数学板子

## 数论

### 素数

#### Miller-Rabin 素性测试

int miller\_robin(ll n){  
 if(n==2) return 1;  
 if(n%2==0||n<2) return 0;  
 ll m=n-1,q=0;  
 while(m%2==0) m/=2,q++;  
 for(int a:test\_int){  
 if(a>=n) break;  
 ll x=qpow(a,m,n);  
 for(int i=0;i<q;i++){  
 ll x1=x\*x%n;  
 if(x1==1&&x!=1&&x!=n-1) return 0;  
 x=x1;  
 }  
 if(x!=1) return 0;  
 }  
 return 1;  
}

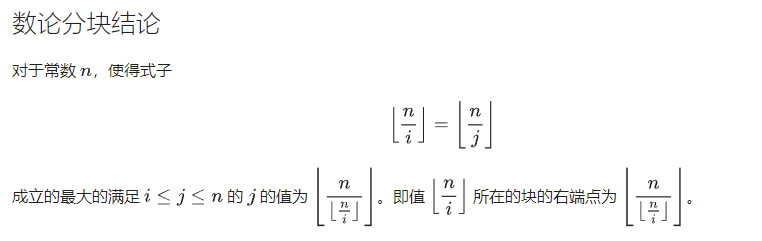
#### Pollard-rho

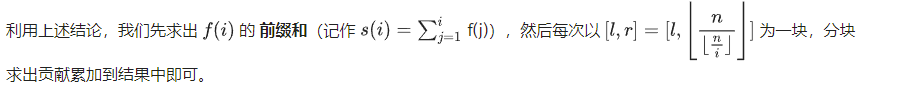
map<ll,ll> ans;  
ll pollard\_rho(ll x){  
 ll s=0,t=0,c=rd()%(x-1)+1,d=1;  
 for(ll val,step,g=1;d==1;g<<=1,s=t){  
 val=1;  
 for(step=1;d==1&&step<=g;step++){  
 t=((lll)t\*t+c)%x;  
 val=(lll)val\*abs(t-s)%x;  
 if(step%100==0) d=\_\_gcd(val,x);  
 }  
 d=\_\_gcd(val,x);  
 }  
 return d;  
}  
void fac(ll n){  
 if(n<2) return;  
 if(miller\_robin(n)){  
 ans[n]++;return;  
 }  
 ll p=n;  
 while(p>=n) p=pollard\_rho(n);  
 fac(n/p),fac(p);  
}

#### exgcd

LL exgcd(LL a,LL b,LL &x,LL &y)//扩展欧几里得算法   
{  
 if(b==0)  
 {  
 x=1,y=0;  
 return a;  
 }  
 LL ret=exgcd(b,a%b,y,x);  
 y-=a/b\*x;  
 return ret;  
}  
LL getInv(int a,int mod)//求a在mod下的逆元，不存在逆元返回-1   
{  
 LL x,y;  
 LL d=exgcd(a,mod,x,y);  
 return d==1?(x%mod+mod)%mod:-1;  
}

#### 数论分块





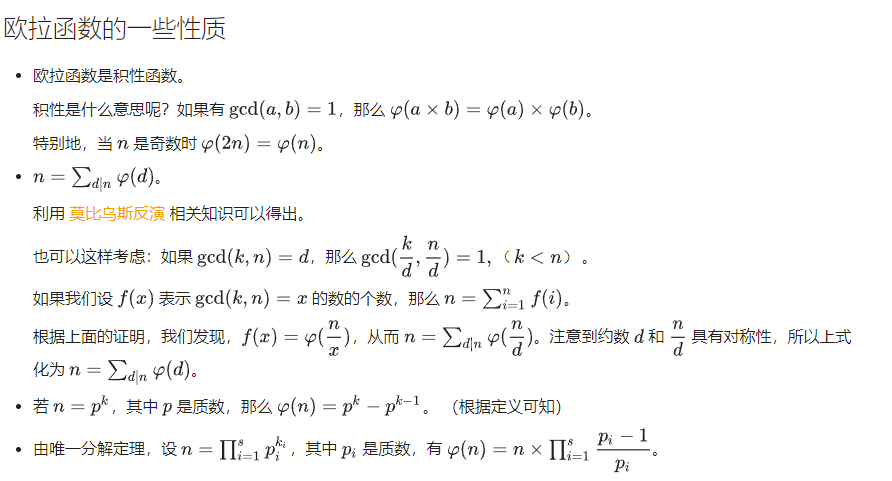
long long H(int n) {  
 long long res = 0; // 储存结果  
 int l = 1, r; // 块左端点与右端点  
 while (l <= n) {  
 r = n / (n / l); // 计算当前块的右端点  
 res += (r - l + 1) \* 1LL \*  
 (n / l); // 累加这一块的贡献到结果中。乘上 1LL 防止溢出  
 l = r + 1; // 左端点移到下一块  
 }  
 return res;  
}

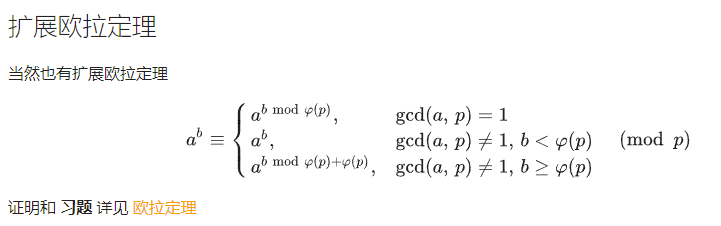
### 欧拉函数与筛法

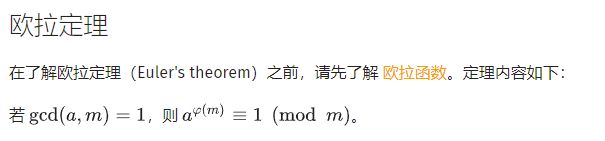
#### 筛法

int T;ll n;  
vector<ll> prime;  
bool pvis[maxn]={};  
void eulershai() {  
 const int n=maxn-5;  
 rep(i,2,n) {  
 if (pvis[i]==0) prime.push\_back(i);  
 for (ll j:prime) {  
 if (i\*j>n) break;  
 pvis[i\*j]=1;  
 if (i%j==0) break;  
 }  
 }  
}

#### 欧拉函数相关







#### 乘法逆元的限制

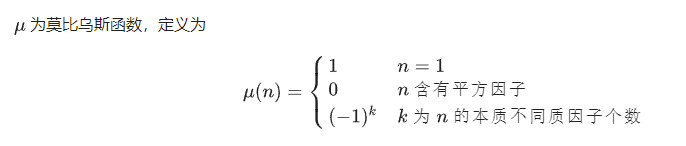


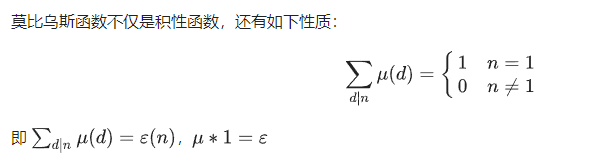
### BSGS

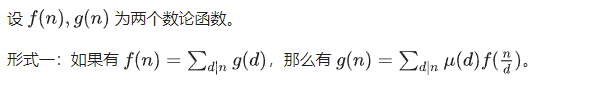
ll log\_mod(ll a,ll b,ll mod) {  
 map<ll,ll> mp;  
 ll m=sqrt(mod+0.5),p=1,x=1;  
 if(b==1) return 0;  
 for(int i=0;i<m;i++) mp[b\*p%mod]=i,p=(p\*a)%mod;  
 for(ll i=m;i<=mod;i+=m){  
 x=(x\*p)%mod;  
 if(mp.count(x)) return i-mp[x];  
 }  
 return -1;  
}

### 莫比乌斯反演

#### 莫比乌斯函数与反演结论









### 杜教筛

#include<map>  
using namespace std;  
#define ll long long   
ll BSGS(ll y,ll z,ll p) {  
 map<ll,ll> ma;  
 ll m=sqrt(p),tmp=0;  
 if(y%p==0&&z==0) return 1;  
 if(y%p==0&&z!=0) return -1;  
 for(int i=0;i<=m;i++) {  
 if(!i) {tmp=z%p;ma[tmp]=i;continue;}  
 tmp=(tmp\*y)%p;  
 ma[tmp]=i;  
 }  
 tmp=1;ll t= power(y,m,p);//快速幂  
 for(int i=1;i\*i<=p;i++) {  
 tmp=(tmp\*t)%p;  
 if(ma[tmp]) {  
 ll ans=i\*m-ma[tmp];  
 return ans;  
 }  
 }  
 return -1;  
}

## 多项式

### FFT && MTT

#define rep(i,a,b) for(int i=a;i<(int)b;i++)  
typedef long long ll;  
const double pi=acos(-1);  
typedef complex<double> C;  
void fft(vector<C>&a,int n,int ty){//n==a.size()  
 vector<C> w(n);w[0].real(1);  
 for(int i=0,j=0;i<n;i++){  
 if(i>j) swap(a[i],a[j]);  
 for(int l=n/2;(j^=l)<l;l/=2);  
 }  
 for(int i=1,j,k;i<n;i\*=2){  
 C t(cos(pi/i),ty\*sin(pi/i));  
 for(j=i-2;j>=0;j-=2) w[j+1]=(w[j]=w[j/2])\*t;  
 for(j=0;j<n;j+=i\*2)  
 for(k=j;k<j+i;k++){  
 C y=a[k+i]\*w[k-j];  
 a[k+i]=a[k]-y;a[k]+=y;  
 }  
 }  
 if(ty!=1) for(int i=0;i<n;i++) a[i]/=n;  
}  
int mod;  
template<class ty>//precision up to 10^23  
void operator\*=(vector<ty>&a,vector<ty>&b){  
 #define x real  
 #define y imag  
 int n=a.size(),m=b.size(),  
 len=1<<int(ceil(log2(n+m)));  
 vector<C> c(len),d(len),e(len);  
 rep(i,0,n) c[i]=C(a[i]>>15,a[i]&0x7fff);  
 rep(i,0,m) d[i]=C(b[i]>>15,b[i]&0x7fff);  
 fft(c,len,1),fft(d,len,1);  
 rep(i,0,len){  
 int j=(len-i)&(len-1);  
 e[i]=d[i]\*C((c[i].x()+c[j].x())/2,(c[i].y()-c[j].y())/2);  
 d[i]=d[i]\*C((c[i].y()+c[j].y())/2,(c[j].x()-c[i].x())/2);  
 }  
 fft(d,len,-1),fft(e,len,-1);  
 a.resize(len=n+m-1);  
 rep(i,0,len){  
 ll x=e[i].x()+0.5,y=e[i].y()+0.5,  
 z=d[i].x()+0.5,w=d[i].y()+0.5;  
 ll t=(x%mod<<30)+((y+z)%mod<<15)+w;  
 a[i]=(t%mod+mod)%mod;  
 }  
}  
template<class ty>//faster  
vector<ty> operator\*(vector<ty>a,vector<ty>b){  
 int n=a.size(),m=b.size(),  
 len=1<<int(ceil(log2(n+m)));  
 vector<C> c(len),d(len);  
 rep(i,0,n) c[i]=a[i];  
 rep(i,0,m) d[i]=b[i];  
 fft(c,len,1),fft(d,len,1);  
 rep(i,0,len) c[i]\*=d[i];  
 fft(c,len,-1);  
 a.resize(len=n+m-1);  
 rep(i,0,len) a[i]=c[i].real()+0.5;  
 return a;  
}  
int main(){  
 int n,m;  
 scanf("%d%d%d",&n,&m,&mod);  
 vector<int> a(n+1),b(m+1);  
 for(int&i:a) scanf("%d",&i);  
 for(int&i:b) scanf("%d",&i);  
 a\*=b;// a = a \* b是fft a \*= b 是mtt  
 for(int i:a) printf("%d ",i);  
}

### NTT

求一个原根

是φ(m)的所有不同素因数，则g是m的原根，当且仅当对任意,都有

如果是原根,群的阶次方才为1,如果不是原根,群的阶的约数次方就会出现1

大质数表：

const int maxn=2e5+9;  
const ll mod=998244353,g=3;//mod=1004535809,469762049  
void ntt(ll a[],int n,int ty){  
 for(int i=0,j=0;i<n;i++){  
 if(i>j) swap(a[i],a[j]);  
 for(int l=n/2;(j^=l)<l;l/=2);  
 }  
 for(int i=1;i\*2<=n;i\*=2){  
 ll wi=qpow(g,(mod-1)/(2\*i));  
 if(ty==-1) wi=qpow(wi,mod-2);  
 for(int j=0;j<n;j+=2\*i){  
 ll w=1;  
 for(int k=j;k<j+i;k++){  
 ll u=a[k],v=w\*a[k+i]%mod;  
 a[k]=(u+v)%mod;  
 a[k+i]=(u-v+mod)%mod;  
 w=wi\*w%mod;  
 }  
 }  
 }  
 if(ty==-1) {  
 ll t=qpow(n,mod-2);  
 for(int i=0;i<n;i++) a[i]=a[i]\*t%mod;  
 }  
}

### FMT && FWT

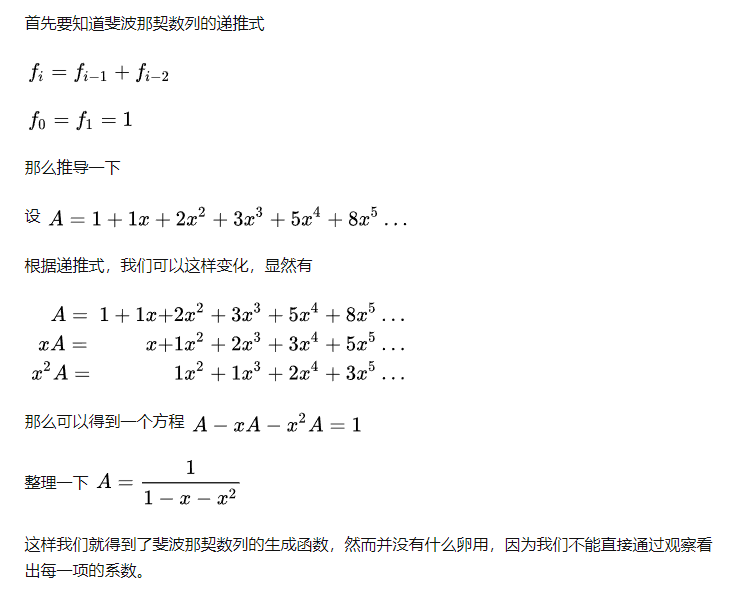
#define fwt\_loop for(int i=1,j,k;i<n;i\*=2)\  
 for(j=0;j<n;j+=2\*i) for(k=j;k<j+i;k++)   
void fwt\_or(ll a[],int n,ll x){  
 fwt\_loop a[i+k]=(a[i+k]+a[k]\*x)%mod;  
}  
void fwt\_and(ll a[],int n,ll x){  
 fwt\_loop a[k]=(a[k]+a[i+k]\*x)%mod;  
}  
void fwt\_xor(ll a[],int n,ll x){  
 fwt\_loop{  
 ll y=a[k],z=a[i+k];  
 a[k]=(y+z)\*x%mod;  
 a[i+k]=(y+mod-z)\*x%mod;  
 }  
}

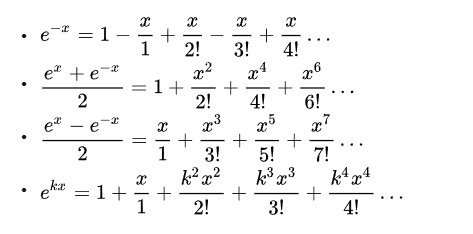
## 生成函数

### 经典生成函数公式

$\Large{\frac{1}{1 - x}= \sum\_{i = 0}^{\infty}}x^i$ $\Large{e^x = \sum\_{i = 0}^{\infty}\frac{x^i}{i!}}$

### fib的生成函数推导过程





## 线性代数

### 高斯消元

// n 为未知数个数，m 为方程个数，返回方程组的解  
//(多解/无解返回一个空的 vector)  
// mat[1~n]:增广矩阵,每行! 0 !位置为常数  
typedef bitset<2009> mat[2009];  
vector<int> gauss(mat a,int n,int m){  
 int l=1;  
 for(int c=1;c<=n;c++,l++) {  
 int p=l;  
 for(int i=l;i<=m;i++) if(a[i][c]) p=i;  
 if(!a[p][c]) {l--;continue;}  
 if(p!=l) swap(a[p],a[l]);  
 for(int i=1;i<=m;i++) if(l!=i&&a[i][c])  
 a[i]^=a[l];  
 }//如果无解或多解，返回一个自由元个数的负数  
 if(l<=n) return vector<int>(1,l-n-1);  
 vector<int> ans(n+1,0);  
 for(int i=1;i<=n;i++) ans[i] = a[i].test(0);  
 return ans;//否则返回一个01矩阵  
}

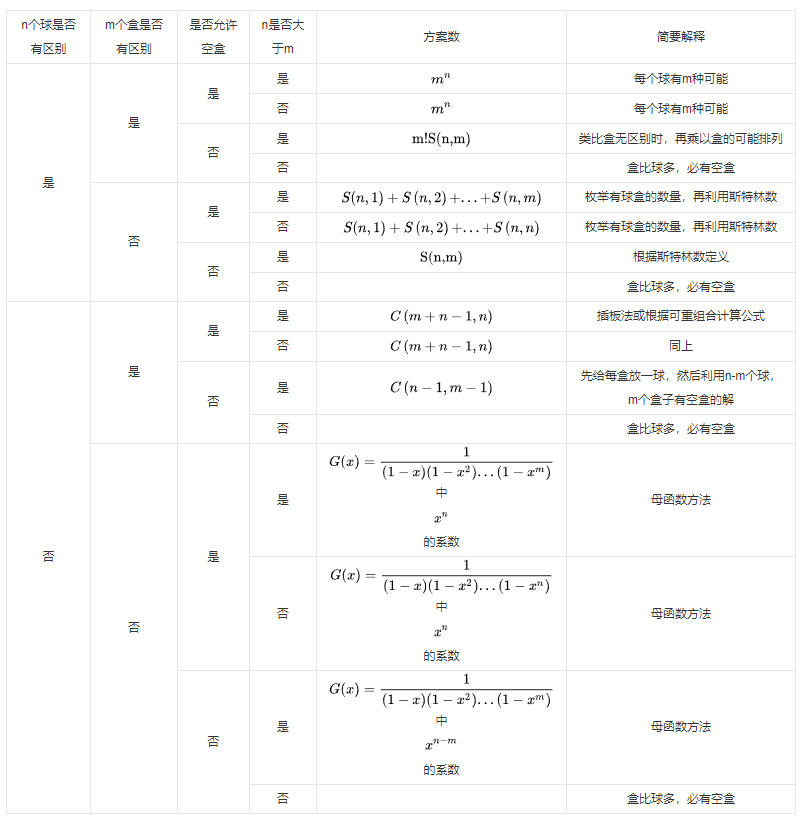
### 线性基插入

给定n个整数（数字可能重复），求在这些数中选取任意个，使得他们的异或和最大。

ll s[66],a[66];  
int n,sz;  
void insert(ll x){  
 rep(i,0,62)if(x>>i&1){  
 if(s[i]) x^=s[i];  
 else {  
 swap(s[i],x);  
 sz++;break;  
 }  
 }  
}

## 组合数学

### 球桶模型



namespace binomial{  
 const int mmm=2e6+9;  
 ll fac[mmm],ivf[mmm];  
 ll qpow(ll x,ll n,ll c=1){  
 for(;n;n/=2,x=x\*x%mod)if(n&1)c=c\*x%mod;  
 return c;  
 }  
 void init(){  
 fac[0]=1;  
 for(int i=1;i<mmm;i++) fac[i]=fac[i-1]\*i%mod;  
 ivf[mmm-1]=qpow(fac[mmm-1],mod-2);  
 for(int i=mmm-1;i>0;i--) ivf[i-1]=ivf[i]\*i%mod;  
 }  
 ll C(int n,int m){  
 if(!fac[0]) init();  
 if(n<0||m<0||n<m) return 0;  
 return fac[n]\*ivf[m]%mod\*ivf[n-m]%mod;  
 }  
 ll catalan(int n){return C(2\*n,n)-C(2\*n,n-1);}  
}

### 错位排列

$\huge{ f(n) = (n - 1)(f (n -1) + f(n - 2))}$

### 卡特兰数

1. 有 个人排成一行进入剧场。入场费 5 元。其中只有 个人有一张 5 元钞票，另外 人只有 10 元钞票，剧院无其它钞票，问有多少种方法使得只要有 10 元的人买票，售票处就有 5 元的钞票找零？
2. 一位大城市的律师在她住所以北 个街区和以东 个街区处工作。每天她走 个街区去上班。如果他从不穿越（但可以碰到）从家到办公室的对角线，那么有多少条可能的道路？
3. 在圆上选择 个点，将这些点成对连接起来使得所得到的 条线段不相交的方法数？
4. 对角线不相交的情况下，将一个凸多边形区域分成三角形区域的方法数？
5. 一个栈（无穷大）的进栈序列为 有多少个不同的出栈序列？
6. 个结点可构造多少个不同的二叉树？

#### 常见卡特兰递推式与应用

$\LARGE{ H\_n = \frac{C\_{2n}^{n}}{n + 1}}$ $\LARGE{H\_n = C\_{2n}^{n} - C\_{2n}^{n - 1}}$

#### 非降路径问题

* 方案一 、 到 的非降路径方案是
* 方案二、 从到的除端点外不接触直线的的非降路径的非降路径数
* 方案三、 从到的除端点外不穿过直线的的非降路径数

## 博弈

### Nim 博弈

有若干堆石子，每堆石子的数量都是有限的，合法的移动是“选择一堆石子并拿走若干颗（不能不拿）”，拿走最后一颗石子的人赢。

先手必胜当且仅当

将局势从N 转 P 的方法，异或一次即可

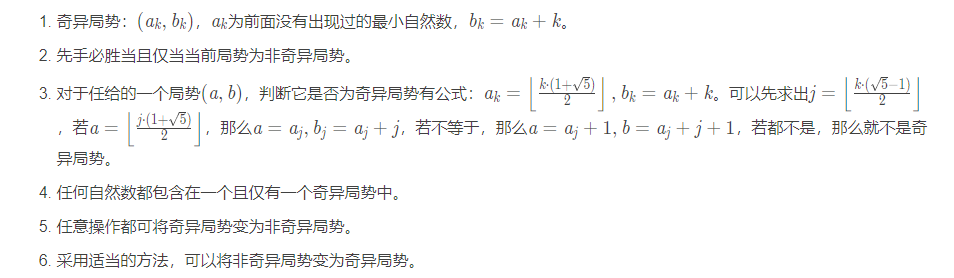
### 巴什博弈

有若干堆石子，每堆石子的数量都是有限的，合法的移动是“选择一堆石子并拿走1 − m颗”，拿走最后一颗石子的人赢。

令 吸纳收必胜当且仅当

### 威佐夫博弈

有两堆各若干个物品，两人轮流从任意一堆中取至少一个或同时从两堆中取同样多的物品，规定每次至少去一个，多者不限，最后取光者得胜。



### anti-Nim

有若干堆石子，每堆石子的数量都是有限的，合法的移动是“选择一堆石子并拿走若干颗（不能不拿）”，拿走最后一颗石子的人输。

先手必胜当且仅当：

1. 所有堆的石子数量均不大于，且游戏的**SG** 值为**0**。
2. 存在石子数量大于的堆，并且游戏的**SG** 值不为**0**。

### Every-Nim

 Every-SG游戏规定，对于所有还没有结束的子游戏，游戏者必须对该子游戏进行操作。除此之外，其他规则与普通SG游戏相同。

**结论**：  
对于SG值为0的点，我们需要知道最快几步能将游戏带入终止状态；对于SG值不为0的点，需要知道最慢几步会被带入终止状态。  
我们用step 来表示这个步数，

先手胜当且仅当最大步数为奇数。

### Moore Nimk

有n堆石子，每次至少选1堆，最多选m堆，每堆都可以拿任意正数个石子。

**结论**：将每堆石子的数量写为二进制形式，然后将每一位上的1数量相加后模，若每一位上的的数量经过计算后都为，则为必败态。（Nim游戏即为时的特殊情况）。

### 阶梯 Nim

 在阶梯的每一层上有若干个石子，每次可以选择任意层的任意个石子将其移动到下一层，最后不能移动的人输。

**结论**：对所有奇数阶梯的石子数量做异或运算，若结果为0 00则为必败态。

### Multi-SG

在符合规则的前提下，一个单一的子游戏的后继可以为多个子游戏。其它规则与SG游戏相同。