组合数学及相关计数法

一、计数原理

1.加法原理

举个例子：从甲地到乙地有海陆空三种选择，坐船有3班，坐车有5班，坐飞机有2班，问从甲地到乙地共几种选择

解：这就是个幼儿园的题$3+5+2=10$

加法原理(分类计数原理)：完成一件事情共有n类方法，第一类方法有$n1$种方案，第二类有$n2$种方案…那么完成这一件事共有$n1+n2+n3+...$种方法，注意分类需要不重不漏

2.乘法原理

依旧举个例子：从甲地到乙地需要途径A地，从甲地到A地有5种方法，从A地到乙地有3种方法，问从甲地到乙地共有几种选择

解：这个大概要小学水平了$5*3=15$

乘法原理(分步计数原理)：完成一件事需要分成n步进行，第一步有$n1$种方法，第二步有$n2$种方法…那么完成这件事共有$n1*n2*...$种方法，依旧不重不漏

3.排列与组合

1.排列：从n个不同的元素中取出$m(m\leq n)$个元素，按照一定的顺序排成一列，叫做从n个不同元素中取出m个元素的一个排列，用符号$A(n,m)$表示

$A(n,m)=n*(n-1)*(n-2)*...*(n-m+1)=n!/(n-m)!$

2.组合：从n个不同元素中取出$m(m\leq n)$个元素的所有组合的个数，叫做从n个不同元素中取出m个元素的组合数，用符号$C(n,m)$表示

$C(n,m)=A(n,m)/A(m,m)=n!/(m!(n-m)!)$

几种常用策略/方法

1）特殊元素和特殊位置优先策略

例：由$0,1,2,3,4,5$可组成多少个没有重复数字的五位奇数

解:由于末位和首位有特殊要求，应该优先安排，以免不合要求的元素占了这两个位置。

先排末位共有$C(3,1)$，然后排首位共有$C(4,1)$，最后排其它位置共有$A(4,3)$

由分步计数原理得$ans=C(3,1)C(4,1)A(4,3)$

2）相邻元素捆绑策略

例：7人站成一排 ,其中甲乙相邻，且丙丁相邻, 共有多少种不同的排法.

解:先将甲乙两元素捆绑成整体并看成一个复合元素，同时丙丁也看成一个复合元素，再与其它元素进行排列，同时对相邻元素内部进行自排。

由分步计数原理可得共$A(5,5)A(2,2)A(2,2)$种不同的排法。

例：记者要为5名志愿者和他们帮助的2位老人拍照，要求排成一排，2位老人相邻但不排在两端，求不同排法的数量。

解：第一步排两端，共有$C(5,1)*C(4,1)$(或$A(5,2)$)种排列方式

第二步将两个老人看作一个整体和剩余4个志愿者全排列，有$A(4,4)$种排列方式

第三步两个老人内部全排列，有$A(2,2)$种排列方式

所以共$A(5,2)*A(4,4)*A(2,2)$种排列方式

3）不相邻问题插空策略

例：一个晚会的节目有4个舞蹈,2个相声,3个独唱,舞蹈节目不能连续出场,则节目的出场顺序有多少种?

解:第一步，排列2个相声和3个独唱，共有$A(5,5)$种方案

第二步，将四种舞蹈插入第一步排好的5个元素中间包含首尾两个空位共有$A(6,4)$种不同的方法,节目的不同顺序共有$A(5,5)A(6,4)$种。

不相邻问题通常用插空法：

把要求不相邻的元素放在一边，先排其他元素，再将不相邻的元素插在已经排好的元素之间的空位上。

4）定序问题倍缩空位插入策略

例：7人排队，其中甲乙丙3人顺序一定，共有多少不同的排法

解：共有不同排法种数是$A(7,7)/A(3,3)$

(倍缩法)对于某几个元素顺序一定的排列问题，可先把这几个元素与其他元素一起进行排列，然后用总排列数除以这几个元素之间的全排列数

5）排列问题求幂策略

例：把6名实习生分配到7个车间实习，共有多少种不同的分法

解:完成此事共分六步:把第一名实习生分配到车间有7种分法，把第二名实习生分配到车间也有7种分法

依此类推，由乘法原理共有7^6种不同的排法。

6）环排问题线排策略

例：8人围桌而坐,共有多少种坐法?

解：围桌而坐与坐成一排的不同点在于，坐成圆形没有首尾之分，所以固定一人，并从此位置把圆形展成直线其余7人共有$(8-1)!=7!$种排法

一般地，n个不同元素作圆形排列，共有$(n-1)!$种排法

如果从n个不同元素中取出m个元素作圆形排列，共有$A(n,m)/m$种排法

7）多排问题直排策略

例：8人排成前后两排,每排4人,其中甲乙在前排,丙在后排,共有多少排法

解：8人排前后两排,相当于8人坐8把椅子，可以把椅子排成一排。

前排有2个特殊元素，方案数为$A(4,2)$

后4个位置上有一个特殊元素丙，方案数为$A(4,1)$

其余的5人在5个位置上任意排列，方案数为$A(5,5)$

共有$A(4,2)A(4,1)A(5,5)$种方案

8）排列组合混合问题先选后排策略

例：有5个不同的小球,装入4个不同的盒内，每盒至少装一个球，求共有多少不同的装法

解：第一步从5个球中选出2个组成复合元共有$C(5,2)$种方法.再把4个元素(包含一个复合元素)装入4个不同的盒内有$A(4,4)$种方法

根据分步计数原理装球的方法共有$C(5,2)A(4,4)$。

9）平均分组问题除法策略

例：6本不同的书平均分成3堆,每堆2本共有多少分法

解：分三步取书得$C(6,2)C(4,2)C(2,2)$种方法，但这里出现重复计数的现象，每种方案计算了$A(3,3)$次，故最终答案为$C(6,2)C(4,2)C(2,2)/A(3,3)$

10）重排列

例：由四面红旗，三面蓝旗，二面黄旗，五面绿旗排成的一排彩旗有多少种?

解：将14面彩旗排成一个排列，方案数$A(14,14)$，其中红旗之间每种排列等价，方案数$A(4,4)$，以此类推

共有$A(14,14)/(A(4,4)A(3,3)A(2,2)A(5,5))$种

常用结论

将一个长度为n的序列划分成m段非空子串的方案数为$C(n-1,m-1)$，相当于在$n-1$个空位中选择$m-1$个插入隔板的方案数

将一个长度为n个序列划分成m段可空子串的方案数为$C(m+n-1,m-1)$，相当于在$m+n-1$个位置中选择$m-1$个作为隔板的方案数.

[HNOI2012] 排队

题目描述

某中学有 n 名男同学，m 名女同学和两名老师要排队参加体检。他们排成一条直线，并且任意两名女同学不能相邻，两名老师也不能相邻，那么一共有多少种排法呢？（注意：任意两个人都是不同的）

输入格式

只有一行且为用空格隔开的两个非负整数 n 和 m，其含义如上所述。

输出格式

仅一个非负整数，表示不同的排法个数。注意答案可能很大。

提示

对于 30% 的数据 $n \leq 100$，$m \leq 100$。
对于 100% 的数据 $n \leq 2000$，$m \leq 2000$。

将男生和老师混在一起，将女生往里插

如果两个老师在一起 方案数为$A_{n+1}^{n+1}A_2^2$
两个老师之间必须插进一个女生
将两个老师和一个女生打包成一坨当成一个男生
方案数为$mA_{n+1}A_2$
剩余m-1个女生，插进n+2个空位中
总方案数$mA_{n+1}A_2A_{m-1}C_{n+2}^{m-1}$

如果两个老师不在一起 方案数为$A_{n+2}-A_{n+1}A_2$
此时女生随便插 方案数为$(A_{n+2}-A_{n+1}A_2)A_mC_{n+3}^m$

最终答案为$mA_{n+1}A_2A_{m-1}C_{n+2}^{m-1}+(A_{n+2}-A_{n+1}A_2)A_mC_{n+3}^m$

4.组合数取模

题1：求组合数
有$q(q <= 10000)$组询问，每组询问两个整数$n, m(1 <= m <= n <= 2000)$,求$C_n^m \bmod (10^9+7)$
递推法
递推式：$C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$
从n个不同得数中选出m个不得方案数是C_n^m，
对第1个数有选和不选两种决策：
若不选，则从剩下得$n-1$中选m个，即C_{n-1}^m.
若选，则从剩下得$n-1$中选$m-1$个，即C_{n-1}^{m-1}.

```
#include <iostream>//时间复杂度n^2
using namespace std;

const int N=2010, P=1e9+7;
int C[N][N];

void init(){
  for(int i=0; i<N; i++) C[i][0] = 1;
  for(int i=1; i<N; i++)
    for(int j=1; j<=i; j++)
      C[i][j]=(C[i-1][j]+C[i-1][j-1])%P;
}
int main(){
  init();
  int T, n, m;
  cin >> T;
  while(T--){
    cin >> n >> m;
    printf("%d\n", C[n][m]);
  }
  return 0;
}
```

i\j	0	1	2	3	4	5	6
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	5	10	10	5	1	
6	1	6	15	20	15	6	1

杨辉三角公式

1. $C_n^0 = C_n^n = 1$
2. $C_n^m = C_n^{n-m}$
3. $C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$

题2：求组合数

有$q(q <= 10000)$组询问，每组询问两个整数$n, m(1 <= m <= n <= 10^5)$,求$C_n^m \bmod (10^9+7)$

递推法：$C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$，会TLE。

考虑用$C_n^m = \frac{n!}{(n-m)!m!}$直接计算。

开两个数组分别存模意义下得阶乘和阶乘的逆元。

用$f[x]$存$x!(mod\,p)的值$

用$g[x]$存$(x!)^{-1}(mod\,p)的值$

因为p是质数并且n, m都小于p，即n, m于p互质(因为数据范围给了$n和m$最大值都比10^9+7小，$10^9+7又是个大质数$,所以互质)，所以根据费马小定理$a * a^{p-2} \equiv 1(mod\,p)$,

可以用快速幂求逆元：

$C_n^m(mod\,p) = f[n] * g[n-m] * g[m](mod\,p)$

```
#include <iostream>
using namespace std;

typedef long long LL;
const int N=100010, P=1e9+7;
LL f[N], g[N];

LL qpow(LL a, int b){
  LL res = 1;
  while(b){
    if(b & 1) res=res*a%P;
    a = a*a%P;
    b >>= 1;
  }
  return res;
}
void init(){
  f[0] = g[0] = 1;
  for(int i=1; i<N; i++){
    f[i] = f[i-1]*i%P;
    g[i] = g[i-1]*qpow(i,P-2)%P;
  }
}
LL getC(LL n, LL m){
  return f[n]*g[m]%P*g[n-m]%P;
}
int main(){
  init();
  int q, n, m;
  cin >> q;
  while(q--){
    cin >> n >> m;
    printf("%lld\n", getC(n,m));
  }
  return 0;
}
```

时间复杂度$O(nlogp)$
求逆元也可以递推：阶乘的逆元
$\frac{1}{i!}(mod\ p)=\frac{1}{i}\times\frac{1}{(i-1)!}(mod\ p)=qpow(i,p-2)\times g[i-1](mod\ p)$

```
// 阶乘逆元
typedef long long LL;
const int N=1e6+5;
LL fac[N];        //阶乘
LL inv[N];        //逆元

// 求阶乘
void get_fac()
{
    fac[0] = inv[0] = 1;
    for (int i = 1 ; i <= 1000000 ; i++)
    {
        fac[i] = fac[i-1] * i % m;
        inv[i] = quick_power(fac[i],m-2,m);
        //表示i的阶乘的逆元 ， 费马小定理：fac[i]的逆元就是fac[i]的m-2次幂
    }
}

// 求组合数
inline LL get_C( LL a, LL b ) //   C(a,b) = a!/((a-b)!*b!) % mod
{
    return fac[a] * inv[a-b] % m * inv[b] % m;
}
```

大组合数取模问题
给定整数n, m, p的值，求$C_n^m(\mathrm{mod}\ p)$的值。其中$1<=m<=n<=10^{18}$，$1<=p<=10^5$，保证p为质数。

$Lucas$定理用于求解大组合数取模的问题，**其中模数必须为素数。**

4.1 用途

正常的组合数 $C(n,m)$ 运算，当 n 比较小的时候，可以通过递推公式求解。
当 n 很大，模数 p 为质数，且 $n<p$ 时，此时可以利用乘法逆元来解决问题。

但是当模数是一个不大的质数的时候，即 $n>p$时，公式中某个分母上某个数可能不存在逆元（比如分母上某个数是 p 的倍数），此时就不能简单地通过递推求解来得到答案，需要用到 Lucas 定理。

4.2 求解方式

对于质数 p，有

$$C(n,m) \equiv C(n/p, m/p) \times C(n \bmod p, m \bmod p) \pmod p$$

上式中，$n \bmod p$ 和 $m \bmod p$ 一定小于 p，可以直接求解，$C(n/p,m/p)$ 可以继续用 Lucas 定理求解。这也就要求 p 的范围不能够太大，一般在 10^5 左右。边界条件：当 $m=0$ 的时候，返回 1。

4.3 参考代码

```
LL C(int n, int m, int p){
  return f[n]*g[m]*g[n-m]%p;//其中f[n]是n的阶乘，g[m]是m! 的逆元
}


ll Lucas(ll n, ll m, ll p) {
  if (m == 0) return 1;
  return (C(n % p, m % p, p) * Lucas(n / p, m / p, p)) % p;
}
```

4.4 Lucas 定理证明

引理1： $C_p^x \equiv 0(mod\, p).0<x<p$

因为$C_p^x = \frac{p!}{x!(p-x)!} = \frac{p(p-1)!}{x(x-1)!(p-x)!} = \frac{p}{x}C_{p-1}^{x-1}$

所以$C_p^x \equiv p*inv(x)*C_{p-1}^{x-1} \equiv 0(mod\, p).$

其中$x<p$,所以x和p是互质的，所以$inv(x)$是存在的。

引理2： $(1+x)^p \equiv 1+x^p(mod\, p)$

由二项式定理$(1+x)^p = \sum_{i=0}^{p} C_p^i x^i$

由引理1可知，只剩$i=0,p$两项，得证。

Lucas 定理证明：
令$n=ap+b, m=cp+d$
$(1+x)^n \equiv \sum_{i=0}^{n} C_n^i x^i (mod\, p) ----------(1)$

$(1+x)^n \equiv (1+x)^{ap+b}$

$\equiv ((1+x)^p)^a*(1+x)^b$
$\equiv (1+x^p)^a*(1+x)^b$ //根据引理2
$\equiv \sum_{i=0}^{a} C_a^i x^{ip} * \sum_{j=0}^{b} C_b^j x^j (mod\, p) --(2)$
(1)式中x^m的系数为C_n^m
(2)式中$x^m=x^{cp+d}$$=x^{cp}*x^d$的系数为$C_a^c C_b^d$
根据对应项系数相等：
所以$C_n^m \equiv C_a^c C_b^d (mod\, p)$

即$C_n^m \equiv C_{n/p}^{m/p} C_{n\ mod\ p}^{m\ mod\ p} (mod\, p)$,证毕。

时间复杂度为 $O(plogp+log_p n)$。
P3807 【模板】卢卡斯定理/Lucas 定理模板：

```cpp
#include <iostream>
using namespace std;

typedef long long LL;
const int N = 100010;
LL f[N], g[N];

LL qpow(LL a, int b, int p){
  LL res = 1;
  while(b){
    if(b & 1) res=res*a%p;
    a = a*a%p;
    b >>= 1;
  }
  return res;
}
void init(int p){
  f[0] = g[0] = 1;
  for(int i=1; i<=p; i++){
    f[i] = f[i-1]*i%p;
    g[i] = g[i-1]*qpow(i,p-2,p)%p;
  }
}
LL getC(int n, int m, int p){
  return f[n]*g[m]*g[n-m]%p;
}
int lucas(LL n, LL m, int p){
  if(m==0) return 1;
  return lucas(n/p,m/p,p)*getC(n%p,m%p,p)%p;
}
int main(){
  int q, n, m, p;
  cin >> q;
  while(q--){
    cin >> n >> m >> p;
    init(p);
    printf("%d\n",lucas(n+m,n,p));
  }
  return 0;
}
```

5 中国剩余定理（CRT）

5.1 算法简介

「物不知数」问题

> 有物不知其数，三三数之剩二，五五数之剩三，七七数之剩二。问物几何？

中国剩余定理 (CRT) 可求解如下形式的一元线性同余方程组。

$$\begin{cases} x \equiv r_1 \pmod{m_1} \\ x \equiv r_2 \pmod{m_2} \\ \vdots \\ x \equiv r_n \pmod{m_n} \end{cases}$$

（其中模数 $m_1, m_2, \cdots, m_n$ 为两两互质整数）
求x的最小非负整数解。（两两互质，不代表两个数都是质数）

5.2 中国剩余定理过程：

1. 计算所有模数的乘积 $M = m_1 * m_2 * m_3 ..... * m_n$;
2. 对于第 i 个方程：
   a. 计算 $c_i = \frac{M}{m_i}$
   b. 计算 c_i在模m_i意义下的逆元c_i^{-1}。(逆元一定存在，因为c_i中已经把m_i除掉了)
3. 方程组的唯一解为：$x = \sum_{i=1}^{n} r_i c_i c_i^{-1} \bmod M$。

例:

$$\begin{cases} x \equiv 2 \pmod 3 \\ x \equiv 3 \pmod 4 \\ x \equiv 1 \pmod 5 \end{cases}$$

$1.M = 3 \times 4 \times 5$
$2.c_1 = 20, c_1^{-1} = 2,$ 因为$20x \equiv 1(mod\ 3)$
$c_2 = 15, c_2^{-1} = 3,$ 因为$15x \equiv 1(mod\ 4)$
$c_3 = 12, c_3^{-1} = 3,$ 因为$12x \equiv 1(mod\ 5)$
$3.x = (2 \times 20 \times 2 + 3 \times 15 \times 3 + 1 \times 12 \times 3)\%60 = 251\%60 = 11$

5.3 算法证明

证明:
先证明$\sum_{i=1}^{n} r_i c_i c_i^{-1}$（先不对$M$取模）满足每个$x \equiv r_i(mod\ m_i)$

分情况:(i和j，j是下面“故”那的枚举变量)
当$i \neq j$时，$c_j \equiv 0(mod\ m_i)$,则$c_j c_j^{-1} \equiv 0(mod\ m_i)$（因为$c_j$中去掉了$m_j$但是还存在$m_i$）

当$i\ = j$时，$c_i \not\equiv 0(mod\ m_i)$,则$c_i c_i^{-1} \equiv 1(mod\ m_i)$ (因为c_i在模m_i意义下的逆元c_i^{-1})
故 $x \equiv \sum_{j=1}^{n} r_j c_j c_j^{-1}(\mathrm{mod} m_i)$。

$\equiv r_i c_i c_i^{-1}(\mathrm{mod} m_i)$（因为当$i \neq j$时，都等于0，所以只剩下$i$那一项了）

$\equiv r_i(\mathrm{mod} m_i)$

而$\sum_{i=1}^{n} r_i c_i c_i^{-1}(\mathrm{mod} M)$对$m_i$来说，只是减去了$m_i$的若干倍，不影响余数$r_i$.证毕。
（因为M本身就是m_i的整数倍，所以模M,其实就相当于减去了m_i的若干倍）

5.4 Luogu P1495 【模板】中国剩余定理（CRT）/ 曹冲养猪

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

typedef long long LL;
LL n, a[11], b[11];

LL exgcd(LL a,LL b,LL &x,LL &y){
  if(b==0){x=1, y=0; return a;}
  LL d, x1, y1;
  d = exgcd(b, a%b, x1, y1);
  x = y1, y = x1-a/b*y1;
  return d;//x就是逆元
}
LL CRT(LL m[], LL r[]){
  LL M = 1, ans = 0;
  for(int i=1;i<=n;i++) M*=m[i];
  for(int i=1; i<=n; i++){//时间复杂度由n和扩欧来决定的
    LL c = M/m[i], x, y;
    exgcd(c, m[i], x, y);
    ans = (ans+r[i]*c*x%M)%M;
  }
  return (ans%M + M)%M;
}
int main(){
  scanf("%lld", &n);
  for(int i = 1; i <= n; ++i)
    scanf("%lld%lld", a+i, b+i);
  printf("%lld\n", CRT(a,b));
  return 0;
}
时间复杂度$O(nlogc)$.
```

6.扩展中国剩余定理

问题

求解线性同余方程组

$$\begin{cases} x \equiv r_1 \ (mod\ m_1) \\ x \equiv r_2 \ (mod\ m_2) \\ \quad \cdots \\ x \equiv r_n \ (mod\ m_n) \end{cases}$$

其中 $m_1, m_2, \cdots, m_n$ 为**不一定两两互质**的整数，
求 x 的最小非负整数解。

中国剩余定理（CRT）已不可行

其构造解 $x = \sum_{i=1}^{n} r_i c_i c_i^{-1} \ (mod\ M)$

其中 $c_i x \equiv 1 \ (mod\ m_i)$，即$c_i x + m_i y = 1 = \gcd(c_i, m_i)$

根据裴蜀定理，c_i, m_i 应该互质，$c_i = \frac{m_1 \times \cdots \times m_n}{m_i}$

如果 c_i, m_i 不互质，则 c_i^{-1} 不存在，算法失效

扩展中国剩余定理（EXCRT）

前两个方程：$x \equiv r_1 \ (mod\ m_1)$，$x \equiv r_2 \ (mod\ m_2)$

转化为不定方程：$x = m_1 p + r_1 = m_2 q + r_2$

则 $m_1 p - m_2 q = r_2 - r_1$

由裴蜀定理，

当 $\gcd(m_1, m_2) \nmid (r_2 - r_1)$ 时，无解

当 $\gcd(m_1, m_2) \mid (r_2 - r_1)$ 时，有解

由扩欧算法，

得特解 $p = p * \frac{r_2 - r_1}{gcd}$，$q = q * \frac{r_2 - r_1}{gcd}$

其通解 $P = p + \frac{m_2}{gcd} * k$，$Q = q - \frac{m_1}{gcd} * k$

所以 $x = m_1 P + r_1 = \frac{m_1 m_2}{gcd} * k + m_1 p + r_1$

前两个方程**等价合并**为一个方程 $x \equiv r \ (mod\ m)$

其中 $r = m_1 p + r_1$，$m = lcm(m_1, m_2)$

所以 n 个同余方程只要合并 $n-1$ 次，即可求解

Luogu P4777 【模板】扩展中国剩余定理（EXCRT）

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

typedef __int128 LL;
const int N = 100005;
LL n, m[N], r[N];

LL exgcd(LL a,LL b,LL &x,LL &y){
  if(b==0){x=1, y=0; return a;}
  LL d, x1, y1;
  d = exgcd(b, a%b, x1, y1);
  x = y1, y = x1-a/b*y1;
  return d;
}
LL EXCRT(LL m[], LL r[]){
  LL m1, m2, r1, r2, p, q;
  m1 = m[1], r1 = r[1];
  for(int i=2; i<=n; i++){
    m2 = m[i], r2 = r[i];
    LL d = exgcd(m1,m2,p,q);
    if((r2-r1)%d){return -1;}
    p=p*(r2-r1)/d; //特解
    p=(p%(m2/d)+m2/d)%(m2/d);//因为特解p有可能小于0，所以通过模+模的形式变成最小正整数
    r1 = m1*p+r1;
    m1 = m1*m2/d;
  }
  return (r1%m1+m1)%m1;
}
int main(){
  scanf("%lld", &n);
  for(int i = 1; i <= n; ++i)
    scanf("%lld%lld", m+i, r+i);
  printf("%lld\n",EXCRT(m,r));
  return 0;
}
```

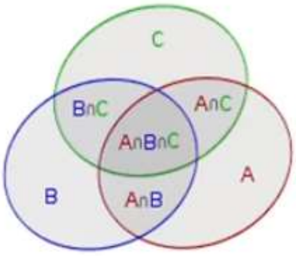
7.容斥原理

集合的并

问题：一个班里参加数理化竞赛的人数统计如表，那么班里参加竞赛的共有多少人？

竞赛学科	数学	物理	化学	数学物理	物理化学	数学化学	数理化
人数	5	6	5	3	2	2	1

把参加数学、物理、化学竞赛的学生集合分别用 A, B, C 表示，则学生总数等于 $|A \cup B \cup C|$。
如果把这三个集合的元素个数 $|A|, |B|, |C|$ 直接加起来，会有一些元素重复统计了，因此需要减去 $|A \cap B|, |B \cap C|, |C \cap A|$，但又有一小部分多减了，需要把 $|A \cap B \cap C|$ 加回来。



即 $|A \cup B \cup C|$

$$
\begin{aligned}
&= |A| + |B| + |C| \\
&\quad - |A \cap B| - |B \cap C| - |C \cap A| \\
&\quad + |A \cap B \cap C| \\
&= 5 + 6 + 5 - 3 - 2 - 2 + 1 \\
&= 10
\end{aligned}
$$

$A_1, A_2, \cdots, A_n$ 是 n 个集合；则这 n 个集合并集的元素个数是：

$$
|\bigcup_{i=1}^{n} A_i| = \sum_{i=1}^{n} |A_i| - \sum_{i<j} |A_i \cap A_j| + \sum_{i<j<k} |A_i \cap A_j \cap A_k| - \cdots + (-1)^{n-1} |A_1 \cap A_2 \cap \cdots \cap An|
$$

解析：

n 个集合进行并运算后，元素个数，通常使用 容斥原理 进行计算；

$\sum_{i=1}^{n} |A_i|$：将每个集合中的 元素个数 相加，该值大于 总元素数，需要进行修正；（系数值 $(-1)^0$）

$\sum_{i<j} |A_i \cap A_j|$：减去两两相交的元素个数，该值又小于 总元素数，继续进行修正；（系数值 $(-1)^1$）

$\sum_{i<j<k} |A_i \cap A_j \cap A_k|$：加上三个集合相交的元素个数，该值大于 总元素数，继续进行修正；（系数值 $(-1)^2$）

减去四个集合相交的元素个数，该值小于 总元素数，继续进行修正；（系数值 $(-1)^3$）

$\vdots$

$(-1)^{n-1} |A_1 \cap A_2 \cap \cdots \cap An|$：加上 $(-1)^{n-1}$ 乘以 $n-1$ 个集合相交的元素个数；（系数值 $(-1)^{n-1}$）

上述 奇数个集合 交集元素个数 前系数是 正数，偶数个集合 交集元素个数 前系数是 负数；

例题：
给定一个整数n 和m个不同的质数$p_1, p_2, \ldots, p_m$。
请你求出$1 \sim n$中能被$p_1, p_2, \ldots, p_m$中的至少一个数整除的整数有多少个。

输入格式
第一行包含整数n 和 m。
第二行包含m个质数。

输出格式
输出一个整数，表示满足条件的整数的个数。

数据范围
$1 \le m \le 16,$
$1 \le n, pi \le 10^9$
输入样例:
10 2
2 3
输出样例:
7

例 $n = 10, p_1 = 2, p_2 = 3, p_3 = 5,$
求 1~10 中能整除 2 或 3 或 5 的数的个数。
即 {2,3,4,5,6,8,9,10}，共 8 个。
$S_1 = \{2,4,6,8,10\},\ S_2 = \{3,6,9\},\ S_3 = \{5,10\},$
$S_1 \cap S_2 = \{6\}, S_1 \cap S_3 = \{10\}, S_2 \cap S_3 = S_1 \cap S_2 \cap S_3 = \{\}$

容斥原理，考察能被$p_i(1 \le i \le m)$，能被p_i整除的数的数量为$\lfloor \frac{n}{p_i} \rfloor$，结果加上这些数量，但是会有重复，比如同时能被两个质数$p_i, p_j(1 \le i, j \le m)$整除的数据被计算了两次，需要减去，…，因此最终的答案：

$\sum_i \lfloor \frac{n}{p_i} \rfloor - \sum_{i,j} \lfloor \frac{n}{p_i \times p_j} \rfloor + \sum_{i,j,k} \lfloor \frac{n}{p_i \times p_j \times p_k} \rfloor - ......$

时间复杂度：一共有$2^m - 1$项相加减，每次计算主要是质数相乘，最多相乘$m - 1$次，因此时间复杂度为$O(2^m \times m)$。

1. **交集的大小**等于 n 除以质数的乘积。
即$|S_1| = \frac{n}{p_1}$，$|S_1 \cap S_2| = \frac{n}{p_1 * p_2}$，$|S_1 \cap S_2 \cap S_3| = \frac{n}{p_1 * p_2 * p_3}$

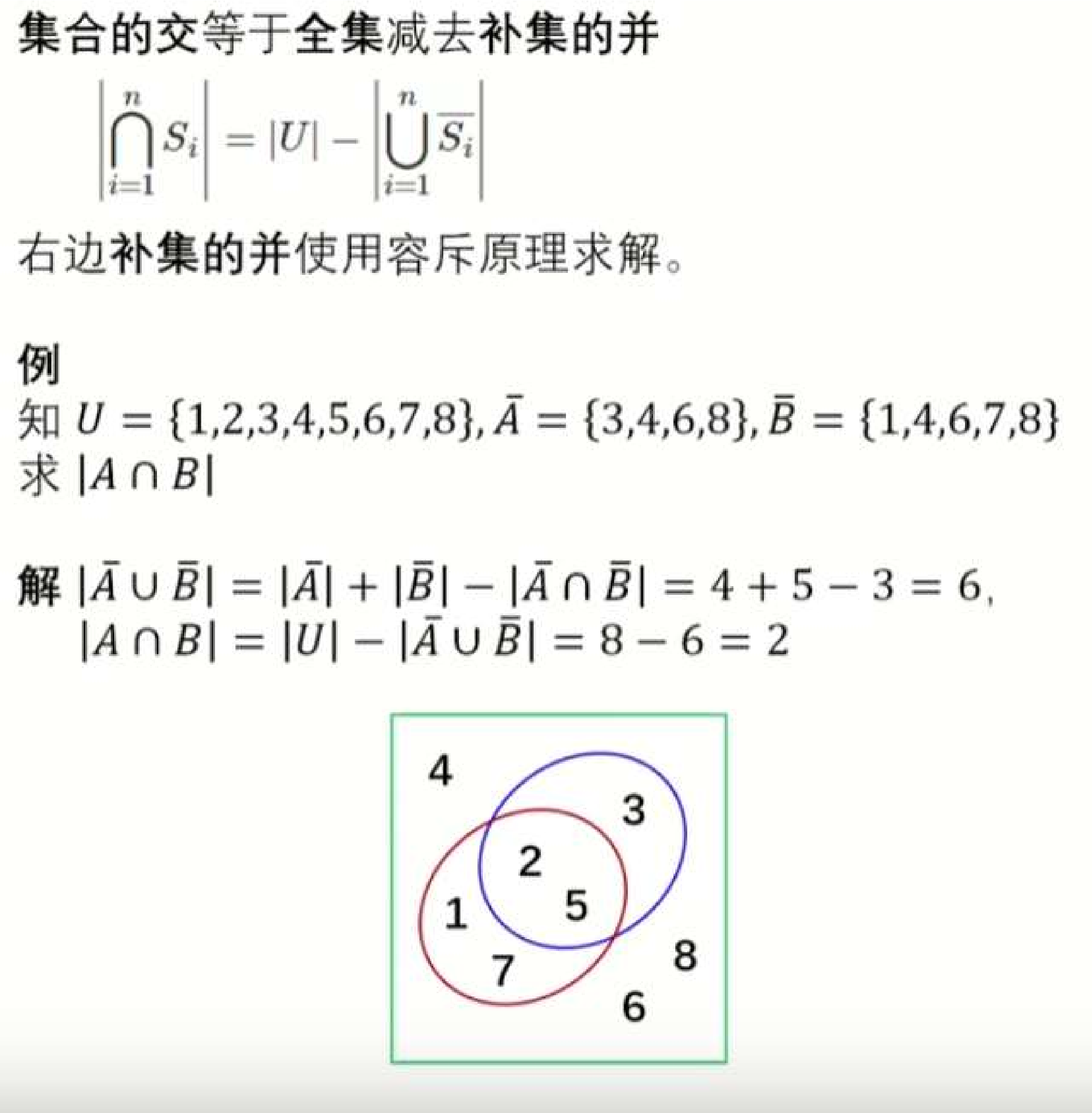
2. 使用的**二进制位**来表示每个集合**选与不选的状态**。
若有3个质数，就需要3个二进制位来表示所有状态。
$001 \to S_1$，$010 \to S_2$，$100 \to S_3$，
$011 \to S_1 \cap S_2$，$101 \to S_1 \cap S_3$，$011 \to S_2 \cap S_3$，
$111 \to S_1 \cap S_2 \cap S_3$
我们只要枚举从 001 到 111 的每个状态，
就可以计算出全部**交集的交错和**。

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

typedef long long LL;
const int N = 20;
int n, m, prim[N];

int calc(){ //容斥原理
  int res = 0;
  for(int i=1; i<1<<m; i++)
  {//枚举状态
    int t = 1, sign = -1;
    for(int j=0; j<m; j++) //过滤状态
      if(i & 1<<j)
      {
        if((LL)t*prim[j] > n)//如果质数得乘积大于n了
        {
          t = 0; break;
        }
        t *= prim[j]; //质数的积
        sign = -sign;
      }
    if(t) res += n/t*sign; //交集的和
  }
  return res;
}
int main(){
  cin >> n >> m;
  for(int i=0; i<m; i++) cin>>prim[i];
  cout << calc();
  return 0;
}
```

集合的交



[HAOI2008] 硬币购物

题目描述

共有 4 种硬币。面值分别为 c_1, c_2, c_3, c_4。

某人去商店买东西，去了 n 次，对于每次购买，他带了 d_i 枚 i 种硬币，想购买 s 的价值的东西。请问每次有多少种付款方法。

输入格式

输入的第一行是五个整数，分别代表 c_1, c_2, c_3, c_4, n。

接下来 n 行，每行有五个整数，描述一次购买，分别代表 d_1, d_2, d_3, d_4, s。

输出格式

对于每次购买，输出一行一个整数代表答案。

数据规模与约定

- 对于 100% 的数据，保证 $1 \le c_i, d_i, s \le 10^5$，$1 \le n \le 1000$。

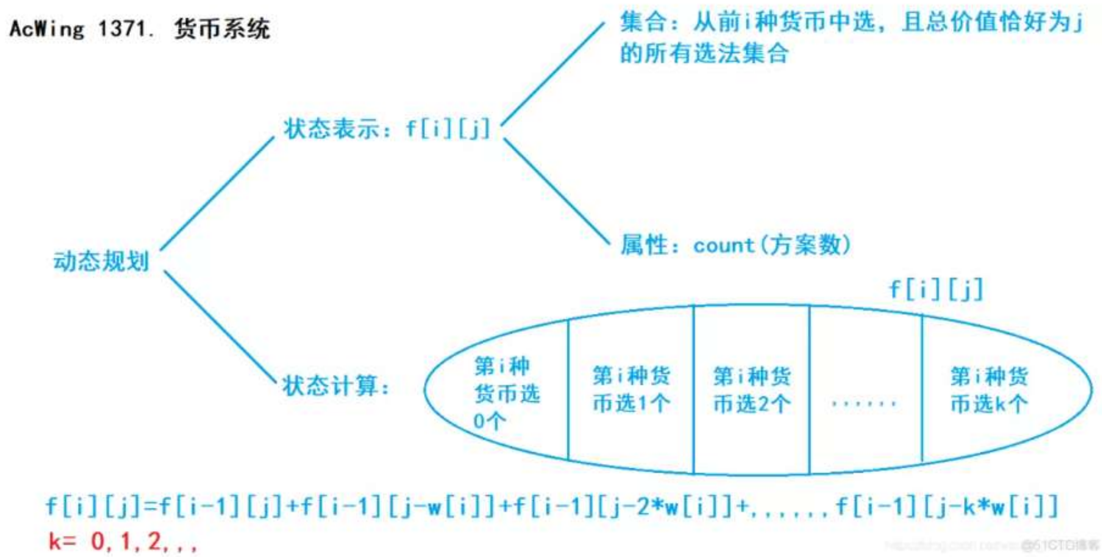
如果用优化的**多重背包**做，时间是 $O(4sn)$，会TLE。

等价于求方程 $\sum_{i=1}^{4} c_i x_i = s, x_i \le d_i$ 的非负整数解的个数。
直接求满足 $x_i \le d_i$ 条件下的交集困难，考虑补集思想。
求不受限制下的**全集**减去 $x_i \ge d_i + 1$ 条件下的**补集的并**。

先跑**完全背包**，预处理出硬币数量无限制的方案数 $f[i]$，
全集的大小就是 $f[s]$。时间是 $O(4s)$。
再考虑补集，如果只有第 j 种硬币超额使用，其方案数为 $f[s - c_j(d_j + 1)]$，应减去。如果 4 种硬币均超额使用，应减去 $\sum_{j=1}^{4} f[s - c_j(d_j + 1)]$。这样，两种都超额的方案都被减了两次，所以都加一次回来。三种都超额的方案又被多加了，所以要减去……超额使用的方案就是**补集的并**。

时间是 $O(4 * 2^4 * n)$，总复杂度 $O(4s + 4 * 2^4 * n)$

完全背包求方案数:



状态表示： $f[i][j]$表示从前i种货币中选，且总价值恰好为j的所有选法集合的方案数。

那么$f[n][m]$就表示表示 从前n种货币中选，且总价值恰好为m的所有选法集合的方案数，即为答案。

集合划分：

按照第i种货币可以选 0个,1个，2个，3个，，，，k个划分集合 $f[i][j]$。其中$k * w[i] <= j$,也就是说在背包能装下的情况下，枚举第i种货币可以选择几个。

状态计算：

$f[i][j] = f[i-1][j] + f[i-1][j - w[i]] + f[i-1][j - 2 * w[i]],,,,,, + f[i-1][j - k * w[i]]$

```
#include<iostream>
#include<cstdio>

using namespace std;
const int N = 30,M = 1e4 +10;
long long  f[N][M]; // 方案数很大使用long long 来存
int w[N];
int main()
{
    int n,m;
    cin>>n>>m;
    for(int i = 1; i <= n; i++) cin>>w[i];

    f[0][0] = 1; // 使用0种货币，凑0元钱，也是一种方案

    for(int i = 1; i <= n; i++)
      for(int j = 0; j <= m; j++)
      {
          for(int k = 0; k*w[i] <= j; k++)
            f[i][j] += f[i-1][j-k*w[i]];
      }
    cout<<f[n][m]<<endl;
    return 0;
}
```

一维DP:

考虑优化

v代表第i件物品的体积

$f[i][j] = f[i-1][j] + f[i-1][j-v] + f[i-1][j-2v]+,,,,,+f[i-1][j-kv])$

$f[i][j-v] = f[i-1,[j-v] + f[i-1][j-2v]+,,,,,+f[i-1][j-kv])$

因此:

$f[i][j] = f[i-1][j] + f[i][j-v])$

图示:

v代表第i件物品的体积

$$f[i][j] = f[i-1][j]+f[i-1][j-v]+f[i-1][j-2v]+,....,+f[i-1][j-kv])$$

$$f[i][j-v]= \underbrace{f[i-1,[j-v]+f[i-1][j-2v]+,....,+f[i-1][j-kv])}_{f[i][j-v]}$$

因此:

$$f[i][j] = f[i-1][j]+f[i][j-v])$$

去掉一维:

状态计算为: `f[j] = f[j] + f[j-v]`

```
#include<iostream>
#include<cstdio>
using namespace std;
const int N = 1e4 + 10;
long long f[N];
int main()
{
    int m,n;
    cin>>n>>m;
    f[0] = 1;  //初始化 f[0][0] = 1
    for(int i = 1; i <= n; i++)
    {
        int v;
        cin>>v;
        for(int j = v; j <= m; j++)
          f[j] += f[j-v];   // 状态计算方程
    }
    cout<<f[m]<<endl;
    return 0;
}
```

[HAOI2008] 硬币购物:

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

typedef long long LL;
int c[4],d[4],n,s;
LL f[100005];

void pack_pre(){ //完全背包预处理
    f[0] = 1;
    for(int i=0; i<4; i++)
      for(int j=c[i]; j<100005; j++)
        f[j] += f[j-c[i]];
}
LL calc(LL s){ //容斥原理
  LL res = 0;
  for(int i=1; i<1<<4; i++){//枚举状态
    LL t = 0, sign = -1;
    for(int j=0; j<4; j++) //过滤状态
      if(i & 1<<j){
        t += c[j]*(d[j]+1);
        sign = -sign;
      }
    if(s>=t) res += f[s-t]*sign;
  }
  return f[s]-res;
}
int main(){
    for(int i=0; i<4; i++) scanf("%d",&c[i]);
    pack_pre();
    scanf("%d", &n);
    while(n--){
        for(int i=0; i<4; i++) scanf("%d",&d[i]);
        scanf("%d",&s);
        printf("%lld\n", calc(s));
    }
    return 0;
}
```

8.卡特兰数

卡特兰数是组合数学中一个常出现于各种计数问题中的数列。其前几项为（从第零项开始）:$1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, \ldots$
卡特兰数是组合数学中一个常出现于各种计数问题中的数列。其前几项为（从第零项开始）:$1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, \ldots$

卡特兰数两个递推式，两个通项公式:
卡特兰数C_n满足以下递推关系:
$C_{n+1} = C_0C_n + C_1C_{n-1} + \cdots + C_nC_0$

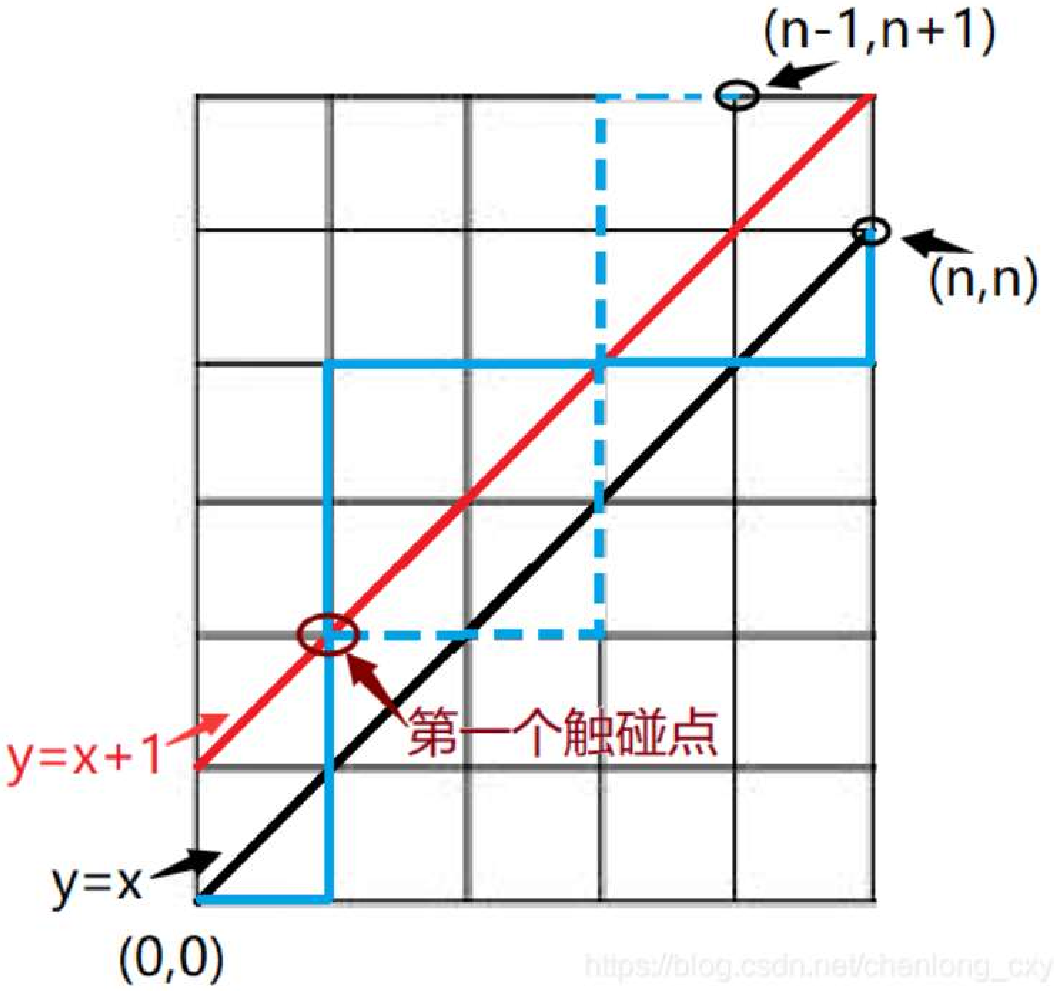
设$h(n)$为$catalan$数的第$n+1$项，令$h(0)=1, h(1)=1$，$catalan$数满足递推式:
$h(n) = h(0)*h(n-1) + h(1)*h(n-2) + \cdots + h(n-1)*h(0)$，$(n >= 2)$
例如:
$h(2) = h(0)*h(1) + h(1)*h(0) = 1*1 + 1*1 = 2$
$h(3) = h(0)*h(2) + h(1)*h(1) + h(2)*h(0) = 1*2 + 1*1 + 2*1 = 5$

卡特兰数 (Catalan)
以走网格为例，从格点 $(0,0)$ 走到格点 (n,n)，只能**向右或向上走**，
并且不能越过对角线的路径的条数，就是**卡特兰数**， 记为 H_n。

通项公式
(1) $H_n = C_{2n}^{n} - C_{2n}^{n-1}$　　(2) $H_n = \frac{1}{n+1} C_{2n}^{n}$　　(3) $H_n = \frac{4n-2}{n+1} H_{n-1}$

证明(1)式
先求**路径总数**，在 $2n$ 次移动中选 n 次向右移动，即 C_{2n}^{n}。
再求**非法路径**，即越过对角线的路径。
把 $y = x + 1$ 这条线画出来，碰到即说明是一条非法路径。
所有的非法路径与这条线有至少一个交点，把第一个交点设为 $(a, a+1)$，
把 $(a, a+1)$ 之后的路径全部按照 $y = x + 1$ 这条线对称过去，这样，最后的终点就会变成 $(n-1, n+1)$。
所有非法路径对称后都唯一对应着一条到 $(n-1, n+1)$ 的路径，
所以非法路径数就是 C_{2n}^{n-1}，合法路径数就是 $C_{2n}^{n} - C_{2n}^{n-1}$。
证明(2)式

$$
\begin{aligned}
H_n &= C_{2n}^{n} - C_{2n}^{n-1} = \frac{(2n)!}{n!\,n!} - \frac{(2n)!}{(n+1)!\,(n-1)!} \\
&= \frac{(2n)!}{n!(n-1)!}\left(\frac{1}{n} - \frac{1}{n+1}\right) = \frac{(2n)!}{n!n!(n+1)} = \frac{1}{n+1} C_{2n}^{n}
\end{aligned}
$$

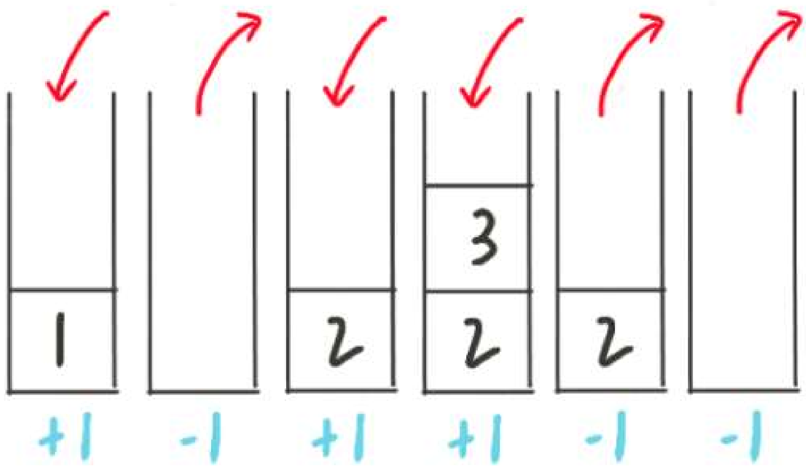


卡特兰数特征：

从$(0,0)$到(n,n)，不能越过对角线，即任意时刻，向上走的步数不能超过向右走的步数，一旦超过，就是非法的。
一种操作数不能超过另外一种操作数，或者两种操作不能有交集，这些操作的合法方案数，通常是卡特兰数。

卡特兰数的应用

1.n个元素进栈序列为：1，2，3，4，…，n，则有多少种出栈序列。
我们将进栈表示为$+1$，出栈表示为-1，则 1 3 2 的出栈序列可以表示为：+1 -1 +1 +1 -1 -1。



根据栈本身的特点，每次出栈的时候，必定之前有元素入栈，即对于每个 -1 前面都有一个 +1 相对应。因此，出栈序列的 所有前缀和 必然大于等于 0，并且 +1 的数量 等于 -1 的数量。

接下来让我们观察一下 n = 3 的一种出栈序列：+1 -1 -1 +1 -1 +1。序列前三项和小于 0，显然这是个非法的序列。

如果将 第一个 前缀和小于 0 的前缀，即前三项元素都进行取反，就会得到：-1 +1 +1 +1 -1 +1。此时有 3 + 1 个 +1 以及 3 - 1 个 -1。

因为这个小于 0 的前缀和必然是 -1，且 -1 比 +1 多一个，取反后，-1 比 +1 少一个，则 +1 变为 n + 1 个，且 -1 变为 n - 1 个。进一步推广，对于 n 元素的每种非法出栈序列，都会对应一个含有 n + 1 个 +1 以及 n - 1个 -1 的序列。

如何证明这两种序列是一一对应的?

假设非法序列为 A，对应的序列为 B。每个 A 只有一个"第一个前缀和小于 0 的前缀"，所以每个 A 只能产生一个 B。而每个 B 想要还原到 A，就需要找到"第一个前缀和大于 0 的前缀"，显然 B 也只能产生一个 A。

每个 B 都有 n + 1 个 +1 以及 n - 1 个 -1，因此 B 的数量为 C_{2n}^{n+1} ，相当于在长度为 2n 的序列中找到 `n + 1` 个位置存放 +1。相应的，非法序列的数量也就等于 C_{2n}^{n+1} 。

出栈序列的总数量共有 C_{2n}^{n} ，因此，合法的出栈序列的数量为 $C_{2n}^{n}-C_{2n}^{n+1}=\frac{C_{2n}^{n}}{n+1}$ 。

问题分析:
如果将入栈设为用0代表，出栈用1代表，因为每个元素都要出栈，进栈，所以就有n个0，和n个1,而一个数只有先进栈才能出栈，所以这个问题就变成了，求n个0和n个1，通过一系列的组合，组成一个长度为$2n$的序列，满足任意前缀中0的个数都不少于1的个数的序列数量为。请注意这个任意前缀就限定了在前缀中0肯定在1的前面。例如如果$n=2$，排列顺序为1010，其前缀1和前缀101不满足0的个数都不少于1的个数，但当排序为0101时，任意前缀均满足0的个数都不少于1的个数。所以就将该问题抽象成了卡特兰数问题。

2.n对括号，则有多少种 “括号匹配” 的括号序列.

思路：n对括号相当于有$2n$个符号，n个左括号、n个右括号，可以设问题的解为$f(2n)$。第0个符号肯定为左括号，与之匹配的右括号必须为第$2i+1$字符。因为如果是第$2i$个字符，那么第0个字符与第$2i$个字符间包含奇数个字符，而奇数个字符是无法构成匹配的。
通过简单分析，$f(2n)$可以转化如下的递推式$f(2n)=f(0)*f(2n-2)+f(2)*f(2n-4)+...+f(2n-4)*f(2)+f(2n-2)*f(0)$。简单解释一下，$f(0)*f(2n-2)$表示第0个字符与第1个字符匹配，同时剩余字符分成两个部分，一部分为0个字符，另一部分为$2n-2$个字符，然后对这两部分求解。$f(2)*f(2n-4)$表示第0个字符与第3个字符匹配，同时剩余字符分成两个部分，一部分为2个字符，另一部分为$2n-4$个字符。依次类推。
假设$f(0)=1$，计算一下开始几项，$f(2)=1,f(4)=2,f(6)=5$。结合递归式，不难发现$f(2n)$等于$h(n)$。

3.n个节点构成的二叉树，共有多少种情形？$h(n)$

思路：可以这样考虑，根肯定会占用一个结点，那么剩余的$n-1$个结点可以有如下的分配方式，$T(0,n-1),T(1,n-2),...T(n-1,0)$，设$T(i,j)$表示根的左子树含i个结点，右子树含$j$个结点。
设问题的解为$f(n)$，那么$f(n)=f(0)*f(n-1)+f(1)*f(n-2)+...+f(n-2)*f(1)+f(n-1)*f(0)$。假设$f(0)=1$，那么$f(1)=1,f(2)=2,f(3)=5$。结合递推式，不难发现$f(n)$等于$h(n)$。

4.求一个凸多边形区域划分成三角形区域的方法数？$h(n-2)$

思路：以凸多边形的一边为基，设这条边的2个顶点为A和B。从剩余顶点中选1个，可以将凸多边形分成三个部分，中间是一个三角形，左右两边分别是两个凸多边形，然后求解左右两个凸多边形。
设问题的解$f(n)$，其中n表示顶点数，那么$f(n)=f(2)*f(n-1)+f(3)*f(n-2)+...f(n-2)*f(3)+f(n-1)*f(2)$。$f(2)*f(n-1)$表示三个相邻的顶点构成一个三角形，那么另外两个部分的顶点数分别为2和$n-1$。
设$f(2)=1$，那么$f(3)=1,f(4)=2$, $f(5)=5$。结合递推式，不难发现$f(n)$ 等于$h(n-2)$。

5.在圆上选择$2n$个点，将这些点成对连接起来使得所得到的n条线段不相交的方法数？$h(n)$

思路：以其中一个点为基点，编号为0，然后按顺时针方向将其他点依次编号。那么与编号为0相连点的编号一定是奇数，否则，这两个编号间含有奇数个点，势必会有个点被孤立，即在一条线段的两侧分别有一个孤立点，从而导致两线段相交。设选中的基点为A，与它连接的点为B，那么A和B将所有点分成两个部分，一部分位于A、B的左边，另一部分位于A、B的右边。然后分别对这两部分求解即可。
设问题的解$f(n)$，那么$f(n)=f(0)*f(n-2)+f(2)*f(n-4)+f(4)*f(n-6)+...f(n-4)*f(2)+f(n-2)*f(0)$。$f(0)*f(n-2)$表示编号0的点与编号1的点相连，此时位于它们右边的点的个数为0，而位于它们左边的点为$2n-2$。依次类推。
$f(0)=1,f(2)=1,f(4)=2$。结合递归式，不难发现$f(2n)$ 等于$h(n)$。

9.BSGS算法

求解高次同余方程
给定整数 a, b, p，a, p 互质，
求满足 $a^x \equiv b \ (mod\ p)$ 的最小非负整数 x。

BSGS 算法 (Baby Step Giant Step)
由扩展欧拉定理 $a^x \equiv a^{x\ mod\ \varphi(p)}\ (mod\ p)$
可知 a^x 模 p 意义下的最小循环节为 $\varphi(p)$，
因 $\varphi(p) < p$，故考虑 $x \in [0, p]$，必能找到最小整数 x。

如果暴力枚举，时间是 $O(p)$ 的。

令 $x = im - j$，其中 $m = \lceil \sqrt{p} \rceil$，$i \in [1, m]$，$j \in [0, m-1]$
则 $a^{im-j} \equiv b\ (mod\ p)$，
即 $(a^m)^i \equiv ba^j\ (mod\ p)$。

1. 先枚举 j，把 (ba^j, j) 丢入哈希表，如果 key 重复，用更大的 j 替代旧的
2. 再枚举 i，计算$(a^m)^i$，到哈希表中查找是否有相等的 key，找到第一个即结束。则最小的 $x = im - j$

枚举 j, i 的次数都是$\sqrt{p}$ 的，所以时间是 $O(\sqrt{p})$ 的。

先证明一个结论：若原方程有解，则$x \in [0, P-1)$一定有解。

因为$\gcd(A, P) = 1$，由费马小定理得$A^{P-1} \equiv 1 (\mod P)$。

则有$A^{x+k(P-1)} \equiv A^x (\mod P)$，其中$k$为整数。

换句话说就是出现了循环。这样一来，这个方程的最小解就一定在$[0, P-1)$中了。

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long LL;

LL bsgs(LL a, LL b, LL p){
  a %= p; b %= p;
  if(b == 1) return 0; //x=0
  LL m = ceil(sqrt(p));
  LL t = b;
  unordered_map<int,int> hash;
  hash[b] = 0;
  for(int j = 1; j < m; j++){
    t = t*a%p; //求b*a^j
    hash[t] = j;
  }//一小步

  LL mi = 1;
  for(int i = 1; i <= m; i++)
    mi = mi*a%p; //求a^m
  t = 1;
  for(int i=1; i <= m; i++){
    t = t*mi%p; //求(a^m)^i, 一大步
    if(hash.count(t))
      return i*m-hash[t];
  }
  return -1; //无解
}
int main(){
  int a, p, b;
  cin >> p >> a >> b;
  int res=bsgs(a, b, p);
  if(res==-1) puts("no solution");
  else cout << res << endl;
  return 0;
}
```

P3846 [TJOI2007] 可爱的质数/【模板】BSGS