有向图，每条边代表连接的两个点上的数字需要满足某个比例，求是否有一种合法的方案给每个点分配一个数字

- 可以使用 dfs 或者并查集来做，依次分配数字
- 用整数型会溢出，用浮点会爆精度
- 可以考虑将所有数字对某个 P 取模，然后进行验证
- 但是这样会造成假阳性，所以需要取多个 P，分别做一遍，可以很大概率通过本题，原理与多模数 hash 类似

P2054 [AHOI2005] 洗牌

一开始有 n 张牌按顺序排列，每次按照规则洗牌，求洗 m 次之后的第 l 张牌。



- 把每次洗牌的过程反过来，会发现，假设一个位置上原来的数字是 x，洗牌前的数字是 $2x\%(n+1)$
- 解方程 $2^m x \equiv l \pmod{(n+1)}$
- 求出 2^m 的逆元即可，可以用扩欧，也可以用欧拉函数