数论基础选讲

山东省实验中学

宁华

问题引入：

- 一个笼子里装着鸡和兔子，已知一共有20只脚，问鸡和兔子的数量分别有多少？

问题引入：

- 一个笼子里装着鸡和兔子，已知一共有20只脚，问鸡和兔子的数量分别有多少？

- 解：设鸡有x只，兔子有y只，则有
- $2x + 4y = 20$

- 如何求解这个方程呢？

裴蜀等式

- 法国数学家艾蒂安·裴蜀
- 线性不定方程ax + by = c（又被称为裴蜀等式）是一个丢番图方程。

- 丢番图方程（Diophantine Equation）：有一个或者几个变量的整系数方程，它们的**求解仅仅在整数范围内进行**。最后这个限制使得丢番图方程求解与实数范围方程求解有根本的不同。丢番图方程又名不定方程、整系数多项式方程，是**变量仅容许是整数**的多项式等式。

如何求解 ax + by = c

- 两个问题：
- 1、方程一定有解吗？
- 2、如果方程有解，该如何求解？

1、方程一定有解吗？

- 例如：
- 2x + 4y = 20 有解吗？
- 2x + 4y = 15 有解吗？

裴蜀定理 Bézout's identity 又称贝祖定理

- 裴蜀定理得名于法国数学家艾蒂安·裴蜀。

- ax+by=c 有解的充要条件是 gcd(a,b)|c。

ax+by=c 有解的充要条件是 gcd(a,b)|c。

- 必要性
- d=gcd(a,b)，则d|a, d|b
- 则对任意整数 x, y，有 d|ax+by 成立，即：d|c

- 充分性
- d|c，则对于 d 的倍数 c，要证明 ax+by=c，只需证明 ax+by=d
- （可通过扩展欧几里得构造出解）

- 例如，
- 2x + 4y = 15 无解，因为 2 和 4 的最大公约数是 2，2 不能整除 15
- 2x + 4y = 20 有解，因为 2 | 20，所以原方程一定有解。
- 事实上有
- 2×(-2) + 4×6 = 20,
- 2×0 + 4×5 = 20,
- 2×2 + 4×4 = 20,
- 2×4 + 4×3 = 20,
- ……

推论

- 特别地，ax+by=gcd(a,b) 一定有解

- 例如，2和4的最大公约数是2，则方程 2x + 4y = 2 一定有解。
- 事实上有
- 2×(-3) + 4×2 = 2,
- 2×1 + 4×0 = 2,
- 2×3 + 4×(-1) = 2,
- ……

问题：

- $3x+5y=1$
- $2x+4y=1$
- $8x+9y=1$
- ....
- $ax+by=1$

- 如何判断以上方程是否有解？

- 推论：ax+by=1有解 当且仅当 a,b互质

- 裴蜀等式也可以用来给最大公约数定义：d其实就是最小的可以写成ax + by形式的正整数。这个定义的本质是整环中“理想”的概念。

2、如果 ax+by=c 有解，该如何求解？

- 裴蜀定理给出了判断裴蜀等式是否有解的条件。那么当裴蜀等式有解时，如何求出它的解呢？

- 裴蜀等式有解时必然有无穷多个整数解，每组解x、y都称为裴蜀数。

2、如果 ax+by=c 有解，该如何求解？

- 如果能求出一组特解x1,y1，则通解为：
- x=x1+(b/d)*k
- y=y1-(a/d)*k
- 其中d=gcd(a,b), k为任意整数。明显，b/d与a/d是互素的。

- 思考：为什么通解不可以是：
- x=x1+b*k
- y=y1-a*k
- 呢？

- 如何求出 ax+by=c 的一组特解x1, y1呢？

- 记d=gcd(a,b)
- 如果能求出 ax + by = d 的一组特解 x0, y0，那么 ax + by = c 的一组特解为 x1=x0*c/d, y1=y0*c/d

问题

- 如何求出 ax+by=d 的一组特解呢？这里d=gcd(a,b)

扩展欧几里德算法

- 如何求出 ax+by=d 的一组特解呢？这里d=gcd(a,b)
- 考虑使用欧几里德算法的思想，令a=bq+r，其中r=a mod b；递归求出bx+ry=d的一个解。
- 设求出bx+ry=d的一个解为$x=x_0,y=y_0$，考虑如何把它变形成ax+by=d的解。
- 将a=bq+r代入ax+by=d、化简得b(xq+y)+rx=d
- 我们令$xq+y=x_0,x=y_0$，则上式成立
- 故$x=y_0,y=x_0-y_0q$为ax+by=d的解
- 边界情况：b=0时，令x=1,y=0

```
#include<iostream>
using namespace std;
int exgcd(int a,int b,int &x,int &y)
{
	if(b==0)
	{
		x=1;
		y=0;
		return a;
	}
	int gcd=exgcd(b,a%b,x,y);
	int t=x;
	x=y;
	y=t-(a/b)*y;
	return gcd;
}
int main()
{
		int a,b,x,y;
		cin>>a>>b;
		cout<<exgcd(a,b,x,y)<<endl;
		cout<<x<<" "<<y<<endl;
		return 0;
}
```

扩展欧几里德算法

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

总结：

- 至此，ax + by = c 的求解得以解决。

- 1、利用exgcd求ax+by=gcd(a,b)的一组解x0,y0
- 2、判断gcd(a,b)能否整除c，若不能，则原方程无解。否则进入第3步。
- 3、得到ax+by=c的一组特解x1=x0*c/d, y1=y0*c/d
- 4、得到ax+by=c的通解x=x1+b/d*k, y=y1-a/d*k

- 另：也可求出ax+by=gcd(a,b)的通解后得到ax+by=c的通解。

问题：怎么求ax+by=c的x的最小正整数解？

- 刚才已经得到 ax + by = c 的通解：
- x=x0*c/d+b/d*k, y=y0*c/d-a/d*k 其中d=gcd(a,b)

- 令 b1 = b/d , 则 x 的 最小正整数解 为 ：
- x1 = (x0*c/d % b1 + b1) % b1
- 对应的 y = (c - a*x1) / b.

#1581. [POJ2115] C Looooops

- for(i=A;i!=B;i+=C){i%=(2^k)};
- 输入A，B，C，k，问你循环多少次？
- Sample Input
- 3 3 2 16
- 3 7 2 16
- 7 3 2 16
- 3 4 2 16
- 0 0 0 0
- Sample Output
- 0
- 2
- 32766
- FOREVER

#1582. [POJ1061] 青蛙的约会

- 两只青蛙沿赤道同朝西跳，第一只青蛙的起点是x, 每次跳m米，第二只从y开始每次跳n米，赤道长度为L，两只青蛙每次均同时起跳，同时落地，问两只青蛙最少几步相遇？
- Input
- 输入只包括一行5个整数x，y，m，n，L，其中x≠y < 2000000000，0 < m、n < 2000000000，0 < L < 2100000000。
- Output
- 输出碰面所需要的最少跳跃次数，如果永远不可能碰面则输出一行"Impossible"
- Sample Input
- 1 2 3 4 5
- Sample Output
- 4

#1583. [POJ 2142] The Balance

- 有两种类型的砝码，质量分别为 a 和 b，每种砝码均有无穷多个。
- 有一个天平，左右两端均可放砝码和物品。
- 现在要在天平上称出质量为 d 的物品。
- 问：怎样放置砝码，才能使放置的砝码数量尽可能的少？
- 若有多种放法的砝码数量相同，则使得放置砝码的总质量尽可能小。
- 请输出满足以上条件的情况下所使用的两种砝码的数量。

多元一次不定方程

- $a_1x_1+a_2x_2+a_3x_3+...+a_nx_n=c$

数学里的倒数

- a*b=1
- 则称b是a的倒数，a是b的倒数
- 或者说a、b互为倒数。

- a的倒数可写为1/a、a^-1

数论倒数（逆元）

- 数论倒数，又称逆元
- 数论中的倒数是有特别的意义滴

- 你以为a的倒数在数论中还是1/a吗

- (･∀･)哼哼~天真

先来引入求余概念

- (a + b) % p = (a%p + b%p) %p （对）
- (a - b) % p = (a%p - b%p) %p （对）
- (a * b) % p = (a%p * b%p) %p （对）
- (a / b) % p = (a%p / b%p) %p （错）

- 为什么除法是错的？
- 证明是对的难，证明错的只要举一个反例
- (100/50)%20 = 2　≠　(100%20) / (50%20) %20 = 0

可是有时候我们就是需要进行带除法的求余运算

- 比如，求：123456789*987654321*23456789*87654321/365*3456789*456789/567*7654321 % 123

- 对于一些题目，我们必须在中间过程中进行求余，否则数字太大，电脑存不下，那如果这个算式中出现除法，我们是不是对这个算式就无法计算了呢？

- 答案当然是 NO (>o<)

- 这时就需要逆元了

- 我们知道
- 如果a*x = 1
- 那么x是a的倒数，x = 1/a
- 但是a如果不是1，那么x就是小数

- 但是在数论中，怎么能有小数呢

- 数论中，大部分情况都有求余
- 所以现在问题变了
- a*x = 1 (mod p)
- 那么x一定等于1/a吗
- 不一定
- 所以这时候，我们就把x看成a的倒数，只不过加了一个求余条件，所以x叫做：a关于p的逆元

- 比如2 * 3 % 5 = 1，那么3就是2关于5的逆元，或者说2和3关于5互为逆元
- 这里3的效果是不是跟1/2的效果一样，所以才叫数论倒数
- a的逆元，我们用inv(a)来表示

- 那么
- (a / b) % p
- = (a * inv(b)) % p
- = (a % p * inv(b) % p) % p

- 这样就把除法，完全转换为乘法了。
- (･ω･)，乘法超容易

正儿八经给逆元下个定义：

- 若$ax \equiv 1 \pmod p$，则称x是a关于模p的逆元，常记做a^{-1}。

- 今后我用inv(a)表示a的逆元。

- 一般用于求 a/b mod p
- 转换为求 a*inv(b) mop p

- 正篇开始
- 逆元怎么求
- (忘了说，a和p互质，a才有关于p的逆元)
- 例如 $4x\equiv 1 \pmod 2$
- 怎么可能嘛~~~

求逆元方法1：

- 费马曾经说过：不想当数学家的数学家不是好数学家（(￣▽￣)~*我随便说的，别当真）

- 费马小定理(Fermat's little theorem)是数论中的一个重要定理，在1636年提出，其内容为：
- 若p是质数，且a,p互质，那么 a^(p-1)除以p的余数恒等于1。
- 即：a^(p-1)≡1(mod p)

- 费马小定理
- a^(p-1) ≡1 (mod p)
- 两边同除以a
- a^(p-2) ≡1/a (mod p)
- 什么(,,• ₃ •,,)，这可是数论，还敢写1/a
- 应该写a^(p-2) ≡ inv(a) (mod p)
- 所以inv(a) = a^(p-2) (mod p)
- 这个用快速幂求一下，复杂度O(logn)
- (ง •̀_•́)ง

求逆元方法1：

```
typedef long long LL;
LL pow_mod(LL a, LL b, LL p){ // 求 a^b %p
    LL ret = 1;
    while(b){
        if(b & 1) ret = (ret * a) % p;
        a = (a * a) % p;
        b >>= 1;
    }
    return ret;
}
LL Fermat(LL a, LL p){ //费马小定理求a关于p的逆元 前提: p是质数，且a,p互质
        return pow_mod(a, p-2, p);
}
```

求逆元方法2：

- **如果求a关于b的逆元，但是b不是质数，费马小定理就失效了。**
- 此时，扩展欧几里德算法出场 ~
- 还记得扩展欧几里德吗？（不记得的话，欧几里德会伤心的(╭ ͡° ͜ʖ ͡°) ╭♡ ）

- a*x + b*y = 1
- 如果a、b互质，则方程有解
- 这个解的x就是a关于b的逆元
- y就是b关于a的逆元
- 为什么呢？

- a*x + b*y = 1
- 两边同时模b
- a*x % b + b*y % b = 1 % b
- a*x % b = 1 % b
- a*x = 1 (mod b)
- 你看你看，出现了！！！(/≥▽≤/)
- 所以x是a关于b的逆元
- 同样可证明y是b关于a的逆元

再看逆元的定义

- 若$ax \equiv 1 \pmod b$，则称x是a关于模b的逆元，常记做a^{-1}。

- 回忆同余的性质。**上式等价于ax+by=1**

- 如何求逆元？等价于解方程ax+by=1
- 因此逆元不一定存在：
  存在的充要条件为(a,b)=1
- 推论：p是质数，p不整除a，则a模p的逆元存在。

逆元

- 结论：在[0,b)的范围内，a关于模b的逆元(若存在)是唯一的。
- 证明：
- 反证法，若a有两个逆元$0<x1<x2<b$，
  即$ax_1 \equiv ax_2 \equiv 1 \pmod b$，
- 那么有$b|a(x_2-x_1)$成立
- 又由于(a,b)=1，因此$b|x_2-x_1$。
- 其中$0<x_2-x_1<b$，产生了矛盾。

逆元

```
// 利用exgcd求逆元
int get_inv(int a, int b) {
    int x, y;
    int d=exgcd(a, b, x, y);
    if(d!=1)return -1;
    return x;// return (x%b+b)%b;
}
```

```
#include<cstdio>
typedef long long LL;
void ex_gcd(LL a, LL b, LL &x, LL &y, LL &d){
    if (!b) {d = a, x = 1, y = 0;}
    else{
        ex_gcd(b, a % b, y, x, d);
        y -= x * (a / b);
    }
}
LL inv(LL t, LL p){//如果不存在，返回-1
    LL d, x, y;
    ex_gcd(t, p, x, y, d);
    return d == 1 ? (x % p + p) % p : -1;
}
int main(){
    LL a, p;
    while(~scanf("%lld%lld", &a, &p)){
        printf("%lld\n", inv(a, p));//求a关于p的逆元, p不必是质数
    }
}
```

求逆元方法3：

- 求 i 关于p的逆元

- 当p是个质数的时候有
- inv(i) = (p - p / i) * inv(p % i) % p

- 这为啥是对的咩？

- 证明不想看的孩子可以跳过。。。（￣0￣）

i 关于p的逆元 inv(i) = (p - p / i) * inv(p % i) % p

- 证明：
- p ÷ i = k ... r
- p = k * i + r 其中 r = p % i, k = p / i
- (r + k * i) % p = 0
- 移项得 ---------- r % p = - k * i % p
- 两边同乘以 inv(i) 得 -------- r * inv(i) % p = -k % p
- 两边同乘以 inv(r) 得 -------- inv(i) = - k * inv(r) % p
- 所以 ---------- inv(i) = - p/i * inv(p%i) %p
- 于是 ---------- inv(i) = (p - p/i) * inv(p%i) % p

- 然后一直递归到1为止，因为1的逆元就是1

递归

```
#include<cstdio>
typedef long long LL;
LL inv(LL a, LL p) {//求a关于p的逆元，注意: a要小于p，传参前先把a%p一下
    return a == 1 ? 1 : (p - p / a) * inv(p % a, p) % p;
}
int main(){
    LL a, p;
    while(~scanf("%lld%lld", &a, &p)){
        printf("%lld\n", inv(a%p, p));//前提:p是质数
    }
}
```

- 如果要求 1 ~ n 中所有整数在模 p 意义下的乘法逆元呢？
- 递推：
- ```
  for (inv[1] = 1, i = 2; i < p; ++i)
      inv[i] = (p - p / i) * inv[p % i] % p;
  ```
  ```

线性求逆元

- 线性求逆元，1~n 关于p的逆元都求出来了。
- ```
  inv[1]=1;
  for(int i=2;i<p;i++)
      inv[i]=(p-p/i)*inv[p%i]%p;
  for(int i=p;i<=n;i++)
      if(gcd(i,p)!=1)inv[i]=-1;//互质才有逆元
      else inv[i]=inv[i%p];
  for(int i=1;i<=n;i++)
      printf("%d\n",inv[i]);
  ```
  ```

线性求 1!, 2!, … n! 关于p的逆元 (p是质数且n<p)

- 方法1：倒推
- 先求出fac[i] = i! % p
- 再求inv_fac[i] 表示 i! 的逆元（利用exgcd或快速幂）
- 再倒推就可以了。inv_fac[i] = inv_fac[i + 1] * (i + 1) % p

```
int inv( int b, int p ) {
    int a, k;
    exgcd( b, p, a, k );
    if( a < 0 ) a += p;
    return a;
}
void init( int n ) {
    Fact[ 0 ] = 1;
    for( int i = 1; i <= n; ++i ) Fact[ i ] = Fact[ i - 1 ] * i % Mod;
    INV[ n ] = inv( Fact[ n ], Mod );
    for( int i = n - 1; i >= 0; --i ) INV[ i ] = INV[ i + 1 ] * ( i + 1 ) % Mod;
    return;
}
```

- 方法2：顺推
- 先求出 i 模p的逆元:
- inv[i]=(p−p/i)*inv[p%i]%p
- 然后求 i!的逆元
- inv_fac[i]=inv_fac[i−1]*inv[i]%p

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

例题：组合数取模

- 回答T次询问
- 每次给出两个数n、k，询问C(n, k) mod 998244353(一个质数)

- $T \le 10^5$
- $0 \le k \le n \le 10^7$，

分析：

- C(n, k)=n!/(k!(n-k)!)
- 线性求逆，预处理n!以及n!的逆元
- O(1)回答询问

LUCAS定理(求组合数取模)

- 定义 : n,m是非负整数， p是素数时 , Lucas(n,m)=C(n,m)%p
- 公式 :

$$Lucas(n,m) = C_{n\%p}^{m\%p} * Lucas(n/p, m/p)\%p$$

- 代码 :
- LL Lucas(LL n,LL m){
- if(m==0)return 1ll;
- return C(n%mod,m%mod)*Lucas(n/mod,m/mod)%mod;
- }

例 NOIP2012D2T1 同余方程

- 描述
- 求关于x的同余方程ax ≡ 1 (mod b)的最小正整数解。
- 输入格式
- 输入只有一行，包含两个正整数a, b，用一个空格隔开。
- 输出格式
- 输出只有一行，包含一个正整数x0，即最小正整数解。输入数据保证一定有解。
- 样例输入
- 3 10
- 样例输出
- 7
- 数据规模
- 对于40%的数据，2 ≤b≤ 1,000；
- 对于60%的数据，2 ≤b≤ 50,000,000；
- 对于100%的数据，2 ≤a, b≤ 2,000,000,000。

方法1：

- ax ≡ 1 (mod b) ========> ax + by = 1
- 求 ax + by = 1 的 x 的最小正整数解

```
scanf("%d%d",&a,&b);
exgcd(a,b,x,y);
while(x<0) x+=b; // x = (x%b+b)%b;
printf("%d\n",x);
```

方法2：

- 逆元

例 JZOJ 5809 数羊

- 问题描述
- 牧羊人A和牧羊人B总是很无聊，所以他们要玩一个游戏。A有a只羊，B有b只羊。他们想要知道a^b的因子和是多少。这就很为难两个牧羊人了，由于答案太大，你能不能告诉我答案取模9901的数。
- 输入
- 仅一行，为两个正整数a和b
- 输出
- a^b的因子和对9901的余数。
- 样例输入
- 2 3
- 样例输出
- 15

- 将a进行质因数分解，将每个质因子的指数乘上b，

- 最后的答案就是等比数列的和，但是由于涉及取模，所以我们除法要用逆元。

HDU 4828 Grids

- Problem Description
- 度度熊最近很喜欢玩游戏。这一天他在纸上画了一个2行N列的长方形格子。他想把1到2N这些数依次放进去，但是为了使格子看起来优美，他想找到使每行每列都递增的方案。不过画了很久，他发现方案数实在是太多了。度度熊想知道，有多少种放数字的方法能满足上面的条件？
- Input
- 第一行为数据组数T(1<=T<=100000)。
- 然后T行，每行为一个数N(1<=N<=1000000)表示长方形的大小。
- Output
- 对于每组数据，输出符合题意的方案数。由于数字可能非常大，你只需要把最后的结果对1000000007取模即可。

- Sample Input
- 2
- 1
- 3
- Sample Output
- Case #1:
- 1
- Case #2:
- 5

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

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

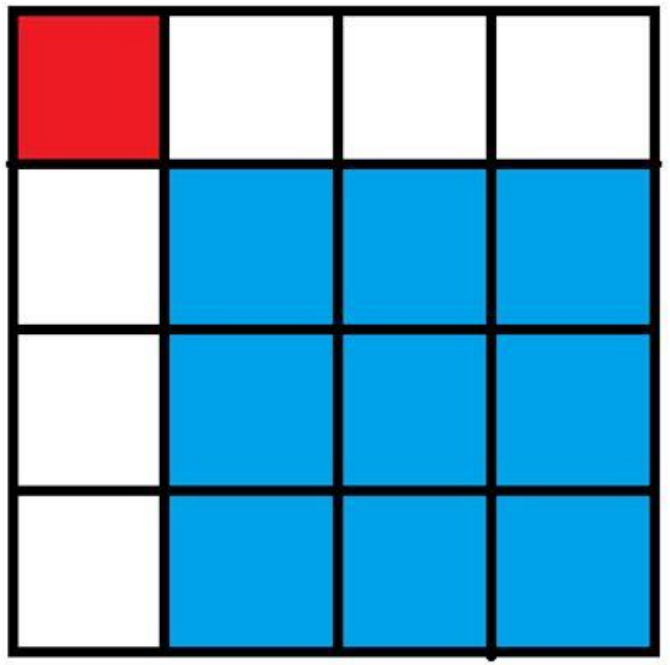
1	3	5
2	4	6

- 假设0代表这个数放在第一排，1代表这个数放在第二排
- 序列 00001111 就是
- 1 2 3 4
- 5 6 7 8
- 序列 01010101 就是
- 1 3 5 7
- 2 4 6 8
- 这个问题就转化成有几种满足题目条件的01序列
- 通过观察发现每个位置之前0的个数大于等于1的个数，如果把0看成入栈, 1看成出栈, 这个问题变转化成n个数的入栈的出栈次序的种类数，也就是卡特兰数

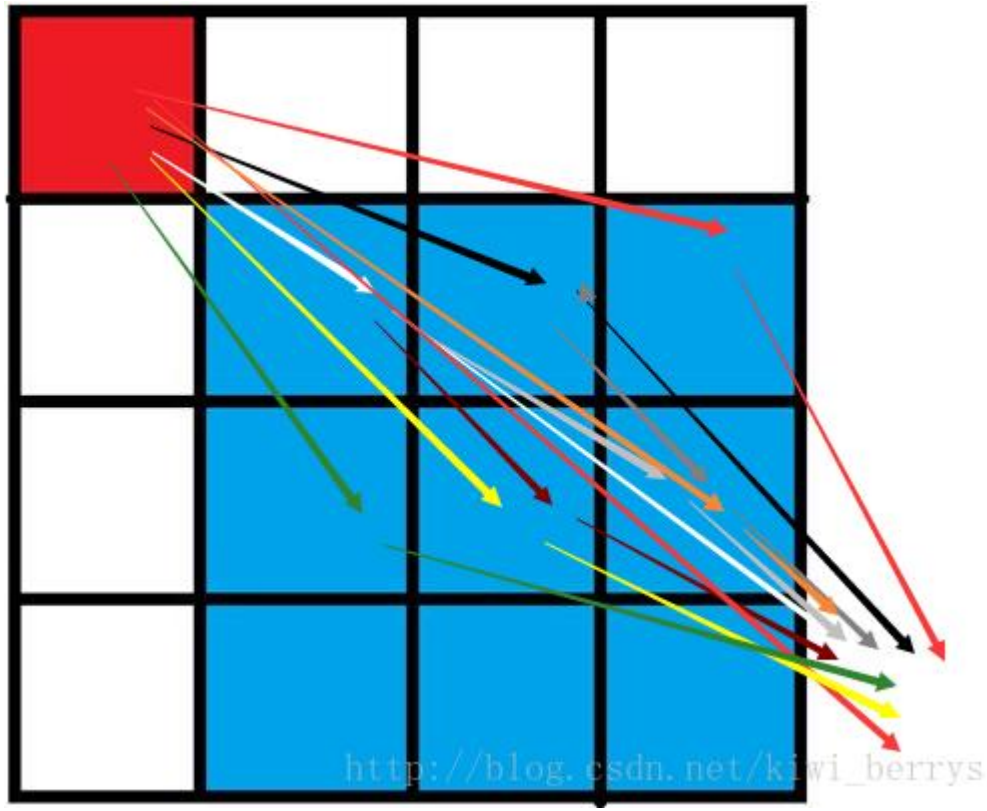
- $h(n)= h(0)*h(n-1)+h(1)*h(n-2) + ... + h(n-1)*h(0)\ (n\geq 2)$
- $h(n)=h(n-1)*(4*n-2)/(n+1)$
- $h(n+1)=h(n) * (4*n + 2) / (n + 2)$
- $h(n)=C(2n,n)/(n+1)\ (n=0,1,2,...)$
- $h(n)=C(2n,n) - C(2n,n-1)\ (n=0,1,2,...)$

例 HDU5698 瞬间移动

- 有一个无限大的矩形，初始时你在左上角（即第一行第一列），每次你都可以选择一个右下方格子，并瞬移过去（如从下图中的红色格子能直接瞬移到蓝色格子），求到第n行第m列的格子有几种方案，答案对1000000007取模。



- Input
- 多组测试数据。
- 两个整数n,m(2<= n,m<=100000)
- Output
- 一个整数表示答案
- Sample Input
- 4 5
- Sample Output
- 10



打表找规律

- 首先自己写4组数据
- n m ans
- 5 2 1
- 5 3 4
- 5 4 10
- 5 5 20

- 然后杨辉三角打表
- 1
- 1 1
- 1 2 1
- 1 3 3 1
- 1 4 6 4 1
- 1 5 10 10 5 1
- 1 6 15 20 15 6 1

- 我们发现规律就是求组合数C(n+m-4,m-2)

- 我们要从(1,1)点走到(n,m)点，只能向右下角的格子里走，那么(1,1)所在的行和列不能走,(n,m)所在的行和列不能走，那么能走的面积就是(n−2)*(m−2)这么大的面积，行和列分别考虑，实质上就是求组合数C(n−2, n−2+m-2)

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mod=1e9+7;
const ll N=200000+20;
ll fac[N]= {1,1},inv[N]= {1,1},f[N]= {1,1};
/*
fac[i]:i的阶乘
inv[i]:i的阶乘的逆元
f[i]:i的逆元
*/
ll C(ll a,ll b)//除以一个数等于乘这个数的逆元
{
    return fac[a]*inv[b]%mod*inv[a-b]%mod;
}
```

```
void init()
{
    for(ll i=2; i<N; i++)
    {
        fac[i]=fac[i-1]*i%mod;
        f[i]=(mod-mod/i)*f[mod%i]%mod;
        inv[i]=inv[i-1]*f[i]%mod;
    }
}

int main()
{
    ll n,m;
    init();
    while(cin>>n>>m)
        cout<<C(m+n-4,m-2)<<endl;
    return 0;
}
```

HDU 5184

Codeforces 521C (经典)组合数取模【逆元】

线性同余方程

- 形如$ax \equiv c \pmod b$的方程，称为线性同余方程。
- 等价于$ax+by=c$；因此有解条件为$(a,b)|c$

- 若$(a,b)=1$，则x有唯一解$x \equiv a^{-1} c \pmod b$。
- 否则设$(a,b)=d$，$a=a'd$, $b=b'd$, $c=c'd$
- 那么有$a'x+b'y=c'$，即$a'x \equiv c' \pmod{b'}$
- 这里$(a', b')=1$，因此有$x \equiv (a')^{-1} c \pmod{b'}$

- 综上，任意的线性同余方程总可以判定为无解、或化为$x \equiv a \pmod m$的形式。

线性同余方程组

- 考虑形如$x \equiv a_i \pmod{m_i}$的若干方程联立得到的方程组，如：
  - $x \equiv 2 \pmod 3$ ..... (1)
  - $x \equiv 3 \pmod 5$ ..... (2)
  - $x \equiv 5 \pmod 7$ ..... (3)
- 下面是一种可行的解法：
  - 由(1)设$x=3y+2$,代入(2)得到$3y+2 \equiv 3 \pmod 5$,解得$y \equiv 2 \pmod 5$
  - 设$y=5z+2$,代入(3)得到$3(5z+2)+2 \equiv 5 \pmod 7$,解得$z \equiv 4 \pmod 7$
  - 设$z=7k+4$,则$x=3(5(7k+4)+2)+2=105k+68$
  - 因此$x \equiv 68 \pmod{105}$