数学知识

# 矩阵乘法

线性代数 ： 将一行数和矩阵数相乘，行里面有几个数字，矩阵就是几 *几的规格，对于每一列，将行竖起来乘列，一次和矩阵中的每一列对应相乘最后相加即是下一次结果行，按顺序填入，例如斐波那契，Fn = [fn,fn+1,Sn]* [011,111,001] = Fn+1 = [fn+1,fn+2,sn+1];中间的是按行写的 -----> Fn *A = Fn+1 ----> Fn = F1* A^(n-1),矩阵相乘是第一个的第 i 行乘于第 二 个的第 j 列

## 定理

矩阵乘法具有结合率，即是A*B*C = A*(B* C)，矩阵乘法没有交换律，矩阵乘法满足 （A *B）* (B *C) = A* C,

矩阵乘法 + 快速幂，A *B X B* C = A \* C 即是 A 行 B 列的矩阵乘于 B 行 C 列的矩阵得到 A 行 C 列的矩阵，计算原则 ：将前面的矩阵每一行顺时针翻转90度，和第二个矩阵对应的位置相乘求和为一个位置的结果一次填入到结果矩阵的位置中，先行后列，从左到右(第 i 行乘于第 j 列得到的是(i,j)位置的值)

### 伪代码

for(int i=1;i<=A;i++)  
{  
 for(int j=1;j<=C;j++)  
 {  
 for(int k=1;k<=B;k++)  
 {  
 R[i][j] += P[i][k] \* Q[k][j];  
 }  
 }  
}  
复制

## 例题 1 1303 斐波那契的前 n 项和

#include<bits/stdc++.h>  
using namespace std;  
#define \_ ios::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
#define endl '\n'  
#define ll long long   
const int N = 3;  
int n,m;  
void mul(int c[],int a[],int b[][N])  
{  
 int temp[N] = {0};  
 for(int i=0;i<N;i++)  
 {  
 for(int j=0;j<N;j++)  
 {  
 temp[i] = (temp[i] + (ll)a[j] \* b[j][i]) % m;  
 }  
 }  
 memcpy(c,temp,sizeof temp);//尺寸只能用全局变量或者矩阵变量，传 c 的话是指针长度不是数组长度  
}  
void mul(int c[][N],int a[][N],int b[][N])//矩阵乘法  
{  
 int temp[N][N] = {0};  
 for(int i=0;i<N;i++)  
 {  
 for(int j=0;j<N;j++)  
 {  
 for(int k=0;k<N;k++)  
 {  
 temp[i][j] = (temp[i][j] + (ll)a[i][k] \* b[k][j]) % m;  
 }  
 }  
 }  
 memcpy(c,temp,sizeof temp);  
}  
int main()  
{  
 \_;  
 cin >> n >> m;  
 int f1[3] = {1,1,1};//f1 = 1,f2 = 1 斐波那契  
 int a[N][N] =   
 {  
 {0,1,0},  
 {1,1,1},  
 {0,0,1}  
 };  
 n --;//求得是 n - 1 次方  
 while(n)  
 {  
 if(n & 1) mul(f1,f1,a);// res = res \* a (a 是矩阵)  
 mul(a,a,a);// a = a \* a;  
 n >>= 1;  
 }  
 cout << f1[2] << endl;  
}// 5 1000 ---> 12  
复制

## 例题 2 1304 佳佳的斐波那契

Fn = F1 *A ^ (n - 1) ,求 Tn = F1 + 2* F2 + ... + n *Fn , n* Sn - Tn = (n - 1) *F1 + (n-2)* F2 + ... + F(n-1)...①,(n+1) *S(n+1)-T(n+1) = n* F1 + (n-1) *F2 + ... + Fn ...②, ② - ① = Sn， 引入 Pn = n* Sn - Tn,则有Pn = n \* n - Tn,Pn = P(n-1) + S(n-1),Sn = S(n-1) + fn,fn = f(n-1) + f(n-2) ---->

Fn = [fn,f(n+1),Sn,Pn] \* {0100,1110,0011,0001} = F(n+1) = [f(n+1),f(n+2),S(n+1),P(n+1)]，结合快速幂，F1 = [1,1,1,0],Fn = F1 \* A^(n-1)  
复制

#include<bits/stdc++.h>  
using namespace std;  
#define \_ ios::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
#define endl '\n'  
#define ll long long   
const int N = 4;  
int n,m;  
void mul(int c[][N],int a[][N],int b[][N])  
{  
 static int t[N][N];//辅助矩阵，防止传入aaas时改变了a但是后面的a不应该改变  
 memset(t,0,sizeof t);//这里一定要写t，不能写c，写c返回的是指针长度，而非数组长度  
 for(int i=0;i<N;i++)  
 {  
 for(int j=0;j<N;j++)  
 {  
 for(int k=0;k<N;k++)  
 {  
 t[i][j] = (t[i][j] + (ll)a[i][k] \* b[k][j]) % m;  
 }  
 }  
 }  
 memcpy(c,t,sizeof t);  
}  
int main()  
{  
 \_;  
 cin >> n >> m;  
 //{fn,fn+1,sn,pn},pn = n \* sn - tn  
 int f1[N][N] = {1,1,1,0};  
 int a[N][N] = {  
 {0,1,0,0},{1,1,1,0},  
 {0,0,1,1}, {0,0,0,1}  
 };  
 int k = n - 1;  
 while(k)  
 {  
 if(k & 1)  
 mul(f1,f1,a);//f1=f1\*a  
 mul(a,a,a);  
 k >>= 1;  
 }  
 cout << (((ll)n \* f1[0][2] - f1[0][3]) % m + m) % m << endl;  
 return 0;  
}// 5 5 ---> 1  
复制

## 例题 3 1305 GT考试

准考证为n位数，不吉利数为m位，开头都可以是 0 ，求准考证号中不出现不吉利数的个数，对 k 取模

#include<bits/stdc++.h>  
using namespace std;  
#define \_ ios::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
#define endl '\n'  
#define ll long long   
const int N = 25;  
int n,m,mod;  
char str[N];  
int ne[N];  
int a[N][N];  
//f[i][j]表示长度是 i 且不含 S 串，且末尾部分与 S 匹配的最大长度是 j 的所有字符串的集合  
//f[i + 1][k] += f[i][j],加上一个字母后匹配的开头一定不会前移，f[i+1][0]=(a00\*f[i][0]+a01\*f[i][1]..)  
//f[i+1][1]=a10\*f[i][0] + a11\*f[i][1]...令F[i+1]看做f[i+1][0]~[m-1]的向量  
//A 即是 a00,a01,....构成的矩阵，转换为矩阵乘法  
void mul(int c[][N],int a[][N],int b[][N])  
{  
 static int t[N][N];  
 memset(t,0,sizeof t);  
 for(int i=0;i<m;i++)  
 {  
 for(int j=0;j<m;j++)  
 {  
 for(int k=0;k<m;k++)  
 {  
 t[i][j] = (t[i][j] + a[i][k] \* b[k][j]) % mod;  
 }  
 }  
 }  
 memcpy(c,t,sizeof t);  
}  
  
int qmi(int k)  
{  
 int f0[N][N] = {1};f0[0][0];//算一中方案  
 while(k)  
 {  
 if(k & 1)//f0=f0\*a   
 mul(f0,f0,a);  
 mul(a,a,a);//a=a\*a  
 k >>= 1;  
 }  
 int res = 0 ;  
 for(int i=0;i<m;i++)  
 res = (res + f0[0][i]) % mod;  
 return res;  
}  
  
int main()  
{  
 \_;  
 cin >> n >> m >> mod;  
 cin >> str + 1;  
 //kmp  
 for(int i=2,j=0;i<=m;i++)  
 {  
 while(j && str[j + 1] != str[i]) j = ne[j];  
 if(str[j + 1] == str[i]) j ++;  
 ne[i] = j;  
 }  
  
 //初始化 A ,有多少个可以转移过来，结合KMP即可获得f[i+1][k]有fij转移过来的系数  
 for(int j=0;j<m;j++)  
 {  
 for(int c='0';c<='9';c++)  
 {  
 int k = j;  
 while(k && str[k + 1] != c)   
 k = ne[k];  
 if(str[k + 1] == c)  
 k ++;  
 if(k < m)//k==m包含不满足  
 a[j][k] ++;  
 }  
 }  
 //Fn = F0 \* A^n  
 cout << qmi(n) << endl;  
}// 4 3 100 111 ---- > 81  
复制

# 扩展中国剩余定理

#include <bits/stdc++.h>  
using namespace std;  
typedef \_\_int128 ll;  
const int N = 1e5 + 10;  
ll x, y, d; int n;  
  
void exgcd(ll &x, ll &y, ll a, ll b)   
{  
 if(!b) d = a, x = 1, y = 0;  
 else exgcd(y, x, b, a % b), y -= a / b \* x;  
}  
  
ll gcd(ll a, ll b)   
{  
 return b ? gcd(b, a % b) : a;  
}  
  
ll lcm(ll a, ll b)   
{  
 return a / gcd(a, b) \* b;  
}  
  
ll A1, b, A2, B;  
  
void merge()   
{  
 exgcd(x, y, A1, A2);  
 ll c = B - b;  
 if(c % d) puts("-1"), exit(0);  
 x = x \* c / d % (A2 / d);//x \* c / d是代表当前值的解，通解是x = x0 + a2 / d,所以最小解就是 %   
 if(x < 0) x += A2 / d;  
 ll mod = lcm(A1, A2);//下一步需要将A1变为A1和A2的最小公倍数  
 b = (A1 \* x + b) % mod; // x = a1\*x+b + k\*[a1,a2]，[]是最小公倍数 -> x = x0 + ka  
 if(b < 0) b += mod;  
 A1 = mod;//a1 a2的最小公倍数 //上式 x0即是更新后的m ，a即是更新后的a1  
   
}  
  
int main() {  
 cin >> n;  
 for(int i = 1 ; i <= n ; ++ i)   
 {  
 long long \_A, \_B;  
 cin>>\_A>>\_B, A2 = \_A, B = \_B;  
 if(i > 1) merge();  
 else A1 = A2, b = B;  
 }  
 printf("%lld\n", (long long)(b % A1));  
 return 0;  
}  
复制

# 同余

## 例题 1 202 最幸运的数字

最少多少个8连在一起组成的正整数是L的倍数，L|888.. ----> L|(8*1...1) = 8*(9...9)/9 = 8*(10^x-1)/9, ----> 9L|8*(10^x-1),d=gcd(L,8) ----> (9L/d)|(10^x-1) ----> 10^x 同余于 1 (mod c) c = 9L / d ,欧拉定理：a ^ φ（t） = 1 (mod n), gcd(a,n) = 1 ----> 10 和 c 不是互质的一定无解，是互质的一定有解 ，欧拉函数只能说有解，但不一定是最小解，结论是：x | φ(t)，即是最小解一定是 φ(t) 的一个约数，

#include<bits/stdc++.h>  
using namespace std;  
#define \_ ios::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
#define endl '\n'  
#define ll long long   
ll qmul(ll a,ll k,ll b) //或者使用 \_\_int128  
{  
 ll res = 0;  
 while(k)  
 {  
 if(k & 1) res = (res + a) % b;  
 a = (a + a) % b;  
 k >>= 1;  
 }  
 return res ;  
}  
ll qmi(ll a,ll k,ll b)//这题 L 的范围太大，可能会爆 ll ,len(L) = 9,如果数十分大，可能爆 18 ll  
{  
 ll res = 1;  
 while(k)  
 {  
 if(k & 1)  
 {  
 res = qmul(res,a,b);  
 //res = res \* a % b;  
 }  
 //a = a \* a % b;  
 a = qmul(a,a,b);  
 k >>= 1;  
   
 }  
 return res;  
}  
ll get\_euler(ll C)  
{  
 ll res = C;  
 for(ll i=2;i<=C/i;i++)  
 {  
 if(C % i == 0)  
 {  
 while(C % i == 0) C /= i;  
 res = res \* (i - 1) / i;//1 - 1 / i 欧拉函数  
 }  
 }  
 if(C > 1) res = res / C \* (C - 1);  
 return res;  
  
}  
int main()  
{  
 \_;  
 int t = 1;  
 ll L;  
 while(cin >> L ,L)  
 {  
 int d = 1;  
 while(L % (d \* 2) == 0 && d \* 2 <= 8 ) d \*= 2;//不必写gcd，因为求得是L和8，很小没必要只是求有多少个 2 即可。  
 ll C = 9 \* L / d;  
 ll phi = get\_euler(C);  
 if(C % 2 == 0 || C % 5 == 0)//代表10 和 C 不是互质的无解  
 {  
 cout << "Case " << t ++ << ":" << 0 << endl;  
 continue;  
 }  
 //枚举φ(c)的所有约数  
 ll res = 1e18;  
 for(ll d=1;d\*d<=phi;d++)  
 {  
 if(phi % d == 0)  
 { //注意这样枚举是只有小的那一个约数是 d ，大的那一个要写一下  
 if(qmi(10,d,C) == 1) res = min(res,d);  
 if(qmi(10,phi/d,C) == 1) res = min(res,phi / d);//大的  
 }  
 }  
 cout << "Case " << t ++ << ":" << res << endl;  
 }  
}//8 11 16 0 ----> Case 1:1 Case 2:2 Case 3:0  
复制

# 中国剩余定理

/\*表达整数的奇怪方式 X 同余与 mi (mod ai) 求最小X\*/   
// x = k1\*a1+m1 k1 = k1 + k \* a2/d 通解，代入   
#include <iostream>   
using namespace std;   
#define ll long long   
   
ll exgcd(ll a,ll b,ll &x,ll&y)//x y加引用是因为后续要传回主函数中   
{   
 if(!b)   
 {   
 x = 1,y = 0;   
 return a;   
 }   
 ll d = exgcd(b,a % b,y,x);   
 y -= a / b \* x;   
 return d;   
}   
   
int main()   
{   
 int n;   
 cin >> n;   
 bool has\_answer = true;//判断是否有解   
 ll a1,m1;   
 cin >> a1 >> m1;   
 for(int i=0;i<n-1;i++)   
 {//不断求解 k1a1 - kiai = mi - m1的系数k1 ki   
 ll a2,m2;   
 cin >> a2 >> m2;   
 ll k1,k2;   
 ll d = exgcd(a1,a2,k1,k2);//exgcd求系数   
 if((m2 - m1) % d)//判断是否有解，斐蜀定理-有解一定是最大公约数的倍数   
 {//取余结果不是0证明不是倍数，无解 has\_answer = false;   
 break;   
 }   
 k1 \*= (m2-m1)/d;//此时求得解是为d是对应的解   
 ll t = a2 / d; //保证k变得足够小，最小正整数解 k = k1 + a2 / d 通解   
 k1 = (k1 % t + t) % t;//将k1变为最小的正整数解，防止溢出，所以变成最小的就是 % 一下   
 m1 = a1 \* k1 + m1; // x = a1\*k1+m1 + k\*[a1,a2]，[]是最小公倍数 -> x = x0 + ka   
 a1 = abs(a1 / d \* a2);//a1 a2的最小公倍数 //上式 x0即是更新后的m ，a即是更新后的a1   
 }//防止负数的干扰，输出符合条件的最小的正整数解   
 //最终化为一个式子 x = ka1 + m1 解即是 m1 % a1 + a1 ) % a1 if(has\_answer) cout << (m1 % a1 + a1) % a1 << endl;   
 else puts("-1"); }  
复制

# 扩展中国剩余定理

//差距在于大范围，会爆 long long  
#include <bits/stdc++.h>  
using namespace std;  
typedef \_\_int128 ll;  
const int N = 1e5 + 10;  
ll x, y, d; int n;  
  
void exgcd(ll &x, ll &y, ll a, ll b)   
{  
 if(!b) d = a, x = 1, y = 0;  
 else exgcd(y, x, b, a % b), y -= a / b \* x;  
}  
  
ll gcd(ll a, ll b)   
{  
 return b ? gcd(b, a % b) : a;  
}  
  
ll lcm(ll a, ll b)   
{  
 return a / gcd(a, b) \* b;  
}  
  
ll A1, b, A2, B;  
  
void merge() Pp+  
{  
 exgcd(x, y, A1, A2);  
 ll c = B - b;  
 if(c % d) puts("-1"), exit(0);  
 x = x \* c / d % (A2 / d);//x \* c / d是代表当前值的解，通解是x = x0 + a2 / d,所以最小解就是 %   
 if(x < 0) x += A2 / d;  
 ll mod = lcm(A1, A2);//下一步需要将A1变为A1和A2的最小公倍数  
 b = (A1 \* x + b) % mod; // x = a1\*x+b + k\*[a1,a2]，[]是最小公倍数 -> x = x0 + ka  
 if(b < 0) b += mod;  
 A1 = mod;//a1 a2的最小公倍数 //上式 x0即是更新后的m ，a即是更新后的a1  
   
}  
  
int main() {  
 cin >> n;  
 for(int i = 1 ; i <= n ; ++ i)   
 {  
 long long \_A, \_B;  
 cin>>\_A>>\_B, A2 = \_A, B = \_B;  
 if(i > 1) merge();  
 else A1 = A2, b = B;  
 }  
 printf("%lld\n", (long long)(b % A1));  
 return 0;  
}  
复制

# 组合计数

* 递推法
* 隔板法
* 加法原理乘法原理
* 组合数，排列数
* Lucas
* Catalan 数列

## 例题 1 1307 牡牛和牝牛

任意两个牡牛之间至少有 K 只牝牛。每种牛之间没有区别，答案对 5000011取模，求有多少种排队方法。

// 1307  
#include<bits/stdc++.h>  
using namespace std;  
#define endl '\n'  
#define ll long long——  
#define \_ ios::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
//f[i]表示的是所有长度是 i 的且以 1 结尾的字符串数量,上一个 1 的最大下标是 i-k-1 ，所以 f[i] = f[0] + f[1] + .. + f[i - k - 1],f[0] = 1;初值的设置递推一下边界值是否正确，此边界是 f[1],暴力算法是 n^2 使用前缀和优化   
int n,k;  
const int N = 100010,mod = 5000011;  
int f[N],s[N];  
  
int main()  
{  
 \_;  
 cin >> n >> k;  
 f[0] = s[0] = 1;  
 for(int i=1;i<=n;i++)  
 {  
 f[i] = s[max(i-k-1,0)];  
 s[i] = (s[i - 1] + f[i]) % mod;  
 }  
 cout << s[n] << endl;  
}// 4 2 ----> 6  
复制

## 例题 2 1308 方程的解

n = x ^ x % mod,求 a1 + a2 + ... + ak = n 正整数解的数量， 隔板法,将 n 个 1 放置，在之间加入 k-1 个隔板，分成 k 部分，每一部分为 ax ----> C(n-1)(k-1)无取模则需高精度,Cn m = C(n-1)(m-1) + C(n-1)(m)

#include<bits/stdc++.h>  
using namespace std;  
#define \_ ios::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
#define endl '\n'  
#define ll long long   
const int N = 150;//计算一下，大概139位  
int f[1000][100][N];  
int k,x;  
int qmi(int a,int b,int p)  
{  
 int res = 1;  
 while(b)  
 {  
 if(b & 1)  
 res = res \* a % p;  
 a = a \* a % p;  
 b >>= 1;  
 }  
 return res;  
}  
// 高精度  
void add(int c[],int a[],int b[])  
{  
 for(int i=0,t=0;i<N;i++)  
 {  
 t += a[i] + b[i];  
 c[i] = t % 10;  
 t /= 10;  
 }  
}  
int main()  
{  
 \_;  
 cin >> k >> x;  
  
 int n = qmi(x % 1000,x,1000);//防止越界直接先mod 1000  
 //C(n-1)(k-1)  
 for(int i=0;i<n;i++)  
 {  
 for(int j=0;j<=i&&j<k;j++)  
 {  
 if(!j)   
 f[i][j][0] = 1;  
 else add(f[i][j],f[i-1][j] , f[i-1][j-1]);//递推式fij=fi-1j+fi-1j-1  
 }  
 }  
 int \*g = f[n-1][k-1];  
 int i = N - 1;  
 while(!g[i]) i --;//不符合则减少  
 while(i >= 0)  
 cout << g[i --] << endl;   
} // 3 2 --- > 3  
复制

## 例题 3 1309 车的数量

车不能再同一行同一列，mod100003(如果mod是质数可以使用逆元来做)将不规则图形转换为规则图形 ans = Cnk *Pmk 乘法原理,枚举几部分放多少数量的车，先放小的图形，这样可以确定一定会对大的图像产生影响，如果先放大的可能分类，P 是排列数结果是阶乘例如P3 = 1*2\*3

#include<bits/stdc++.h>  
using namespace std;  
#define \_ ios::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
#define endl '\n'  
#define ll long long   
const int N = 2010,mod = 100003;  
int fact[N],infact[N];//阶乘和其逆元阶乘  
int qmi(int a,int k)  
{  
 int res = 1;  
 while(k)   
 {  
 if(k & 1)  
 res = (ll)res \* a % mod;  
 a = (ll) a \* a % mod;  
 k >>= 1;  
 }  
 return res;  
}  
int C(int a,int b)  
{//Cai = a! / ((a-i)!\*i!)  
 if(a < b) return 0;  
 return (ll)fact[a] \* infact[a-b] % mod \* infact[b] % mod;  
}  
int P(int a,int b)  
{//Pai = a! / (a-i)!  
 if(a < b) return 0;  
 return (ll)fact[a] \* infact[a-b] % mod;  
}  
int main()  
{  
 \_;  
 fact[0] = infact[0] = 1;  
 for(int i=1;i<N;i++)  
 {  
 fact[i] = (ll)fact[i-1] \* i % mod;  
 infact[i] = (ll)infact[i - 1] \* qmi(i,mod-2) % mod;//mod是质数由费马小定理，a^(p-1)=1(mod p),则a^p-2 \* a = 1(mod p)则逆元就是a^p-2,或者写成，qmi(fact[i],mod-2)=infact[i]  
 }  
 int a,b,c,d,k;//上面小矩形的长和宽，下面多出来的长和宽  
 cin >> a >> b >>c >> d >> k;  
 int res = 0;  
 for(int i=0;i<=k;i++)  
 {  
 res = (ll)(res + C(b,i) \* P(a,i) % mod \* C(d,k-i) % mod \* P(a + c - i,k -i)) % mod;  
 }  
 cout << res << endl;  
 return 0;  
}// 2 2 2 2 2 ---> 38  
复制

## 例题 4 1310 数三角形

一个格子，端点可以放置点，给一个大格子，选三个点这三个点不共线给出方案数，正难则反，先求出所有方案，减去共线即可，格点 n *m，斜率为 0 和无穷的容易得到，其他斜率>0的先选取一个点作为最下方的点，然后选取最上方的点，一共(n-i)* (m-j)种，然后找这两个点直接的整数点，斜率<0 的同理，选一点作为最上面的点，找下点，找中间点，中间数选法是(gcd(i,j)-1) *(n-i)* (m-j)(1是起点)

#include<bits/stdc++.h>  
using namespace std;  
#define \_ ios::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
#define endl '\n'  
#define ll long long  
int n,m;  
int gcd(int a,int b)  
{  
 return b ? gcd(b,a%b) : a;  
}  
ll C(int n)  
{  
 return (ll)(n)\*(n-1)\*(n-2)/6;  
}  
int main()  
{  
 \_;  
 cin >> n >> m;  
 n ++,m ++;  
 ll res = C(n \* m) - (ll)n \* C(m) - (ll)m \* C(n);  
 for(int i=1;i<=n;i++)  
 {  
 for(int j=1;j<=m;j++)  
 {//上下点是对称的结果  
 res -= 2ll \* (gcd(i,j) - 1) \* (n-i)\*(m-j);  
 }  
 }  
 cout << res << endl;  
}// 2 2 ---> 76  
复制

## 例题 5 1310 数三角形