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

数论分块结论

对于常数 n，使得式子

$$\left\lfloor \frac{n}{i} \right\rfloor = \left\lfloor \frac{n}{j} \right\rfloor$$

成立的最大的满足 $i \leq j \leq n$ 的 j 的值为 $\left\lfloor \frac{n}{\lfloor \frac{n}{i} \rfloor} \right\rfloor$。即值 $\left\lfloor \frac{n}{i} \right\rfloor$ 所在的块的右端点为 $\left\lfloor \frac{n}{\lfloor \frac{n}{i} \rfloor} \right\rfloor$。

利用上述结论，我们先求出 $f(i)$ 的 **前缀和**（记作 $s(i) = \sum_{j=1}^{i}$ f(j)），然后每次以 $[l, r] = [l, \left\lfloor \frac{n}{\lfloor \frac{n}{l} \rfloor} \right\rfloor]$ 为一块，分块求出贡献累加到结果中即可。

```
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

欧拉函数的一些性质

- 欧拉函数是积性函数。
  积性是什么意思呢？如果有 $\gcd(a,b)=1$，那么 $\varphi(a\times b)=\varphi(a)\times\varphi(b)$。
  特别地，当 n 是奇数时 $\varphi(2n)=\varphi(n)$。
- $n=\sum_{d\mid n}\varphi(d)$。
  利用 莫比乌斯反演 相关知识可以得出。
  也可以这样考虑：如果 $\gcd(k,n)=d$，那么 $\gcd(\frac{k}{d},\frac{n}{d})=1$，（$k<n$）。
  如果我们设 $f(x)$ 表示 $\gcd(k,n)=x$ 的数的个数，那么 $n=\sum_{i=1}^{n}f(i)$。
  根据上面的证明，我们发现，$f(x)=\varphi(\frac{n}{x})$，从而 $n=\sum_{d\mid n}\varphi(\frac{n}{d})$。注意到约数 d 和 $\frac{n}{d}$ 具有对称性，所以上式化为 $n=\sum_{d\mid n}\varphi(d)$。
- 若 $n=p^k$，其中 p 是质数，那么 $\varphi(n)=p^k-p^{k-1}$。（根据定义可知）
- 由唯一分解定理，设 $n=\prod_{i=1}^{s}p_i^{k_i}$，其中 p_i 是质数，有 $\varphi(n)=n\times\prod_{i=1}^{s}\frac{p_i-1}{p_i}$。

扩展欧拉定理

当然也有扩展欧拉定理

$$a^b\equiv\begin{cases}a^{b \bmod \varphi(p)}, & \gcd(a,\,p)=1\\ a^b, & \gcd(a,\,p)\ne1,\,b<\varphi(p)\\ a^{b \bmod \varphi(p)+\varphi(p)}, & \gcd(a,\,p)\ne1,\,b\ge\varphi(p)\end{cases} \pmod p$$

证明和 **习题** 详见 欧拉定理

欧拉定理

在了解欧拉定理（Euler's theorem）之前，请先了解 欧拉函数。定理内容如下：

若 $\gcd(a,m)=1$，则 $a^{\varphi(m)}\equiv1\pmod m$。

乘法逆元的限制

注意使用 费马小定理 需要限制 b 是一个素数，而扩展欧几里得算法只要求 $\gcd(a,p)=1$。

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

μ 为莫比乌斯函数，定义为

$$\mu(n)=\begin{cases}1 & n=1\\ 0 & n\text{ 含有平方因子}\\ (-1)^k & k\text{ 为 }n\text{ 的本质不同质因子个数}\end{cases}$$

莫比乌斯函数不仅是积性函数，还有如下性质：

$$\sum_{d|n}\mu(d)=\begin{cases}1 & n=1\\ 0 & n\neq 1\end{cases}$$

即 $\sum_{d|n}\mu(d)=\varepsilon(n)$，$\mu*1=\varepsilon$

设 $f(n),g(n)$ 为两个数论函数。

形式一：如果有 $f(n)=\sum_{d|n}g(d)$，那么有 $g(n)=\sum_{d|n}\mu(d)f(\frac{n}{d})$。

形式二：如果有 $f(n)=\sum_{n|d}g(d)$，那么有 $g(n)=\sum_{n|d}\mu(\frac{d}{n})f(d)$。

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

$gcd(g,m)=1,$ 设$p_1,p_2,\dots p_k$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 $1\leq i\leq k,$都有$g^{(}\varphi(m)/p_i)\neq 1\pmod{m}$

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：$1e9+97,1e9+93,1e9+103,1e14+31,1e16+61,1e18+3,1e18+9$

$31525197391593473=7*2^52+1,g=3$

$180143985094819841=5*2^55+1,g=6$

$1945555039024054273=27*2^56+1,g=5$

$4179340454199820289=29*2^57+1,g=3$

```
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

$\frac{1}{1-x} = \sum_{i=0}^{\infty} x^i$ $\qquad$ $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$

fib的生成函数推导过程

首先要知道斐波那契数列的递推式

$f_i = f_{i-1} + f_{i-2}$

$f_0 = f_1 = 1$

那么推导一下

设 $A = 1 + 1x + 2x^2 + 3x^3 + 5x^4 + 8x^5 \dots$

根据递推式，我们可以这样变化，显然有

$$\begin{aligned} A = &\ 1 + 1x + 2x^2 + 3x^3 + 5x^4 + 8x^5 \dots \\ xA = &\ \quad\ \ x + 1x^2 + 2x^3 + 3x^4 + 5x^5 \dots \\ x^2 A = &\ \qquad\quad 1x^2 + 1x^3 + 2x^4 + 3x^5 \dots \end{aligned}$$

那么可以得到一个方程 $A - xA - x^2 A = 1$

整理一下 $A = \dfrac{1}{1 - x - x^2}$

这样我们就得到了斐波那契数列的生成函数，然而并没有什么卵用，因为我们不能直接通过观察看出每一项的系数。

- $e^{-x} = 1 - \dfrac{x}{1} + \dfrac{x}{2!} - \dfrac{x}{3!} + \dfrac{x}{4!} \dots$
- $\dfrac{e^x + e^{-x}}{2} = 1 + \dfrac{x^2}{2!} + \dfrac{x^4}{4!} + \dfrac{x^6}{6!} \dots$
- $\dfrac{e^x - e^{-x}}{2} = \dfrac{x}{1} + \dfrac{x^3}{3!} + \dfrac{x^5}{5!} + \dfrac{x^7}{7!} \dots$
- $e^{kx} = 1 + \dfrac{x}{1} + \dfrac{k^2 x^2}{2!} + \dfrac{k^3 x^3}{3!} + \dfrac{k^4 x^4}{4!} \dots$

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

n个球是否有区别	m个盒是否有区别	是否允许空盒	n是否大于m	方案数	简要解释
是	是	是	是	m^n	每个球有m种可能
			否	m^n	每个球有m种可能
		否	是	m!S(n,m)	类比盒无区别时，再乘以盒的可能排列
			否		盒比球多，必有空盒
	否	是	是	$S(n,1)+S(n,2)+\ldots+S(n,m)$	枚举有球盒的数量，再利用斯特林数
			否	$S(n,1)+S(n,2)+\ldots+S(n,n)$	枚举有球盒的数量，再利用斯特林数
		否	是	S(n,m)	根据斯特林数定义
			否		盒比球多，必有空盒
否	是	是	是	$C(m+n-1,n)$	插板法或根据可重组合计算公式
			否	$C(m+n-1,n)$	同上
		否	是	$C(n-1,m-1)$	先给每盒放一球，然后利用n-m个球，m个盒子有空盒的解
			否		盒比球多，必有空盒
	否	是	是	$G(x)=\frac{1}{(1-x)(1-x^2)\ldots(1-x^m)}$ 中 x^n 的系数	母函数方法
			否	$G(x)=\frac{1}{(1-x)(1-x^2)\ldots(1-x^n)}$ 中 x^n 的系数	母函数方法
		否	是	$G(x)=\frac{1}{(1-x)(1-x^2)\ldots(1-x^m)}$ 中 x^{n-m} 的系数	母函数方法
			否		盒比球多，必有空盒

```cpp
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

$$f(n) = (n-1)(f(n-1)+f(n-2))$$

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. n 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

$$H_n = \frac{C_{2n}^n}{n+1} \quad H_n = C_{2n}^n - C_{2n}^{n-1}$$

非降路径问题

- 方案一 、 $(0，0)$到(m,n) 的非降路径方案是C_{n+m}^n
- 方案二、 从$(0,0)$到(n,n)的除端点外不接触直线的$y=x$的非降路径的非降路径数 $2C_{2n-2}^{n-1}-2C_{2n-2}^n$
- 方案三、 从$(0,0)$到(n,n)的除端点外不穿过直线的$y=x$的非降路径数$\frac{2}{n+1}C_{2n}^n$

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$nim = a_1 \bigoplus a_2 \bigoplus a_3 ... \bigoplus a_n \neq 0$

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 $b_i = a_i mod(m+1)$ 吸纳收必胜当且仅当$nim = b_1 \bigoplus b_2 \bigoplus b_3 ... \bigoplus b_n \neq 0$

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。
```

1. 奇异局势：(a_k, b_k)，a_k为前面没有出现过的最小自然数，$b_k = a_k + k$。
2. 先手必胜当且仅当当前局势为非奇异局势。
3. 对于任给的一个局势(a, b)，判断它是否为奇异局势有公式：$a_k = \left\lfloor \frac{k\cdot(1+\sqrt{5})}{2} \right\rfloor$，$b_k = a_k + k$。可以先求出$j = \left\lfloor \frac{k\cdot(\sqrt{5}-1)}{2} \right\rfloor$，若$a = \left\lfloor \frac{j\cdot(1+\sqrt{5})}{2} \right\rfloor$，那么$a = a_j, b_j = a_j + j$，若不等于，那么$a = a_j + 1, b = a_j + j + 1$，若都不是，那么就不是奇异局势。
4. 任何自然数都包含在一个且仅有一个奇异局势中。
5. 任意操作都可将奇异局势变为非奇异局势。
6. 采用适当的方法，可以将非奇异局势变为奇异局势。

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1，且游戏的**SG** 值为**0**。
2. 存在石子数量大于1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，

$$SG(x) = \begin{cases} 0[\text{终止状态}] \\ max(step(u)) + 1[SG(V) > 0 \wedge u\text{为}v\text{的后继状态} \wedge SG(U) = 0] \\ min(step(u)) + 1[SG(v) = 0 \wedge u\text{为}v\text{的后继状态}] \end{cases}$$

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$(m+1)$，若每一位上的1的数量经过计算后都为0，则为必败态。（Nim游戏即为$m = 1$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。
```

$$SG(x) = \begin{cases} x-1[x \mod 4=0] \\ x[x \mod 4=1 \vee 2] \\ x+1[x \mod 4=3] \end{cases}$$