基本数学算法 part 2[1]

赖金霖[2]
清华大学　计算机科学与技术系

Aug 11, 2019

[1]本文件可在这里找到: https://github.com/lll6924/public_slides
[2]邮箱: laijl16@mails.tsinghua.edu.cn

素数判定

问题：输入一个正整数 N，判断它是否是质数

- 算法一：从 2 循环到 N-1，试除判断，复杂度 O(N)
- 算法二：从 2 循环到 $\frac{N}{2}$，复杂度 O(N)
- 算法三：从 2 循环到 $\sqrt{N}$，复杂度 O($\sqrt{N}$)
- 算法四：提前打好 $\sqrt{N}$ 以内的质数表，只试除 $\sqrt{N}$ 以内的质数，复杂度 O($\frac{\sqrt{N}}{logN}$)。需要特别注意，打表不能超过最大文件大小
- 如果对素数判定感兴趣，可以自行学习 Miller-Rabin 算法

素数筛选

问题：输入一个正整数 N，输出 [1,N] 内的质数

- 算法一：直接用上页的方法判断每个数字是否是质数，时间复杂度 $O(N\frac{\sqrt{N}}{logN})$
- 算法二：用一个数组 prime[i] 来表示 i 是否是质数，i 从 2 循环到 n，对每个 i，把 i 的倍数置为合数，这个做法复杂度多少?

素数筛选

问题：输入一个正整数 N，输出 [1,N] 内的质数

- 算法一：直接用上页的方法判断每个数字是否是质数，时间复杂度 $O(N\frac{\sqrt{N}}{logN})$
- 算法二：用一个数组 prime[i] 来表示 i 是否是质数，i 从 2 循环到 n，对每个 i，把 i 的倍数置为合数，这个做法复杂度多少?
- 枚举次数为 $\frac{N}{2}+\frac{N}{3}+\frac{N}{4}+...+\frac{N}{N}$，复杂度为 O(NlogN)
- 这个算法除了用于筛选质数，还能用来枚举前 N 个正整数的的因数
- 对任意正整数 j($1 \leq j \leq N$)，如果 i 是 j 的因数且 $i \neq j, i \neq 1$，则循环到 i 时会把 j 筛去

素数筛选

问题：输入一个正整数 N，输出 [1,N] 内的质数

- 算法一：直接用上页的方法判断每个数字是否是质数，时间复杂度 $O(N\frac{\sqrt{N}}{logN})$
- 算法二：用一个数组 prime[i] 来表示 i 是否是质数，i 从 2 循环到 n，对每个 i，把 i 的倍数置为合数，这个做法复杂度多少?
- 枚举次数为 $\frac{N}{2}+\frac{N}{3}+\frac{N}{4}+...+\frac{N}{N}$，复杂度为 O(NlogN)
- 这个算法除了用于筛选质数，还能用来枚举前 N 个正整数的的因数
- 对任意正整数 $j(1 \le j \le N)$，如果 i 是 j 的因数且 $i \neq j, i \neq 1$，则循环到 i 时会把 j 筛去
- 算法三：当循环到 i 时，如果 i 没有被筛去，则 i 一定是质数。而我们只需要筛去质数的倍数，就能筛掉所有合数了
- 枚举次数为 $\frac{N}{2}+\frac{N}{3}+\frac{N}{5}+\frac{N}{7}...$，复杂度为 O(NloglogN)

素数筛选

- 上述算法有许多冗余，如 i=2 和 i=3 时都筛了 6。有没有一种每个合数只会被筛一次的算法呢?
- 在线性筛（欧拉筛）中，每个合数只会被它的最小质因子筛去一次
- 筛 8 需要 2*4，筛 12 需要 2*6，筛 15 需要 3*5
- 性质：每个合数的最小质因子一定小于等于次小质因子
- 在筛法过程中，如果 i 没有被筛去，我们可以把 i 加入质数表
- 而对所有 i（无论是质数还是合数)，我们枚举质数表里的质数 j 作为被筛数的最小质因子，直接筛掉 i*j
- 当 i 是 j 的倍数时，在筛除 i*j 以后退出筛法循环

素数筛选

```
for(int i=2;i<=n;i++){
    if(prime[i]) list[np++]=i;
    for(int j=0;j<np&&i*list[j]<=n;j++){
        prime[i*list[j]]=false;
        if(i%list[j]==0)break;
    }
}
```

- 由于每个合数都只会被筛去一次，所以内层循环的总执行次数是 O(N) 的
- 线性筛算法复杂度是 O(N) 的

素数筛选

让我们以筛选 [2,13] 内的质数为例，理解线性筛的执行过程

j	0	1	2	3	4	5
list	2					

i	2	3	4	5	6	7
prime	true	true	false	true	true	true
	8	9	10	11	12	13
	true	true	true	true	true	true

素数筛选

j	0	1	2	3	4	5
list	2	3				

i	2	3	4	5	6	7
prime	true	true	false	true	false	true
	8	9	10	11	12	13
	true	false	true	true	true	true

素数筛选

j	0	1	2	3	4	5
list	2	3				

i	2	3	4	5	6	7
prime	true	true	false	true	false	true
	8	9	10	11	12	13
	false	false	true	true	true	true

素数筛选

j	0	1	2	3	4	5
list	2	3	5			

i	2	3	4	5	6	7
prime	true	true	false	true	false	true
	8	9	10	11	12	13
	false	false	false	true	true	true

素数筛选

j	0	1	2	3	4	5
list	2	3	5			

i	2	3	4	5	6	7
prime	true	true	false	true	false	true
	8	9	10	11	12	13
	false	false	false	true	false	true

素数筛选

j	0	1	2	3	4	5
list	2	3	5	7		

i	2	3	4	5	6	7
prime	true	true	false	true	false	true
	8	9	10	11	12	13
	false	false	false	true	false	true

后面的循环依次把 11、13 加入素数表，过程简单，略去

区间上的素数筛选

问题：输入 L、R，输出 [L,R] 内的质数。其中
$0 \le D = R - L \le 10^6, 0 \le L \le R \le 10^{12}$

区间上的素数筛选

问题：输入 L、R，输出 [L,R] 内的质数。其中
$0 \le D = R - L \le 10^6, 0 \le L \le R \le 10^{12}$

- 容易看出，我们只需要用 $\sqrt{R}$ 以内的质数筛去 [L,R] 内的合数，就能解决这个问题了
- 对于质数 p，它在 [L,R] 内的倍数有 $p(\lfloor \frac{L-1}{p} \rfloor + 1), p(\lfloor \frac{L-1}{p} \rfloor + 2), ..., p(\lfloor \frac{R}{p} \rfloor)$，个数为 $\lfloor \frac{D+1}{p} \rfloor$
- 这个做法的复杂度是多少?

区间上的素数筛选

问题：输入 L、R，输出 [L,R] 内的质数。其中 $0 \le D = R - L \le 10^6, 0 \le L \le R \le 10^{12}$

- 容易看出，我们只需要用 $\sqrt{R}$ 以内的质数筛去 [L,R] 内的合数，就能解决这个问题了
- 对于质数 p，它在 [L,R] 内的倍数有 $p(\lfloor \frac{L-1}{p} \rfloor + 1), p(\lfloor \frac{L-1}{p} \rfloor + 2), ..., p(\lfloor \frac{R}{p} \rfloor)$，个数为 $\lfloor \frac{D+1}{p} \rfloor$
- 这个做法的复杂度是多少?
- 执行次数约为 $\frac{D}{2} + \frac{D}{3} + \frac{D}{5} + \frac{D}{7} + ...$，总时间复杂度 $O(DloglogR + \sqrt{R})$

积性函数的计算

在上午，我们介绍了使用分解质因数的方法计算积性函数的方法。分解质因数的方法只适用于少量输入的情况。现在，我们考虑在 [1,N] 上计算积性函数 f(x)

一个很好的例子是计算前 N 个正整数的因数个数

- 若 $a = p_1^{a_1} p_2^{a_2} ... p_n^{a_n}$，则 a 的因数个数为 $\prod_{i=1}^{n}(a_i + 1)$

积性函数的计算

在上午，我们介绍了使用分解质因数的方法计算积性函数的方法。分解质因数的方法只适用于少量输入的情况。现在，我们考虑在 [1,N] 上计算积性函数 f(x)

一个很好的例子是计算前 N 个正整数的因数个数

- 若 $a = p_1^{a_1} p_2^{a_2} \dots p_n^{a_n}$，则 a 的因数个数为 $\prod_{i=1}^{n}(a_i + 1)$
- 朴素的算法是，对每个质数 p，枚举它的倍数，计算 p 在倍数中的次数，乘到倍数的答案里
- 这个做法的复杂度是 O(NloglogN)
- $\frac{1}{p} + \frac{1}{p^2} + \frac{1}{p^3} + \dots = \frac{1}{p-1}$
- 是不是很像普通的筛法？用线性筛的思路能加速计算吗？

积性函数的计算

- 在线性筛里，每个合数只会被筛去一次，在筛去这个合数时，可以算出这个合数的因数个数吗?
- 设当前循环到 i，枚举的质数为 j，我们需要筛去 i*j，i 的因数个数为 f(i)
- 那么如果 i 不是 j 的倍数，说明 i*j 中 j 的次数为 1，所以 f(i*j)=f(i)*2
- 如果 i 是 j 的倍数，要怎么做?

积性函数的计算

- 在线性筛里，每个合数只会被筛去一次，在筛去这个合数时，可以算出这个合数的因数个数吗?
- 设当前循环到 i，枚举的质数为 j，我们需要筛去 i*j，i 的因数个数为 f(i)
- 那么如果 i 不是 j 的倍数，说明 i*j 中 j 的次数为 1，所以 f(i*j)=f(i)*2
- 如果 i 是 j 的倍数，要怎么做?
- 我们另外维护一个数组 m(i)，表示 i 的最小质因子的次数
- 如果 i 是 j 的倍数，那么 j 一定是 i 的最小质因子，于是 f(i*j)=f(i)*$\frac{m(i)+2}{m(i)+1}$
- 我们还需要递推出 m(i*j)
- 如果 i 不是 j 的倍数，m(i*j)=1
- 如果 i 是 j 的倍数，m(i*j)=m(i)+1

积性函数的计算

- 对于其他积性函数，也有类似的算法，留给同学们思考
- 一个问题是，如果我们要计算 [1,N] 内的 M 个数的积性函数，该采用质因子分解法还是筛法呢?

积性函数的计算

- 对于其他积性函数，也有类似的算法，留给同学们思考
- 一个问题是，如果我们要计算 [1,N] 内的 M 个数的积性函数，该采用质因子分解法还是筛法呢？
- 采用质因子分解法，复杂度为 O(M$\sqrt{N}$)
- 采用筛法，复杂度为 O(M+N)
- 当 M>N 时，要使用筛法，复杂度为 O(M)
- 当 M≤N 时，筛法复杂度为 O(N)，假设两种方法的常数相同
- 则质因子分解法更优的条件是 $M\sqrt{N} \leq N$
- 综上，当 $M \leq k\sqrt{N}$ 时，采用质因子分解法
- 当 $M > k\sqrt{N}$ 时，采用筛法。其中 k 是某个常数

线性筛加速质因数分解

- 假设我们要将 $1 \sim N$ 的数全部质因数分解
- 我们可以通过线性筛求出每个数 a 的最小质因子 q[a]
- 然后做一遍从 1 到 N 的循环
- 循环到 i 时，它的最小质因子是 q[i]，它的次小质因子是 $q[\frac{i}{q[i]}]$，以此类推
- 我们得到了一个 O(N) 预处理，单次 O(logN) 计算的质因数分解算法

例题讲解-HAOI2012 外星人

输入 N，输出最小的 x，使得将 N 取 x 次欧拉函数后，结果为 1

输入以 $N = \prod_{i=1}^{m} p_i^{q_i}$ 的形式给出，$1 \leq p_i \leq 10^5 \quad 1 \leq q_i \leq 10^9$，
共有 test 组测试数据，$test \leq 50$

样例输入

1
2
2 2
3 1

样例输出

3

输入有一组数据 N=12，$\varphi(12) = 4, \varphi(4) = 2, \varphi(2) = 1$

例题讲解-HAOI2012 外星人

- 对于 $N=\prod_{i=1}^{m} p_i^{q_i}$，$\varphi(N)=\prod_{i=1}^{m}(p_i-1)p_i^{q_i-1}$
- 观察到如果 $N=2^k$，则答案为 k，每次计算后 2 的幂次-1
- 如果 N 还有除 2 之外的质因子，每次计算后 2 的幂次至少 +1
- 所以在 N 变成 2 的幂次之前，2 的幂次是不降的
- 答案是求解过程中产生 2 的个数！（N 为奇数时需要 +1）
- 我们只要计算每个数产生的 2 的个数就行了，设 p 产生的 2 的个数是 f(p)，特别的 f(2)=1
- 若 p 为质数，则 f(p)=f(p-1)
- 若 p 为合数，设 p=m*k，m 是 p 的最小质因子，则 f(p)=f(m)+f(k)
- 最后答案为 $\sum_{i=1}^{m} q_i * f(p_i)$，当 N 为奇数时，答案再 +1

裴蜀定理

裴蜀定理

若 a、b 是整数，且 gcd(a,b)=d，那么对任意整数 x、y，d|ax+by；并且，存在整数 x,y，使得 ax+by=d

证明

- 0. 若 a、b 中有负数，可以取相反数后考虑
- 1. 因为 gcd(a,b)=d，所以 d|a 且 d|b，所以 d|ax+by
- 2. 考虑辗转相减计算 a、b 的最大公因数的过程，用 $s_1a + s_2b$ 和 $t_1a + t_2b$ 来表示两个计算数，两个计算数始终都是 a、b 的线性组合。最后一个计算数一定是 d，所以存在整数 x,y，使得 ax+by=d

一些结论

- 对整数 a、b，a 与 b 互质等价于存在整数 x、y，使得 ax+by=1
- 对整数 a、b，若 gcd(a,b)=d，则 gcd($\frac{a}{d}$,$\frac{b}{d}$)=1
- 如果 (x_0, y_0) 是 ax+by=d=gcd(a,b) 的一组解，那么 $(x_0 + k\frac{b}{d}, y_0 - k\frac{a}{d}) \quad k \in Z$ 是 ax+by=d 的通解

一些结论

- 对整数 a、b，a 与 b 互质等价于存在整数 x、y，使得 ax+by=1
- 对整数 a、b，若 gcd(a,b)=d，则 gcd($\frac{a}{d}$,$\frac{b}{d}$)=1
- 如果 (x_0, y_0) 是 ax+by=d=gcd(a,b) 的一组解，那么 $(x_0 + k\frac{b}{d}, y_0 - k\frac{a}{d}) \quad k \in Z$ 是 ax+by=d 的通解
- 首先，将通解式带入原方程，等式成立
- 然后，考虑 ax+by=d 的两个不同解 (x_1, y_1) 和 (x_2, y_2)，我们有 $a(x_1 - x_2) + b(y_1 - y_2) = 0$，那么 $\frac{a}{d}(x_1 - x_2) + \frac{b}{d}(y_1 - y_2) = 0$。因为 gcd($\frac{a}{d}$,$\frac{b}{d}$)=1，所以 $\frac{b}{d}|(x_1 - x_2)$ 且 $\frac{a}{d}|(y_1 - y_2)$

扩展欧几里得算法

问题：给出正整数 a、b，求一组 x、y，使得 $ax+by=gcd(a,b)$

扩展欧几里得算法

问题：给出正整数 a、b，求一组 x、y，使得 ax+by=gcd(a,b)

- 我们在证明裴蜀定理时，就已经得到了一个算法了
- 记录两个操作数 u、v 关于 a、b 的系数，到 v=0 时，u 关于 a、b 的系数就是原方程的解
- 算法流程 (设 $u = s_1 a + s_2 b, v = t_1 a + t_2 b$):
- 0. 初始设置 $u = a, v = b, s_1 = 1, s_2 = 0, t_1 = 0, t_2 = 1$
- 1. 当 v=0 时，u=gcd(a,b)，方程的解为 $x = s_1, y = s_2$
- 2. 计算 $w = u\%v$，则 $w = u - \lfloor \frac{u}{v} \rfloor v$
- 3. 令 $s_1' = t_1, s_2' = t_2, t_1' = s_1 - \lfloor \frac{u}{v} \rfloor t_1, t_2 = s_2 - \lfloor \frac{u}{v} \rfloor t_2$
- 4. 用 $(v, w, s_1', s_2', t_1', t_2')$ 代替 $(u, v, s_1, s_2, t_1, t_2)$，转到 2
- 这个做法时间复杂度 O(log(max(a,b)))，还有其他做法吗?

扩展欧几里得算法

- 上述算法是基于递推的做法，在求解这类方程时我们通常采用基于递归的扩展欧几里得算法。
- 扩展欧几里得算法的思路是，在辗转相除之后，计算 $ux+vy=gcd(a,b)$ 的解。当 v=0 时，显然有 x=1,y=0 的解
- 设当前递归函数为 (u,v)，则它会递归到 (v,w)，其中 $w=u-\lfloor\frac{u}{v}\rfloor v$，从递归能返回 $vx+wy=gcd(a,b)$ 的解，不妨设解为 x_0,y_0
- 那么我们有 $vx_0+(u-\lfloor\frac{u}{v}\rfloor v)y_0=gcd(a,b)$
- 即 $uy_0+v(x_0-\lfloor\frac{u}{v}\rfloor y_0)=gcd(a,b)$
- 我们得到了 $ux+vy=gcd(a,b)$ 的一组解，返回给上一层
- 这个做法的复杂度也是 O(log(max(a,b)))

扩展欧几里得算法

- 需要注意的是，这两种做法都不能保证最终 (x,y) 的位置，如果要找到满足条件的解，需要根据通解公式进一步计算
- 扩展欧几里得算法可以用来算逆元：设我们要找 a 在模 p 下的逆元，则只要求 $ax-py=1$ 的解即可
- 如果我们给定正整数 a、b、c，求 $ax+by=c$ 的解，要怎么做?

扩展欧几里得算法

- 需要注意的是，这两种做法都不能保证最终 (x,y) 的位置，如果要找到满足条件的解，需要根据通解公式进一步计算
- 扩展欧几里得算法可以用来算逆元：设我们要找 a 在模 p 下的逆元，则只要求 ax-py=1 的解即可
- 如果我们给定正整数 a、b、c，求 ax+by=c 的解，要怎么做?
- 根据裴蜀定理 d=gcd(a,b)|c，否则一定无解。我们计算 e=$\frac{c}{d}$
- 若 (x_0, y_0) 是 $ax + by = d$ 的一组解，则 (ex_0, ey_0) 一定是 $ax + by = c$ 的一组解
- 设 $x_1 = ex_0, y_1 = ey_0$，则 $ax + by = c$ 的通解为 $(x_1 + k\frac{b}{d}, y_1 - k\frac{a}{d}) \quad k \in Z$

扩展欧几里得算法

```
int exgcd(int u,int v,int& x,int& y){
    if(v==0){
        x=1;
        y=0;
        return u;
    }
    int ret=exgcd(v,u%v,x,y);
    int t=x;
    x=y;
    y=t-u/v*y;
    return ret;
}
```

扩展欧几里得算法函数如上所示

扩展中国剩余定理

- 上午的部分，介绍了中国剩余定理
- 其中有一个约束是，$m_1, m_2, ..., m_n$ 两两互质
- 如果我们去掉这一限制，要怎样求解以下的方程组?

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ ... \\ x \equiv a_n \pmod{m_n} \end{cases}$$

扩展中国剩余定理

- 设 M=lcm$(m_1, m_2, ..., m_n)$，容易发现，如果 x 是解，则 $x + kM, \quad k \in Z$ 都是解，且 [x+1,x+M-1] 内没有解
- 考虑 n=2 的情况，我们有两个方程 $x = k_1 m_1 + a_1$ 和 $x = k_2 m_2 + a_2$
- 联立可得 $k_1 m_1 - k_2 m_2 = a_2 - a_1$
- 设 m=gcd(m_1, m_2)，则 m$|(a_2 - a_1)$，否则无解，那么
- $k_1 \frac{m_1}{m} - k_2 \frac{m_2}{m} = \frac{a_2 - a_1}{m}$，在模 $\frac{m_2}{m}$ 意义下，有
- $k_1 \equiv inv(\frac{m_1}{m}) \frac{a_2 - a_1}{m} (mod \ \frac{m_2}{m})$，即
- $k_1 = inv(\frac{m_1}{m}) \frac{a_2 - a_1}{m} + y \frac{m_2}{m} \quad y \in Z$，带入原式，有
- $x = inv(\frac{m_1}{m}) \frac{(a_2 - a_1) m_1}{m} + y \frac{m_1 m_2}{m} + a_1 \quad y \in Z$
- 所以 $x \equiv inv(\frac{m_1}{m}) \frac{(a_2 - a_1) m_1}{m} + a_1 (mod \ \frac{m_1 m_2}{m})$

本段推导参考了
https://www.cnblogs.com/zwfymqz/p/8425731.html

扩展中国剩余定理

- 因为上述推导是可逆的，所以这一个方程的解和原方程组的解是等价的。为了加深理解，我们试试把 x 带回原方程中，这时同余式都成立吗

扩展中国剩余定理

- 因为上述推导是可逆的，所以这一个方程的解和原方程组的解是等价的。为了加深理解，我们试试把 x 带回原方程中，这时同余式都成立吗
- 对于 $x = k_1 m_1 + a_1$，显然成立
- 对于 $x = k_2 m_2 + a_2$，因为 $m_2 | (inv(\frac{m_1}{m})m_1 - m)$，所以成立
- 在 n=2 时，可以把两个方程合并成一个方程，当 n>2 时，我们只需要一遍循环，两两合并，就可以算出最终的通解，时间复杂度 O(nlogM)
- 扩展中国剩余定理有几个要点
- 在模 $\frac{m_2}{m}$ 的意义下，$\frac{m_1}{m}$ 的逆元一定存在，我们可以用欧拉定理、逆元递推或扩展欧几里得法求得
- 计算中乘法容易越界，必要时可以采用快速乘

排列

排列

从 n 个不同的元素中不重复地取出 m 个元素，按照顺序排成一列，称作从 n 个元素中取出 m 个元素的一个排列
从 n 个不同元素中取出 m 个不同元素的不同排列的个数叫做排列数，记为 A_n^m

- $A_n^m = n(n-1)...(n-m+1) = \frac{n!}{(n-m)!}$
- 特别地 $A_n^0 = 1, A_n^n = n!$
- 假如我们要计算 A_n^m，我们可以直接从 n 循环到 n-m+1，将答案乘出
- 圆排列：从 n 个不同元素中不重复地取出 m 个元素放在圆周。当两个圆排列旋转之后对应相同时，认为这两个圆排列相同。
- 圆排列个数为 $\frac{A_n^m}{m}$，取模计算时需要计算 m 的逆元

组合

组合

从 n 个不同的元素中一次取出 m 个不重复的元素为一组，称作从 n 个元素中取出 m 个元素的一个组合
从 n 个不同元素中取出 m 个不同元素的不同组合的个数叫做组合数，记为 C_n^m

- $C_n^m = \frac{A_n^m}{A_m^m} = \frac{n(n-1)..(n-m+1)}{m(m-1)..1} = \frac{n!}{(n-m)!m!}$
- 特别地 $C_n^0 = C_n^n = 1$
- 计算组合数的第一个方法是，按照上述公式，先乘后除计算
- 组合数有许多性质，如 $C_n^m = C_{n-1}^{m-1} + C_{n-1}^m$，考虑第 n 个元素是否选取，就能证明这一公式
- 此外 $C_n^m = C_n^{n-m}$

杨辉三角

1					
1	1				
1	2	1			
1	3	3	1		
1	4	6	4	1	
1	5	10	10	5	1

- 杨辉三角如上图所示，满足 T[n][0]=T[n][n]=1，当 $n > 1, 0 < m < n$ 时 T[n][m]=T[n-1][m-1]+T[n-1][m]
- 由于递推的形式相同，T[n][m]=C_n^m，我们得到了第二个计算组合数的方法，时间复杂度 O(nm)
- 和直接乘除相比，这个方法更适合做大量组合数计算

二项式定理

二项式定理

$$(x+y)^n = C_n^0 x^n + C_n^1 x^{n-1} y + \dots + C_n^n y^n$$

- 二项式定理说的是，杨辉三角的第 i 行又依次是 $(x+y)^i$ 的展开式的各项系数，为什么？

二项式定理

二项式定理

$(x+y)^n = C_n^0 x^n + C_n^1 x^{n-1} y + \ldots + C_n^n y^n$

- 二项式定理说的是，杨辉三角的第 i 行又依次是 $(x+y)^i$ 的展开式的各项系数，为什么？
- 使用数学归纳法，当 i=1 时 $x+y = C_1^0 x + C_1^1 y$
- 如果 i 时成立，考虑 i+1 时，$(x+y)^{i+1} = (x+y)^i (x+y)$
- $= (C_i^0 x^i + C_i^1 x^{i-1} y + \ldots + C_i^i y^i)(x+y)$
- $= C_i^0 x^{i+1} + (C_i^1 + C_i^0) x^i y + \ldots + (C_i^i + C_i^{i-1}) x y^i + C_i^i y^{i+1}$
- $= C_{i+1}^0 x^{i+1} + C_{i+1}^1 x^i y + \ldots + C_{i+1}^{i+1} y^{i+1}$

二项式定理的推论

- 二项式定理中，x 和 y 都可以任取，可以得出许多恒等式
- 令 x=1，则有 $(1+y)^n = C_n^0 y^0 + C_n^1 y^1 + C_n^2 y^2 + \ldots + C_n^n y^n$
- 令 x=y=1，则有 $2^n = C_n^0 + C_n^1 + \ldots + C_n^n$
- 令 x=1,y=2，则有 $3^n = C_n^0 2^0 + C_n^1 2^1 + C_n^2 2^2 + \ldots + C_n^n 2^n$
- 复杂度是 $O(3^n)$ 的题目，一般是用二项式定理推出来的
- 令 $y = \frac{1}{x}$，则有 $(x + \frac{1}{x})^n = C_n^0 x^n + C_n^1 x^{n-2} + \ldots + C_n^n x^{-n}$

例题讲解-NOIP2011 计算系数

题目大意：给定多项式 $(ax+by)^k$，求多项式展开后 $x^n y^m$ 的系数，保证 n+m=k，输出对 10007 取模的结果

输入分别为 a、b、k、n、m，$0 \le k \le 10^3; 0 \le a, b \le 10^6$

样例输入	样例输出
1 1 3 1 2	3

$(x+y)^3 = x^3 + 3x^2y + 3xy^2 + y^3$

例题讲解-NOIP2011 计算系数

- 当 a=b=1 时，我们只需要计算 C_k^n 即可
- 因为 10007 是质数，所以我们可以在计算分母的逆元后，用乘除直接计算
- 因为 $0 \leq k \leq 10^3$，所以还可以用杨辉三角递推
- 当 a 和 b 不全为 1 时，因为 a 和 x 绑在一起，b 和 y 在一起，答案是 $a^n b^m C_k^n$
- 当我们把数据范围调大，如 $0 \leq k \leq 10^6$，就不能用杨辉三角计算了，此时能用逆元计算吗?

例题讲解-NOIP2011 计算系数

- 当 a=b=1 时，我们只需要计算 C_k^n 即可
- 因为 10007 是质数，所以我们可以在计算分母的逆元后，用乘除直接计算
- 因为 $0 \le k \le 10^3$，所以还可以用杨辉三角递推
- 当 a 和 b 不全为 1 时，因为 a 和 x 绑在一起，b 和 y 在一起，答案是 $a^n b^m C_k^n$
- 当我们把数据范围调大，如 $0 \le k \le 10^6$，就不能用杨辉三角计算了，此时能用逆元计算吗?
- 分母里可能就有 10007 的倍数，但最终结果里未必有 10007，所以用逆元计算时需要单独考虑 10007 的倍数。

组合数计算

- 当我们的模数不再是质数时，组合数的分母需要考虑的情况更加复杂
- 有针对非质数模数的组合数计算算法吗?

组合数计算

- 当我们的模数不再是质数时，组合数的分母需要考虑的情况更加复杂
- 有针对非质数模数的组合数计算算法吗?
- 我们可以考虑最终答案的质因数分解
- 对每个质数 p，它在最终答案的质因数分解中的次数为 p 在 n! 中的次数-p 在 m! 中的次数-p 在 (n-m)! 中的次数
- p 在 n! 中的次数要怎么算?

组合数计算

- 当我们的模数不再是质数时，组合数的分母需要考虑的情况更加复杂
- 有针对非质数模数的组合数计算算法吗?
- 我们可以考虑最终答案的质因数分解
- 对每个质数 p，它在最终答案的质因数分解中的次数为 p 在 n! 中的次数-p 在 m! 中的次数-p 在 (n-m)! 中的次数
- p 在 n! 中的次数要怎么算?
- p 在 n! 中的次数 $=\lfloor\frac{n}{p}\rfloor+\lfloor\frac{n}{p^2}\rfloor+\lfloor\frac{n}{p^3}\rfloor+\dots$
- 我们枚举每个 n 以内的质数，计算它在最终分解式中的次数，然后使用快速幂计算答案即可
- 时间复杂度 O($\frac{n}{ln(n)}log_2 n$)=O(n)

Lucas 定理

Lucas 定理

设 n=sp+q, m=tp+r($0 \leq q, r < p$)，则 $C_n^m \equiv C_s^t C_q^r (mod\ p)$

- 当 $0 \leq n, m \leq 10^{18}, 0 < p \leq 10^6$ 时，若 p 为质数，要如何计算组合数 C_n^m

Lucas 定理

Lucas 定理

设 n=sp+q, m=tp+r($0 \le q, r < p$)，则 $C_n^m \equiv C_s^t C_q^r (mod\ p)$

- 当 $0 \le n, m \le 10^{18}, 0 < p \le 10^6$ 时，若 p 为质数，要如何计算组合数 C_n^m
- 根据 Lucas 定理，我们只要将 (n,m) 不断除 p，就可以把答案转化成若干 p 以内组合数的乘积
- 一个细节问题是有可能 $q < r$，此时定义 $C_q^r = 0$ 即可
- 如果感兴趣，可以自行寻找 Lucas 定理的证明，还可以学习扩展 Lucas 定理

组合数的应用

- 从 (0,0) 走到 (n,m)，每次只能向右或向上，问共有几种方式?

组合数的应用

- 从 (0,0) 走到 (n,m)，每次只能向右或向上，问共有几种方式?
- 从 (0,0) 走到 (n,m)，共走 n+m 步，其中 n 步向右，m 步向上
- 从 n+m 中取 n 步向右，剩下的向上，方案数为 C_{n+m}^{n}

组合数的应用

- 从 (0,0) 走到 (n,m)，每次只能向右或向上，问共有几种方式?
- 从 (0,0) 走到 (n,m)，共走 n+m 步，其中 n 步向右，m 步向上
- 从 n+m 中取 n 步向右，剩下的向上，方案数为 C_{n+m}^{n}
- 现有 r 个相同的盒子和 n 个互不相同的球，将这 n 个球放入 r 个盒子中，不允许有空盒，共有多少种放法?

组合数的应用

- 从 (0,0) 走到 (n,m)，每次只能向右或向上，问共有几种方式?
- 从 (0,0) 走到 (n,m)，共走 n+m 步，其中 n 步向右，m 步向上
- 从 n+m 中取 n 步向右，剩下的向上，方案数为 C_{n+m}^{n}
- 现有 r 个相同的盒子和 n 个互不相同的球，将这 n 个球放入 r 个盒子中，不允许有空盒，共有多少种放法?
- 设 f(n,r) 为答案，则 f(n,r)=r*f(n-1,r)+f(n-1,r-1)，可以用递推算出答案，复杂度 O(nr)
- 事实上，f(n,r) 是第二类 Stirling 数，它的通项 f(n,r)=$\frac{1}{r!}\sum_{k=0}^{r}(-1)^k C_r^k (r-k)^n$
- 使用通项计算，复杂度降为 O(rlogn)

Catalan 数

Catalan 数

规定定义域为自然数的满足以下递推式的数列为 Catalan 数：
$f(n) = \sum_{i=0}^{n-1} f(i)f(n-i-1)$，边界条件为 f(0)=1

- 可以验证，f(n) 也满足 $f(n) = \frac{f(n-1)(4n-2)}{n+1}$
- 所以 $f(n) = \frac{C_{2n}^{n}}{n+1} = C_{2n}^{n} - C_{2n}^{n-1}$
- 如果对这些公式的推导感兴趣，可以自己寻找资料
- 我们把计算 Catalan 数化为计算组合数
- Catalan 有什么应用呢?

Catalan 数的应用

- 二叉树形态计数：n 个结点的二叉树共有多少种不同的形态?

Catalan 数的应用

- 二叉树形态计数：n 个结点的二叉树共有多少种不同的形态?
- 设 f(n) 为 n 个结点的二叉树的形态数，则除根节点外，剩下的 n-1 个点在左子树或右子树，所以 $f(n) = \sum_{i=0}^{n-1} f(i)f(n-i-1)$，是 Catalan 数
- 有一个凸 n 边形，可以用 n-3 条不相交的对角线将 n 边形分成 n-2 个三角形，求一共有多少种分法?

Catalan 数的应用

- 二叉树形态计数：n 个结点的二叉树共有多少种不同的形态?
- 设 f(n) 为 n 个结点的二叉树的形态数，则除根节点外，剩下的 n-1 个点在左子树或右子树，所以 $f(n)=\sum_{i=0}^{n-1} f(i)f(n-i-1)$，是 Catalan 数
- 有一个凸 n 边形，可以用 n-3 条不相交的对角线将 n 边形分成 n-2 个三角形，求一共有多少种分法?
- 在 n 边形里任取一条边，则这条边所在的三角形的第三个顶点可以是剩下的 n-2 个顶点之一，故 ans(n)=$\sum_{i=2}^{n-1} ans(i)ans(n-i+1)$，所以 ans(n)=f(n-2)
- 还有许多问题，答案也是 Catalan 数
- 有 $1 \sim n$ 共 n 个数依次进栈，每个数随时可以出栈，不同出栈顺序的数量为 f(n)
- 长度为 2n 的由'('、')' 组成的序列，合法的括号序列（任意前缀的左括号不少于右括号）个数为 f(n)

组合计数原理

- 加法原理：如果事件 A 有 p 种产生方式，事件 B 有 q 种产生方式，且 A 与 B 不重叠，那么事件 A 或 B 有 p+q 种产生方式
- 例如，袋子里有 p 个红球，q 个绿球，则袋子里有 p+q 个红色或绿色的球
- 袋子里有 p 个红球，q 个木球，我们不知道袋子里有几个红色或木质的球

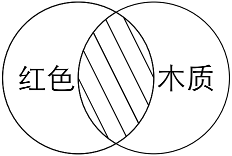
组合计数原理

- 加法原理：如果事件 A 有 p 种产生方式，事件 B 有 q 种产生方式，且 A 与 B 不重叠，那么事件 A 或 B 有 p+q 种产生方式
- 例如，袋子里有 p 个红球，q 个绿球，则袋子里有 p+q 个红色或绿色的球
- 袋子里有 p 个红球，q 个木球，我们不知道袋子里有几个红色或木质的球
- 乘法原理：如果事件 A 有 p 种产生方式，事件 B 有 q 种产生方式，且 A 与 B 独立，那么事件 A 与 B 有 pq 种产生方式
- 例如，从 x 地到 y 地有 p 种路线，从 y 地到 z 地有 q 种路线，则从 x 地到 y 地再到 z 地有 pq 条路线
- 如果从 x 地到 z 地还有 r 条直通路线，则从 x 地到 z 地共有 pq+r 条路线

容斥原理

袋子里有 p 个红球，q 个木球，r 个红色木球，则袋子里有 p+q-r 个红色的或木质的球

如果袋子里共有 s 个球，则袋子里有 s-p-q+r 个既不红又非木质的球



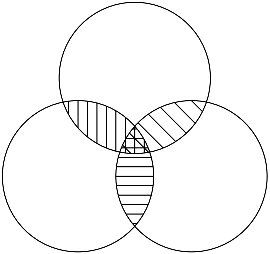
- 设球的集合是 S，红球的集合是 A，木球的集合是 B，则
- $|S|=s,|A|=p,|B|=q,|A\cap B|=r$
- $|A\cup B|=|A|+|B|-|A\cap B|$
- $|A^c\cap B^c|=|S|-|A\cup B|$

容斥原理

容斥原理

设 S 为有限集，$A_i \subseteq S(i=1,2,...,n)$，则
$|\bigcup_{i=1}^{n} A_i| = \sum_{k=1}^{n}(-1)^{k-1}\sum_{1\le i_1<i_2<...<i_k\le n}|\bigcap_{j=1}^{k} A_{i_j}|$

- 常和容斥原理一同使用的公式是 $|\bigcap_{i=1}^{n} A_i^c| = |S| - |\bigcup_{i=1}^{n} A_i|$
- 当 n=3 时，容斥原理公式为 $|A_1 \cup A_2 \cup A_3| = |A_1|+|A_2|+|A_3|-|A_1\cap A_2|-|A_1\cap A_3|-|A_2\cap A_3|+|A_1\cap A_2\cap A_3|$



容斥原理

- $1 \sim 120$ 中，有几个数满足是 4 的倍数或是 6 的倍数或是 9 的倍数?

容斥原理

- $1 \sim 120$ 中，有几个数满足是 4 的倍数或是 6 的倍数或是 9 的倍数?
- 设 $S = \{1, 2, ..., 120\}, A = \{4, 8, ..., 120\}, B = \{6, 12, ..., 120\}, C = \{9, 18, ..., 117\}$
- 答案是 $|A \cup B \cup C|$，考虑使用容斥原理
- $|A| = \frac{120}{4} = 30, |B| = \frac{120}{6} = 20, |C| = \lfloor \frac{120}{9} \rfloor = 13$
- $|A \cap B| = \frac{120}{12} = 10, |A \cap C| = \lfloor \frac{120}{36} \rfloor = 3$
- $|B \cap C| = \lfloor \frac{120}{18} \rfloor = 6, |A \cap B \cap C| = \lfloor \frac{120}{36} \rfloor = 3$
- 所以答案是 30+20+13-10-3-6+3=47

容斥原理的变种

问题：对于 $1 \le x, y \le N$，求 $\sum_{i=1}^{N}\sum_{j=1}^{N} gcd(i,j), 1 \le N \le 10^6$

容斥原理的变种

问题：对于 $1 \le x, y \le N$，求 $\sum_{i=1}^{N}\sum_{j=1}^{N} gcd(i,j), 1 \le N \le 10^6$

- 一个直接的思路是，枚举最大公因数 d，求 gcd(i,j)=d 的数量，不妨设 B_d 表示最大公因数为 d 的 (i,j) 的集合
- $|B_d|$ 等于 $\lfloor \frac{N}{d} \rfloor^2$ 吗?

容斥原理的变种

问题：对于 $1 \le x, y \le N$，求 $\sum_{i=1}^{N}\sum_{j=1}^{N} gcd(i,j), 1 \le N \le 10^6$

- 一个直接的思路是，枚举最大公因数 d，求 gcd(i,j)=d 的数量，不妨设 B_d 表示最大公因数为 d 的 (i,j) 的集合
- $|B_d|$ 等于 $\lfloor \frac{N}{d} \rfloor^2$ 吗?
- 当 d=3，i=12，j=18 时，gcd(i,j)=6
- 设 A_i 表示最大公因数是 i 的倍数的 (i,j) 数量，那么我们要求 $|\bigcup_{i=1}^{n} A_i|$，已知 $|A_i| = \lfloor \frac{N}{i} \rfloor^2$
- 这类问题常用的技巧是，设 $C_k = \bigcup_{i=k}^{n} A_i$，那么 $C_k = \bigcup_{i=k}^{n} B_i, |C_k| = \sum_{i=k}^{n} |B_i|$
- 显然，$|B_n| = |C_n| = 1$，我们要求的是 $|C_1|$，如何能递推得到?

容斥原理的变种

- 对 $1 \leq k < n$ 我们有 $C_k = A_k \cup C_{k+1}$，应用容斥原理
- $|C_k| = |A_k| + |C_{k+1}| - |A_k \cap C_{k+1}|(= |C_{k+1}| + |B_k|)$
- $|A_k|$ 和 $|C_{k+1}|$ 都已知，关键是我们要计算 $|A_k \cap C_{k+1}|$
- $|A_k \cap C_{k+1}| = |A_k \cap (\bigcup_{i=k+1}^{n} B_i)| = |\bigcup_{i=k+1}^{n} A_k \cap B_i|$
- 因为 B_i 互不相交，所以 $A_k \cap B_i$ 也互不相交，所以
- $|A_k \cap C_{k+1}| = \sum_{i=k+1}^{n} |A_k \cap B_i|$
- 而 $|A_k \cap B_i| = \begin{cases} |B_i| & k \mid i \\ 0 & k \nmid i \end{cases}$
- 所以 $|A_k \cap C_{k+1}| = \sum_{i=2}^{\lfloor \frac{n}{k} \rfloor} |B_{ik}|$
- 综上，$|B_k| = \lfloor \frac{N}{k} \rfloor^2 - \sum_{i=2}^{\lfloor \frac{n}{k} \rfloor} |B_{ik}|$
- 我们只需要一个二重循环，就能算出所有 $|B_k|$，然后把所有 $|B_k|$ 相加，算出答案
- 时间复杂度 O(nlogn)

抽屉原理（鸽巢原理）

抽屉原理

把 n+1 件物品放入 n 个抽屉，则至少有一个抽屉里放了两件或两件以上的物品；把 n-1 件物品放入 n 个抽屉，则至少有一个抽屉是空的

- 问题：给出一个整数序列 $a_1, a_2, ..., a_n$，从序列里不重复地取若干个数求和，是否能凑出 n 的倍数?

抽屉原理（鸽巢原理）

抽屉原理

把 n+1 件物品放入 n 个抽屉，则至少有一个抽屉里放了两件或两件以上的物品；把 n-1 件物品放入 n 个抽屉，则至少有一个抽屉是空的

- 问题：给出一个整数序列 $a_1, a_2, ..., a_n$，从序列里不重复地取若干个数求和，是否能凑出 n 的倍数？
- 取数的方案数共有 $2^n - 1$
- 事实上，题目条件太强，导致我们没有思路。即使把取数条件约束到一个区间，也能凑出 n 的倍数了
- 设 $b_i = \sum_{j=0}^{i} a_j$，那么我们得到了 $b_0, b_1, ..., b_n$ 共 n+1 个数
- 这 n+1 个数模 n 的值，必然有两个数相等，不妨设 $b_x \equiv b_y (mod\ n), x < y$，则 $\sum_{i=x+1}^{y} a_i$ 必然是 n 的倍数

例题讲解-SCOI2010 幸运数字

题目大意：定义只由 6、8 组成的整数为“幸运数字”，如 6、8、66、68、86、88。定义“幸运数字”的倍数为“近似幸运数字”。求 [a,b] 内的近似幸运数字数量,$0 \leq a \leq b \leq 10^{10}$

样例输入 1	样例输出 1
1 10	2

样例输入 2	样例输出 2
1234 4321	809

- 显然，我们需要使用容斥原理。
- 容斥原理中集合的数量为“幸运数字”的数量

例题讲解-SCOI2010 幸运数字

- 10^{10} 内的幸运数字有多少个?
- 位数为 10 的幸运数字有 2^{10} 个，位数为 9 的幸运数字有 2^9 个，位数为 8 的幸运数字有 2^8 个，...
- 幸运数字一共约有 2^{11} 个，如果我们应用容斥原理，看上去需要计算 $2^{2^{11}}$ 个数字，不可接受，但其实可行的集合很少

例题讲解-SCOI2010 幸运数字

- 10^{10} 内的幸运数字有多少个?
- 位数为 10 的幸运数字有 2^{10} 个，位数为 9 的幸运数字有 2^9 个，位数为 8 的幸运数字有 2^8 个，…
- 幸运数字一共约有 2^{11} 个，如果我们应用容斥原理，看上去需要计算 $2^{2^{11}}$ 个数字，不可接受，但其实可行的集合很少
- 容斥时我们要枚举这 2^{11} 个数字的子集 T，计算它的倍数的数量，如果 $|T|$ 是奇数，则加；如果 $|T|$ 是偶数，则减
- 枚举子集可以使用搜索，搜索每层枚举每个数字是否选择，如果选择，和之前的数字求最小公倍数。如何剪枝?

例题讲解-SCOI2010 幸运数字

- 10^{10} 内的幸运数字有多少个?
- 位数为 10 的幸运数字有 2^{10} 个，位数为 9 的幸运数字有 2^9 个，位数为 8 的幸运数字有 2^8 个，...
- 幸运数字一共约有 2^{11} 个，如果我们应用容斥原理，看上去需要计算 $2^{2^{11}}$ 个数字，不可接受，但其实可行的集合很少
- 容斥时我们要枚举这 2^{11} 个数字的子集 T，计算它的倍数的数量，如果 $|T|$ 是奇数，则加；如果 $|T|$ 是偶数，则减
- 枚举子集可以使用搜索，搜索每层枚举每个数字是否选择，如果选择，和之前的数字求最小公倍数。如何剪枝?
- 1. 如果最小公倍数 $>$b，显然不需要继续计算

例题讲解-SCOI2010 幸运数字

- 10^{10} 内的幸运数字有多少个?
- 位数为 10 的幸运数字有 2^{10} 个，位数为 9 的幸运数字有 2^9 个，位数为 8 的幸运数字有 2^8 个，...
- 幸运数字一共约有 2^{11} 个，如果我们应用容斥原理，看上去需要计算 $2^{2^{11}}$ 个数字，不可接受，但其实可行的集合很少
- 容斥时我们要枚举这 2^{11} 个数字的子集 T，计算它的倍数的数量，如果 $|T|$ 是奇数，则加；如果 $|T|$ 是偶数，则减
- 枚举子集可以使用搜索，搜索每层枚举每个数字是否选择，如果选择，和之前的数字求最小公倍数。如何剪枝?
- 1. 如果最小公倍数 $>$b，显然不需要继续计算
- 2. 一个幸运数字可以是另一个幸运数字的倍数，如 88 是 8 的倍数，可以直接把 88 去掉

例题讲解-SCOI2010 幸运数字

- 10^{10} 内的幸运数字有多少个?
- 位数为 10 的幸运数字有 2^{10} 个，位数为 9 的幸运数字有 2^9 个，位数为 8 的幸运数字有 2^8 个，...
- 幸运数字一共约有 2^{11} 个，如果我们应用容斥原理，看上去需要计算 $2^{2^{11}}$ 个数字，不可接受，但其实可行的集合很少
- 容斥时我们要枚举这 2^{11} 个数字的子集 T，计算它的倍数的数量，如果 $|T|$ 是奇数，则加；如果 $|T|$ 是偶数，则减
- 枚举子集可以使用搜索，搜索每层枚举每个数字是否选择，如果选择，和之前的数字求最小公倍数。如何剪枝?
- 1. 如果最小公倍数 $>b$，显然不需要继续计算
- 2. 一个幸运数字可以是另一个幸运数字的倍数，如 88 是 8 的倍数，可以直接把 88 去掉
- 3. 把数字从大到小排序再进行搜索
- 加上这些优化，就能通过这题了

一元多项式

一元多项式

设 $a_0, a_1, ..., a_n(a_n \neq 0)$ 是数域 F 内的数，n 是非负整数，那么表达式 $a_n x^n + a_{n-1}x^{n-1} + ... + a_2x^2 + a_1x + a_0$ 叫做数域 F 上的一个一元多项式。一元多项式通常被记做 f(x),g(x),...

- $a_n x^n$ 称为多项式的首项，n 叫做多项式的次数，a_0 叫做多项式的常数项
- $a_i x^i$ 称为多项式的 i 次项，a_i 为 i 次项系数
- 定义两个多项式 f(x),g(x) 相等为，f(x) 和 g(x) 的各次项系数对应相等
- 定理：对于两个次数不超过 n 的多项式 f(x),g(x)，如果对于变数 x 的 n+1 个不同的数都有相同的值，那么这两个多项式恒等

多项式乘法

- $f(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_2 x^2 + a_1 x + a_0$
- $g(x) = b_m x^m + b_{n-1} x^{n-1} + ... + b_2 x^2 + b_1 x + b_0$
- 设有两个多项式如上所示，f(x)g(x) 是多少?
- 使用乘法分配律，再合并同类项，可得
- $f(x)g(x) = \sum_{i=0}^{m+n} (\sum_{j=max(0,i-m)}^{min(i,n)} a_j b_{i-j}) x^i$
- 是不是很像高精度乘法?
- 事实上，多项式乘法的朴素算法和高精度乘法一样，复杂度为 O(nm)

面值问题

问题：有 n 种不同的纸币，第 i 种面值为整数 b_i，每种纸币能取的数量有 $c_1, c_2, ..., c_k$ 张，问有几种凑出 S 元的方法？答案模给定的模数 P

$0 < n \leq 10^2, 0 \leq S \leq 10^3, 0 < P \leq 10^9$

面值问题

问题：有 n 种不同的纸币，第 i 种面值为整数 b_i，每种纸币能取的数量有 $c_1, c_2, ..., c_k$ 张，问有几种凑出 S 元的方法？答案模给定的模数 P

$0 < n \leq 10^2, 0 \leq S \leq 10^3, 0 < P \leq 10^9$

- 显然我们可以使用动态规划，设 f[i][j] 为前 i 种纸币凑出 j 元的方案数
- 那么 f[i][j]=f[i-1][j-c_1 b_i]+f[i-1][j-c_2 b_i]+...+f[i-1][j-c_k b_i]
- 时间复杂度 O(nS^2), 还有其他做法吗?

面值问题

问题：有 n 种不同的纸币，第 i 种面值为整数 b_i，每种纸币能取的数量有 $c_1, c_2, ..., c_k$ 张，问有几种凑出 S 元的方法？答案模给定的模数 P

$0 < n \leq 10^2, 0 \leq S \leq 10^3, 0 < P \leq 10^9$

- 显然我们可以使用动态规划，设 f[i][j] 为前 i 种纸币凑出 j 元的方案数
- 那么 f[i][j]=f[i-1][j-$c_1 b_i$]+f[i-1][j-$c_2 b_i$]+...+f[i-1][j-$c_k b_i$]
- 时间复杂度 O(nS^2), 还有其他做法吗?
- 把第 i 种纸币看做一个多项式:$x^{c_1 b_i} + x^{c_2 b_i} + ... + x^{c_k b_i}$
- 将这 n 个多项式相乘，第 S 项系数就是凑出 S 元的方法
- 每次相乘后将高于 S 次的项去掉
- 时间复杂度 O(nS^2), 和动态规划的复杂度相同
- ~~使用 FFT 优化多项式乘法，复杂度降为 O(nSlogS)~~

模意义下的一元多项式

- 定理:$(a_n\%P)x^n + (a_{n-1}\%P)x^{n-1} + ... + (a_1\%P)x + (a_0\%P) \equiv a_nx^n + a_{n-1}x^{n-1} + ... + a_1x + a_0 (mod\ P)$
- 定理：$a_n(x\%P)^n + a_{n-1}(x\%P)^{n-1} + ... + a_1(x\%P) + a_0 \equiv a_nx^n + a_{n-1}x^{n-1} + ... + a_1x + a_0 (mod\ P)$
- 利用同余里的公式，这些定理都是显然的
- 如果 P 是质数，利用费马小定理，可以把次数大于等于 P-1 的项折回低次项，给定 x 计算一个多项式数值的复杂度是 O(min(P,n))
- 在模 P（质数）的意义下，如何计算多项式乘法?

模意义下的一元多项式

- 定理:$(a_n\%P)x^n+(a_{n-1}\%P)x^{n-1}+...+(a_1\%P)x+(a_0\%P)\equiv a_nx^n+a_{n-1}x^{n-1}+...+a_1x+a_0(mod\ P)$
- 定理：$a_n(x\%P)^n+a_{n-1}(x\%P)^{n-1}+...+a_1(x\%P)+a_0\equiv a_nx^n+a_{n-1}x^{n-1}+...+a_1x+a_0(mod\ P)$
- 利用同余里的公式，这些定理都是显然的
- 如果 P 是质数，利用费马小定理，可以把次数大于等于 P-1 的项折回低次项，给定 x 计算一个多项式数值的复杂度是 O(min(P,n))
- 在模 P（质数）的意义下，如何计算多项式乘法?
- 只要计算每个系数模 P 的值即可
- 如果次数达到 P-1，需要折回

Thanks for listening

Any Questions?