ICPC 代码模板-V2.0 数学

SDU(QD)　sdu1801

2019 年 11 月 14 日

目录

1 数学

1.1 线性筛法

1.1.1 欧拉函数

```
typedef long long ll;
const ll N = 10000000;
bool is[N+10];
int prime[N+10], cnt;
int phi[N+10];
ll n, ans;
void init(){
    phi[1] = 1;
    is[1] = is[0] = 1;
    for(int i = 2; i <= N; i ++){
        if(!is[i]) prime[++cnt] = i, phi[i] = prime[cnt]-1;
        for(int j = 1; j <= cnt; j ++){
            if(i*prime[j] >= N) break;
            is[i*prime[j]] = 1;
            if(i % prime[j] == 0){
                phi[i*prime[j]] = phi[i] * prime[j];
                break;
            }else phi[i*prime[j]] = phi[i] * (prime[j] - 1);
        }
    }
}
```

1.1.2 莫比乌斯函数

```
typedef long long ll;
const ll N = 10000000;
bool is[N+10];
int prime[N+10], cnt;
int miu[N+10];
ll n, ans;
void init(){
    miu[1] = 1;
    is[1] = is[0] = 1;
    for(int i = 2; i <= N; i ++){
        if(!is[i]) prime[++cnt] = i, miu[i] = -1;
        for(int j = 1; j <= cnt; j ++){
            if(i*prime[j] >= N) break;
            is[i*prime[j]] = 1;
            if(i % prime[j] == 0){
                miu[i*prime[j]] = 0;
                break;
            }else miu[i*prime[j]] = - miu[i];
        }
    }
}
```

1.1.3 约数个数

```
//d 约数个数      num 最小素因子的个数
typedef long long ll;
const ll N = 10000000;
bool is[N+10];
int prime[N+10], cnt;
int d[N+10], num[N+10];
ll n, ans;
void init(){
    d[1] = 1,num[1] = 0;
    is[1] = is[0] = 1;
    for(int i = 2; i <= N; i ++){
        if(!is[i]) prime[++cnt] = i, d[i] = 2, num[i] = 1;
        for(int j = 1; j <= cnt; j ++){
            if(i*prime[j] >= N) break;
            is[i*prime[j]] = 1;
            if(i % prime[j] == 0){
                num[i*prime[j]] = num[i] + 1;
                d[i*prime[j]] = d[i] / (num[i] + 1) * (num[i*prime[j]]+1);
                break;
            }else d[i*prime[j]] = d[i] * d[prime[j]], num[i*prime[j]] = 1;
        }
    }
}
```

1.1.4 约数和

```
//sd 约数和  sp      表示从1到最小素因子的最大幂次的和
typedef long long ll;
const ll N = 10000000;
bool is[N+10];
int prime[N+10], cnt;
int sd[N+10], sp[N+10];
ll n, ans;
void init(){
    sp[1] = 0, sd[1] = 1;
    is[1] = is[0] = 1;
    for(int i = 2; i <= N; i ++){
        if(!is[i]) prime[++cnt] = i, sd[i] = i + 1, sp[i] = i + 1;
        for(int j = 1; j <= cnt; j ++){
            if(i*prime[j] >= N) break;
            is[i*prime[j]] = 1;
            if(i % prime[j] == 0){
                sp[i*prim[j]] = sp[i] * prim[j] + 1;
                sd[i*prim[j]] = sd[i] / sp[i] * sp[i*prim[j]];
                break;
            }else sd[i*prim[j]] = sd[i] * sd[prim[j]], sp[i*prim[j]] = 1 + prim[j];
        }
    }
}
```

1.2 求单个欧拉函数

```
typedef long long ll;
```

```
bool is[100010];
ll prime[100010], cnt;
ll n;
ll phi(ll x){
    ll ans = x, i;
    for(i = 1; prime[i] <= x && i <= cnt; i ++){
        if(x % prime[i] == 0) ans = ans/prime[i]*(prime[i]-1);
        while(x % prime[i] == 0) x /= prime[i];
    }
    if(x != 1) ans = ans/x*(x-1);
    return ans;
}
void init(){
    is[1] = is[0] = 1;
    for(int i = 2; i <= 100000; i ++){
        if(!is[i]) prime[++cnt] = i;
        for(int j = 1; j <= cnt; j ++){
            if(i*prime[j] >= 100000) break;
            is[i*prime[j]] = 1;
            if(i % prime[j] == 0) break;
        }
    }
}
```

1.3 拓展欧几里得（同余方程/逆元）

```
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 xx = x;
    x = y, y = xx - a/b * y;
    return GCD;
}
//时间复杂度：O(log n)
```

1.3.1 求乘法逆元

$$ax + by = gcd(a, b)$$

当 $gcd(a, b) = 1$ 时 (a,b 互质)，在 $\pmod b$ 的同余系下，有：

$$ax + by \equiv 1 \pmod b$$

$$ax \equiv 1 \pmod b$$

所以当 $gcd(a, b) = 1$ 时，a 存在 $\pmod b$ 意义下的乘法逆元，$a^{-1} = x$。

1.3.2 欧拉定理求逆元

若 n, a 为正整数，且 n, a 互质，则:

$$a^{\phi(n)} \equiv 1(mod\ n)$$

逆元即为：

$$a^{\phi(n)-1}$$

1.3.3 求解线性同余方程

线性同余方程是最基本的同余方程，“线性”表示方程的未知数次数是一次，即形如：

$$ax \equiv b \pmod{p}$$

此方程有解当且仅当 b 能够被 a 与 p 的最大公约数整除。这时，如果 x_0 是方程的一个解，那么所有的解可以表示为：

$$\{x_0 + k\frac{p}{d} | (k\ z)\}$$

其中 d 是 a 与 p 的最大公约数。在模 p 的完全剩余系 $0, 1, ..., p-1$ 中，恰有 d 个解。注意：算完记得判断答案是否合法，不合法用上式加到合法。假如 $gcd(a,p) = d$，我们可以把 a, b, p 同除以 d，使得 $gcd(a,p) = 1$。

$$ax \equiv b \pmod{p}$$

$$x \equiv ba^{-1} \pmod{p}$$

1.3.4 拓展欧几里得算法推导

不妨设 $a > b, A > B$ 因为 $ax + by = gcd(a,b)$，$gcd(a,b) = gcd(b, a\%b)$ 考虑递归的过程，令 $a = b\ b = a\%b$ 所以

$$bx' + (a\%b)y' = gcd(b, a\%b) = gcd(a,b) = ax + by$$

即：

$$bx' + (a\%b)y' = ax + by$$
$$bx' + (a - \lfloor\frac{a}{b}\rfloor b)y' = ax + by$$
$$bx' + ay' - \lfloor\frac{a}{b}\rfloor by' = ax + by$$
$$ay' + b(x' - \lfloor\frac{a}{b}\rfloor y') = ax + by$$

解得：

$$x = y' \quad y = x' - \lfloor\frac{a}{b}\rfloor y'$$

1.4 高斯消元

```
1  inline int dcmp(double x){
2      if(fabs(x) <= eps) return 0;
3      return x < 0 ? -1 : 1;
4  }
5
6  struct mr{
7      const static int sz = 501;
8      double d[sz][sz];
9      int sx, sy;
10     mr(int x, int y){memset(d, 0, sizeof(d)), sx = x, sy = y;}
11     void init(int *a, int x, int y){
12         if(x > sz || y > sz) exit(-1);
13         else sx = x, sy = y;
14         for(int i = 0; i < x*y; i ++) d[i/x][i%y] = a[i];
15     }
16     bool Gaussian(){
17         for(int i = 0; i < sx; i ++){
18             if(dcmp(d[i][i]) == 0){
19                 int j = i+1;
20                 while(j < sx && dcmp(d[j][i]) == 0) j ++;
21                 if(j >= sx) return false;
22                 else swap(d[i], d[j]);
23             }
24             for(int j = i+1; j < sx; j ++){
```

```
                double v = d[j][i]/d[i][i];
                for(int k = i; k < sy; k ++) d[j][k] -= v * d[i][k];
            }
        }
        for(int i = sx-1; i >= 0; i --){
            if(d[i][i] == 0) return false;
            d[i][sy-1] /= d[i][i], d[i][i] = 1;
            for(int j = i-1; j >= 0; j --) d[j][sy-1] -= d[j][i] * d[i][sy-1], d[j][i]
                = 0;
        }
        return true;
    }
};
```

1.5 中国剩余定理（拓展）

1.5.1 中国剩余定理

中国剩余定理可以求出以下的一元线性同余方程组中 x 的一个解：

$$(S):\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \quad \vdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

假设整数 $m_1, m_2, ..., m_n$ 两两互质，则对任意的整数：$a_1, a_2, ..., a_n$，方程组 (S) 有解，并且通解可以用如下方式构造得到：设 $M = \prod_{i=1}^{n} m_i$，并设 $M_i = \frac{M}{m_i}$，$t_i = M_i^{-1} \pmod{m}_i$ 即：

$$x \equiv \sum_{i=1}^{n} a_i t_i M_i \pmod{M}$$

1.5.2 拓展中国剩余定理

当 $gcd(m_1, m_2) \neq 1$ 时，考虑合并两个方程：

$$x \equiv c_1 \pmod{m_1}$$

$$x \equiv c_2 \pmod{m_2}$$

显然有 $x = c_1 + k_1 m_1$, $x = c_2 + k_2 m_2$, 即 $c_1 + k_1 m_1 = c_2 + k_2 m_2$ 移项得 $k_1 m_1 = k_2 m_2 + c_2 - c_1$ 我们令 $t = gcd(m_1, m_2)$, 则有 $\frac{m_1}{t} k_1 = \frac{m_2}{t} k_2 + \frac{c_2 - c_1}{t}$ 当且仅当 $t | c_2 - c_1$ 时可以合并此时有

$$\frac{m_1}{t} k_1 \equiv \frac{c_2 - c_1}{t} (mod\ \frac{m_2}{t})$$

使用扩展欧几里得求得 k_1 的通解：

$$k_1 = ans + k \times \frac{m_2}{t}\ (k \in \mathbb{Z})$$

将 k_1 代回第一个方程得:

$$x \equiv ans \times m_1 + k \times \frac{m_1 \times m_2}{t} + c_1$$

最终再变换成同余方程的形式：

$$x \equiv ans \times m_1 + c_1 (mod\ \frac{m_1 \times m_2}{t})$$

这是一个同余方程的形式，如此不断合并所有方程即可。

```
#include<iostream>
#include<cstdio>
#define LL long long
using namespace std;
const LL MAXN = 1e6 + 10;
LL K, C[MAXN], M[MAXN], x, y;
```

```
inline LL ksc(unsigned LL x,unsigned LL y, LL p){return ((LL)(x*y-(unsigned LL)((long double)x/p*y)*p)%p+p)%p;}
LL gcd(LL a, LL b) {
    return b == 0 ? a : gcd(b, a % b);
}
LL exgcd(LL a, LL b, LL &x, LL &y) {
    if (b == 0) {x = 1, y = 0; return a;}
    LL r = exgcd(b, a % b, x, y), tmp;
    tmp = x; x = y; y = tmp - (a / b) * y;
    return r;
}
LL inv(LL a, LL b) {
    LL r = exgcd(a, b, x, y);
    while (x < 0) x += b;
    return x;
}
//x%m=c
//最后X%M[n]=C[n]
int main() {
    cin>>K;
    for (LL i = 1; i <= K; i++) scanf("%lld%lld", &M[i], &C[i]);
    bool flag = 1;
    for (LL i = 2; i <= K; i++) {
        LL M1 = M[i - 1], M2 = M[i], C2 = C[i], C1 = C[i - 1], T = gcd(M1, M2);
        if ((C2 - C1) % T != 0) {flag = 0; break;}
        M[i] = (M1 /T * M2);
        C[i] = ksc(inv( M1 / T , M2 / T ),((C2 - C1) / T%(M2 / T)+(M2 / T))%(M2 / T),
            (M2 / T))  * M1 + C1;
        C[i] = (C[i] % M[i] + M[i]) % M[i];
    }
    printf("%lld\n", flag ? C[K] : -1);
    return 0;
}
```

1.6 FFT

```
struct Com{
    double x, y;
    Com(double _x = 0, double _y = 0){x = _x, y = _y;}
    Com operator + (const Com &t)const{return Com(x+t.x, y+t.y);}
    Com operator - (const Com &t)const{return Com(x-t.x, y-t.y);}
    Com operator * (const Com &t)const{return Com(x*t.x-y*t.y, x*t.y+y*t.x);}
};
const int maxn = 300010;
const double pi = acos(-1);
int BIT, R[maxn];
void fft(Com *a, int n, int inv){
    if(R[n-1] != n-1) // 分治FFT把这个if去掉
        for(int i = 0; i < n; i ++) R[i] = (R[i>>1]>>1) | ((i&1)<<(BIT-1));
    for(int i = 0; i < n; i ++) if(i < R[i]) std::swap(a[i], a[R[i]]);
    for(int i = 1; i < n; i <<= 1){
        Com mi(cos(pi/i), sin(pi/i) * inv);
        for(int j = 0; j < n; j += (i<<1)){
            Com x(1, 0);
```

```
            for(int k = 0; k < i; k ++, x = x * mi){
                Com t1 = a[j+k], t2 = x * a[j+k+i];
                a[j+k] = t1 + t2, a[j+k+i] = t1 - t2;
            }
        }
    }
}
Com a[maxn], b[maxn];
char c1[maxn], c2[maxn];
int res[maxn];
int main(){
    int n, m;
    scanf("%s%s", c1, c2);
    n = strlen(c1)-1, m = strlen(c2)-1;
    for(int i = 0; i <= n; i ++) a[i].x = c1[i]-'0';
    for(int i = 0; i <= m; i ++) b[i].x = c2[i]-'0';
    m += n;
    for(n = 1; n <= m; n <<= 1, BIT ++);
    fft(a, n, 1), fft(b, n, 1);
    for(int i = 0; i < n; i ++) a[i] = a[i] * b[i];
    fft(a, n, -1);
    for(int i = 0; i <= m; i ++) res[m-i] = (long long)(a[i].x/n+0.5);
    for(int i = 0; i < m; i ++) res[i+1] += res[i] / 10, res[i] %= 10;
    while(res[m] >= 10) res[m+1] += res[m] / 10, res[m ++] %= 10;
    for(int i = m; i >= 0; i --) printf("%d", res[i]);
    puts("");
    return 0;
}
```

1.7 NTT

1.7.1 代码实现

```
// g 为原根, inv=1正向, inv=-1逆向
void ntt(ll *a, int n, int inv){
    if(R[n-1] != n-1) // 分治NTT把这个if去掉
        for(int i = 0; i < n; i ++) R[i] = (R[i>>1]>>1) | ((i&1) << (BIT-1));
    for(int i = 0; i < n; i ++) if(i < R[i]) swap(a[i], a[R[i]]);
    for(int i = 1; i < n; i <<= 1){
        ll mi = inv == 1 ? pow(g, (mod-1)/(2*i)) : ni(pow(g, (mod-1)/(2*i)));
        for(int j = 0; j < n; j += (i<<1)){
            ll x = 1;
            for(int k = 0; k < i; k ++, x = (x * mi) % mod){
                ll t1 = a[j+k], t2 = (x * a[j+k+i]) % mod;
                a[j+k] = (t1 + t2) % mod;
                a[j+k+i] = (t1 - t2) % mod;
            }
        }
    }
}
```

1.7.2 常用 NTT 模数

g 是 $(\mathrm{mod}\ r*2^k+1)$ 的原根

表 1: 常用 NTT 模数

素数	r	k	g
3	1	1	2
5	1	2	2
17	1	4	3
97	3	5	5
193	3	6	5
257	1	8	3
7681	15	9	17
12289	3	12	11
40961	5	13	3
65537	1	16	3
786433	3	18	10
5767169	11	19	3
7340033	7	20	3
23068673	11	21	3
104857601	25	22	3
167772161	5	25	3
469762049	7	26	3
1004535809	479	21	3
2013265921	15	27	31
2281701377	17	27	3
3221225473	3	30	5
75161927681	35	31	3
77309411329	9	33	7
206158430209	3	36	22
2061584302081	15	37	7
2748779069441	5	39	3
6597069766657	3	41	5
39582418599937	9	42	5
79164837199873	9	43	5
263882790666241	15	44	7
1231453023109121	35	45	3
1337006139375617	19	46	3
3799912185593857	27	47	5
4222124650659841	15	48	19
7881299347898369	7	50	6
31525197391593473	7	52	3
180143985094819841	5	55	6
1945555039024054273	27	56	5
4179340454199820289	29	57	3

1.8 FWT

1.8.1 代码实现

```
#include<iostream>
#define LL long long
using namespace std;
const int MOD=998244353;
const int inv2=(MOD+1)/2;//逆元
void FWT_or(LL *a,int N,int opt){//数组(from 0) 数组长度  1 or -1
    for(int i=1;i<N;i<<=1)
        for(int p=i<<1,j=0;j<N;j+=p)
            for(int k=0;k<i;++k)
                if(opt==1)a[i+j+k]=(a[j+k]+a[i+j+k])%MOD;
                else a[i+j+k]=(a[i+j+k]+MOD-a[j+k])%MOD;
}
void FWT_and(LL *a,int N,int opt){
    for(int i=1;i<N;i<<=1)
        for(int p=i<<1,j=0;j<N;j+=p)
            for(int k=0;k<i;++k)
                if(opt==1)a[j+k]=(a[j+k]+a[i+j+k])%MOD;
                else a[j+k]=(a[j+k]+MOD-a[i+j+k])%MOD;
}
void FWT_xor(LL *a,int N,int opt){
    for(int i=1;i<N;i<<=1)
        for(int p=i<<1,j=0;j<N;j+=p)
            for(int k=0;k<i;++k)
            {
                LL X=a[j+k],Y=a[i+j+k];
                a[j+k]=(X+Y)%MOD;a[i+j+k]=(X+MOD-Y)%MOD;
                if(opt==-1)a[j+k]=1ll*a[j+k]*inv2%MOD,a[i+j+k]=1ll*a[i+j+k]*inv2%MOD;
            }
}
LL A[(1<<17)|1],B[(1<<17)|1];
LL a[(1<<17)|1],b[(1<<17)|1];
int main(){
    int n;cin>>n;
    int N=(1<<n);
    for(int i=0;i<N;i++)scanf("%lld",&A[i]);
    for(int i=0;i<N;i++)scanf("%lld",&B[i]);

    for(int i=0;i<N;i++)a[i]=A[i],b[i]=B[i];
    FWT_or(a,N,1),FWT_or(b,N,1);
    for(int i=0;i<N;i++)a[i]=a[i]*b[i]%MOD;
    FWT_or(a,N,-1);
    for(int i=0;i<N;i++)printf("%lld%c",a[i]," \n"[i+1==N]);

    for(int i=0;i<N;i++)a[i]=A[i],b[i]=B[i];
    FWT_and(a,N,1),FWT_and(b,N,1);
    for(int i=0;i<N;i++)a[i]=a[i]*b[i]%MOD;
    FWT_and(a,N,-1);
    for(int i=0;i<N;i++)printf("%lld%c",a[i]," \n"[i+1==N]);

    for(int i=0;i<N;i++)a[i]=A[i],b[i]=B[i];
```

```
51      FWT_xor(a,N,1),FWT_xor(b,N,1);
52      for(int i=0;i<N;i++)a[i]=a[i]*b[i]%MOD;
53      FWT_xor(a,N,-1);
54      for(int i=0;i<N;i++)printf("%lld%c",a[i],"␣\n"[i+1==N]);
55      return 0;
56  }
```

1.9 博弈游戏

1.9.1 博弈问题综述

所讨论的博弈问题满足以下条件：

1. 玩家只有两个人，轮流做出决策，且操作公平。
2. 游戏的状态集有限，保证游戏在有限步后结束.
3. 博弈双方都使用最优决策。

一个游戏有两个状态，我们称为必胜态和必败态，他们的关系：

- 必胜态表示，从当前状态可以转移到一个必败态。
- 必败态表示，从当前状态无法转移到一个必败态。

也就是说，一个状态不是必胜态，就是必败态。

1.9.2 Bash 博弈（同余理论）

问题：一堆 n 个物品，两个人从轮流中取出 1 到 m 个，无法取者失败。

解法：当 $(m+1)|n$ 时，先手必败，其余情况，先手必胜。当剩余的物品数量为 $(m+1)$ 的整数倍时，为必败态，此时无论先手取多少个，后手只需要使两次取的物品数的总和是 $(m+1)$，下一个状态依然处于必败态。

1.9.3 Nim 博弈（异或理论）

问题：n 堆物品，每堆有 a_i 个，每次在某一堆中操作，取出至少 1 个物品，无法操作者失败。

解法：定义 $S=\oplus_{i=0}^{n}a_i$ 为 n 堆石子的异或和，则当 $S=0$ 时，先手必败，其余情况先手必胜。原理为对于一个异或和为 0 的状态，只要对其中一个数的任意一个二进制位进行改动，其异或和一定会发生变化，变为非 0 值。此时对于异或和非 0 的局面，后手一定可以找到现在异或和中非 0 的最高位，既然次位的异或和为 1，那么在所有的堆中, 至少有一堆该位二进制值为 1。我们选定此数进行减值，将该位改为 0（这时，后面的位无论是什么，都一定比原来小），低于该位的位数按照异或和进行构造，使新的异或和为 0，此时的下一个状态依然是必败态。

1.9.4 Wythoff 博弈

问题：有两堆石子，个数为 x,y（不妨设 $x\le y$），每次每个人可以从任意一堆石子中取任意多的石子或者从两堆石子中取同样多的石子，不能取得人输。

解法：当 $\lfloor\frac{(y-x)(\sqrt{5}+1)}{2}\rfloor=x$ 时先手必败，否则先手必胜。该题可以转化为：在一个仅有第一象限的棋盘上的 (x,y) 处有一个棋子，该棋子只能向两个坐标减小的方向移动。具体地，该棋子可以向平行于 x,y 轴的方向走任意多步，也可以向平行于 $y=x$ 的直线方向走任意多步，不能移动棋子的一方失败。通过在棋盘划出必胜位置线，剩余的位置即为必败位置。

1.9.5 Fibonacci 博弈

问题：一堆石子有 n 个，两人轮流取，先取者第 1 次可以取任意多个，但不能全部取完，以后每次取的石子数不能超过上次取子数的 2 倍，不能取者输。

解法：当 n 是斐波那契数的时候，先手必败，否则先手必胜。

1.9.6 SG 函数（异或理论扩展）

$$SG[x]=\text{mex}\{SG[y]\}$$

其中 y 是 x 所有的后继状态（通过操作改变当前状态为后继态），mex 是最小的没有出现的连续自然数。

游戏的和 游戏 $x_1, x_2, ..., x_n$ 的和为 x，且 $x = \{x_1, x_2, ..., x_n\}$。理解为 x 是由 $x_1, x_2, ..., x_n$ 构成的新游戏。

SG 定理 如果一个游戏的所有子游戏相互独立（即一个子游戏的操作不会影响其他的子游戏），那么

$$SG[x] = SG[x_1] \text{ xor } SG[x_2] \text{ xor } ... \text{ xor } SG[x_n]$$

文字表述为，一些游戏的和的 SG 函数等于各个游戏 SG 函数的异或和。

原理 **SG** 函数本质上是把博弈问题转化为 **Nim** 博弈进行求解，一个子游戏相当于一堆石子，SG 函数的值即为该堆石子的个数，整个游戏的 SG 值为所有子游戏的异或和，即 Nim 博弈中，将每堆石子个数异或起来。若当前局面为必败态（SG 值为 0），改变整个游戏中任意一个子游戏到其后继状态时，所有子游戏 SG 函数值的异或和必然发生变化，取次值二进制的最高位，所有子游戏中的 SG 函数值二进制中，至少有一个数的二进制该位为 1，找到次数所表示的子游戏。该子游戏的 SG 值，是对该子游戏所有后继状态的 SG 值取 mex，这就意味着，我们可以通过执行当前子游戏的某个后继状态，把当前子游戏的 SG 值变为小于该值的任意一个自然数。这相当于 Nim 博弈中取走一些石子做减法。通过这样的变化，后手可以把当前游戏的 SG 函数值再次变为 0，此时下一个状态依旧必败。

1.10 线性预处理逆元

设 P 为题目中的模数。

$$P = \lfloor \frac{P}{i} \rfloor \times i + P\%i$$

一下运算在 $\%P$ 的同余系下进行。

$$-P\%i = \lfloor \frac{P}{i} \rfloor \times i$$

$$-P\%i \times (i)^{-1} \times (P\%i)^{-1} = \lfloor \frac{P}{i} \rfloor \times i \times (i)^{-1} \times (P\%i)^{-1}$$

$$-(i)^{-1} = \lfloor \frac{P}{i} \rfloor \times (P\%i)^{-1}$$

$$(i)^{-1} = -\lfloor \frac{P}{i} \rfloor \times (P\%i)^{-1}$$

因为 $P\%i < i$ 这个数的逆元已经求得，所以借此公式就可以线性处理逆元了。

```
1  ni[i]=-(p/i)*ni[p%i];
```

1.11 Miller-Rabin 素数判定

```
1
2  #define ll long long
3  #define ull unsigned long long
4  #define ld long double
5  #define inl inline
6  using namespace std;
7  inl ll ksc(ull x, ull y, ll p) { return (x * y - (ull) ((ld) x / p * y) * p + p) % p;
       }
8  inl ll ksm(ll x, ll y, ll p) {
9      ll res = 1;
10     for (; y; y >>= 1, x = ksc(x, x, p))if (y & 1)res = ksc(res, x, p);
11     return res;
12 }
13 inl bool mr(ll x, ll p) {
14     if (ksm(x, p - 1, p) != 1)return 0;
15     ll y = p - 1, z;
16     while (!(y & 1)) {
17         y >>= 1;
18         z = ksm(x, y, p);
19         if (z != 1 && z != p - 1)return 0;
20         if (z == p - 1)return 1;
21     }
```

```
22      return 1;
23  }
24  int te_pri[20] = {0, 2, 3, 5, 7, 43, 61, 24251}, te_num = 7;
25  inl bool isprime(ll x) {
26      if (x < 2)return 0;
27      for (int i = 1; i <= te_num; i++)
28          if (x == te_pri[i])return 1;
29      for (int i = 1; i <= te_num; i++)
30          if (!(x % te_pri[i]) || !mr(te_pri[i], x))return 0;
31      return 1;
32  }
33
34  int main() {
35      return 0;
36  }
```

1.12 Pollard-Rho 算法 (大数质因数分解)

```
1
2  #define ll long long
3  #define ull unsigned long long
4  #define ld long double
5  #define inl inline
6  #define re register
7  using namespace std;
8
9  inl ll ksc(ull x, ull y, ll p) { return (x * y - (ull) ((ld) x / p * y) * p + p) % p;
       }
10 inl ll ksm(ll x, ll y, ll p) {
11     ll res = 1;
12     for (; y; y >>= 1, x = ksc(x, x, p))if (y & 1)res = ksc(res, x, p);
13     return res;
14 }
15 inl ll Abs(ll x) { return x < 0 ? -x : x; }
16 inl ll gcd(ll x, ll y) {
17     ll z;
18     while (y)  z = x, x = y, y = z % y;
19     return x;
20 }
21 inl bool mr(ll x, ll p) {
22     if (ksm(x, p - 1, p) != 1)return 0;
23     ll y = p - 1, z;
24     while (!(y & 1)) {
25         y >>= 1;
26         z = ksm(x, y, p);
27         if (z != 1 && z != p - 1)return 0;
28         if (z == p - 1)return 1;
29     }
30     return 1;
31 }
32 ll te_pri[20] = {0, 2, 3, 5, 7, 43, 61, 24251};
33 int te_num = 7;
34 inl bool isprime(ll x) {
35     if (x < 2)return 0;
```

```
    for (int i = 1; i <= te_num; i++)
        if (x == te_pri[i])return 1;
    for (int i = 1; i <= te_num; i++)
        if (!(x % te_pri[i]) || !mr(te_pri[i], x))return 0;
    return 1;
}
inl ll rho(ll p) {
    ll x, y, z, c, g;
    re int i, j;
    while (1) {
        y = x = rand() % p;
        z = 1, c = rand() % p;
        i = 0, j = 1;
        while (++i) {
            x = (ksc(x, x, p) + c) % p;
            z = ksc(z, Abs(y - x), p);
            if (x == y || !z)break;
            if (!(i % 127) || i == j) {
                g = gcd(z, p);
                if (g > 1)return g;
                if (i == j)y = x, j <<= 1;
            }
        }
    }
}
ll ys[1010];//最后的质数分解在ys中
int ind;//每次重新使用ind记得清零
inl void prho(ll p) {
    if (p == 1)return;
    if (isprime(p)) {
        ys[++ind] = p;
        return;
    }
    ll pi = rho(p);
    while (p % pi == 0)p /= pi;
    prho(pi);
    prho(p);
}
ll x, pos;
int main() {
    srand(time(0));
    cin >> x;
    prho(x);
    sort(ys + 1, ys + ind + 1);
    int m = unique(ys + 1, ys + ind + 1) - ys - 1;
    return 0;
}
```

1.13 组合数学相关

1.13.1 相关定理

定理 1 $\{1,2,...n\}$ 的 r 组合 $a_1,a_2,...,a_r$ 出现在所有 r 组合中的字典序位置编号：

$$index = C(n,r) - C(n-a_1,r) - C(n-a_2,r-1) - ... - C(n-a_r,1)$$

定理 2

$$k * C(n,k) = n * C(n-1,k-1)$$

$$C(n,0) + C(n,2) + ... = C(n,1) + C(n,3) + ...$$

$$1 * C(n,1) + 2 * C(n,2) + ... + n * C(n,n) = n * 2^{n-1}$$

1.13.2 错位排列

$$D_n = n! \sum_{k=0}^{n} \frac{(-1)^k}{k!}$$

1.13.3 Catalan 数

凸多边形三角划分,n 个节点组成二叉搜索树，n 对括号正确匹配数目，1-n 的出栈序列

$$C_n = \frac{C(2*n,n)}{n+1}$$

$$C_n = \frac{(4*n-2)}{n+1} * C_n - 1$$

$$C_1 = 1$$

1.13.4 Stirling 数

第一类 Stirling 数 s(p, k) 将 p 个不同元素构成 k 个圆排列的数目。

$$s(p,k) = (p-1) * s(p-1,k) + s(p-1,k-1)$$

第二类 Stirling 数 n 个元素拆分 k 个集合的方案数记为 S(n, k)。

$$S(p,k) = k * S(p-1,k) + S(p-1,k-1)$$

$$S(p,0) = 0, (p >= 1)$$

$$S(p,p) = 1, (p >= 0)$$

$$S(p,1) = 1, (p >= 1)$$

$$S(p,2) = 2^{p-1} - 1, (p >= 2)$$

$$S(p,p-1) = C(p,2)$$

1.13.5 Bell 数

元素个数为 n 的集合的划分数目。

$$B_p = \sum_{i=0}^{p} S(p,i)$$

其中 $S(p,i)$ 表示第二类 Stirling 数。

$$B_p = C(p-1,0) * B_0 + C(p-1,1) * B_1 + ... + C(p-1,p-1) * B_{p-1}$$

1.13.6 卢卡斯定理

若 p 是质数：

$$C(n,m)\%p = C(n/p, m/p) * C(n\%p, m\%p)\%p$$

```
1 #include<iostream>
2 #define LL long long
3 using namespace std;
4 LL n,m;
5 LL fac[100100],mod;
6 LL inv(LL i){
7     LL res=1,j=mod-2;
8     for(;j;j>>=1,i=i*i%mod)if(j&1)res=res*i%mod;
```

```
    return res;
}
LL C(LL a,LL b){return a>b?0:fac[b]*inv(fac[a]*fac[b-a]%mod)%mod;}
//b个里面选a个
LL lucas(LL a,LL b){return a?C(a%mod,b%mod)*lucas(a/mod,b/mod)%mod:1;}
int main(){
    int T;cin>>T;
    fac[0]=1;
    while(T--){
        cin>>n>>m>>mod;
        for(int i=1;i<mod;i++)fac[i]=fac[i-1]*i%mod;
        cout<<lucas(m,n+m)<<endl;
    }
    return 0;
}
```

若 p 不是质数，令 $p = p_1 * p_2 * ... * p_n$，这里 p_i 是质数

$$\begin{cases} C_n^m \equiv a_1 \pmod{p_1} \\ C_n^m \equiv a_2 \pmod{p_2} \\ \quad \vdots \\ C_n^m \equiv a_n \pmod{p_n} \end{cases}$$

则分别求出 $a_1, a_2, ..., a_n$ 用中国剩余定理合并即可。

1.14 蔡勒公式（日期转星期）

0-星期日，1-星期一，2-星期二，3-星期三，4-星期四，5-星期五，6-星期六

```
int Change(int year, int month, int day){
    if(month == 1 || month == 2) month += 12, year --;
    int c = year/100, y = year%100, m = month, d = day;
    int W = c/4 - 2*c + y + y/4 + 26*(m+1)/10 + d - 1;
    if(W < 0) return (W + (-W/7 + 1)*7)%7;
    return W % 7;
}
```

1.15 二次剩余

```
#define LL long long
using namespace std;
LL mod = 1e9 + 7, I_mul;

struct com {
    LL a, b;
    com(LL a = 0, LL b = 0) : a(a), b(b) {}
    bool operator==(com y) { return a == y.a && b == y.b; }
    com operator*(com y) { return com((a * y.a + I_mul * b % mod * y.b) % mod, (b * y.
        a + a * y.b) % mod); }
    com operator^(LL k) {
        com res(1, 0);
        com x(a, b);
        for (; k; x = x * x, k >>= 1)if (k & 1)res = res * x;
        return res;
    }
```

```
};
//判断一个数x是否有二次剩余
bool check_if_residue(LL x, LL M) { return !x || (com(x, 0) ^ ((M - 1) >> 1)) == com
    (1, 0); }
//二次剩余的两个解放在x0和x1中
void solve(LL n, LL p, LL &x0, LL &x1) {
    if (!n) {
        x0 = x1 = 0;
        return;
    }
    mod = p;
    LL g = rand() % mod;
    while (!g || check_if_residue((g * g + mod - n) % mod, mod))g = rand() % mod;
    I_mul = (g * g + mod - n) % mod;
    x0 = (LL) (com(g, 1) ^ ((mod + 1) >> 1)).a;
    x1 = mod - x0;
}
LL ans1, ans2;
LL p;
int main() {
    int T;
    cin >> T;
    while (T--) {
        scanf("%lld%lld", &p, &mod);
        if (!check_if_residue(p, mod)) {
            puts("Hola!");
            continue;
        }
        solve(p, mod, ans1, ans2);
        if (ans1 == ans2)printf("%lld\n", ans1);
        else {
            if (ans1 > ans2)swap(ans1, ans2);
            printf("%lld␣%lld\n", ans1, ans2);
        }

    }
}
```

1.16 BSGS/EX-BSGS

处理最小化 X 使 $Y^X \equiv Z \pmod P$

```

//A^X=B(mod C)求X C为整数
#define MOD 76543
int hs[MOD], head[MOD], next[MOD], id[MOD], top;

void insert(int x, int y) {
    int k = x % MOD;
    hs[top] = x, id[top] = y, next[top] = head[k], head[k] = top++;
}

int find(int x) {
    int k = x % MOD;
    for (int i = head[k]; i != -1; i = next[i])
```

```
14          if (hs[i] == x)
15              return id[i];
16      return -1;
17  }
18
19  int BSGS(int a, int b, int c) {
20      memset(head, -1, sizeof(head));
21      top = 1;
22      if (b == 1)
23          return 0;
24      int m = sqrt(c * 1.0), j;
25      long long x = 1, p = 1;
26      for (int i = 0; i < m; ++i, p = p * a % c)
27          insert(p * b % c, i);//存的是(a^j*b, j)
28      for (long long i = m;; i += m) {
29          if ((j = find(x = x * p % c)) != -1)
30              return i - j;   //a^(ms-j)=b(mod c)
31          if (i > c)
32              break;
33      }
34      return -1;
35  }
36
37  //EXBSGS
38  LL xiao(LL &a, LL &b, LL &c) {
39      LL r = 0, d, x, y, a1 = 1;
40      while ((d = extend_Euclid(a, c, x, y)) != 1) {
41          if (b % d)
42              return -1;
43          c /= d;
44          b /= d;
45          a1 = a1 * (a / d) % c;
46          r++;
47      }
48      if (r == 0)
49          return 0;
50      extend_Euclid(a1, c, x, y);
51      b = (b * x % c + c) % c;
52      return r;
53  }
54
55  LL extend_BSGS(LL a, LL b, LL c) {
56      a %= c;    //简化运算
57      LL r = 1;
58      for (int i = 0; i < 50; i++) {
59          if ((r - b) % c == 0)
60              return i;
61          r *= a;
62          r %= c;
63      }
64      memset(head, -1, sizeof(head));
65      LL cnt = xiao(a, b, c);
66      if (cnt == -1)
67          return -1;
```

```
68      top = 1;
69      if (b == 1)
70          return cnt;
71      LL m = ceil(sqrt(c * 1.0)), j;
72      LL x = 1, p = 1;
73      for (LL i = 0; i < m; ++i, p = p * a % c)
74          insert(p * b % c, i);
75      for (LL i = m;; i += m) {
76          if ((j = find(x = x * p % c)) != -1)
77              return i - j + cnt;
78          if (i > c)
79              break;
80      }
81      return -1;
82  }
```

1.17 杜教筛

```
1   #define LL long long
2   using namespace std;
3   typedef long long ll;
4   const ll N = 5000000;
5   bool is[N + 10];
6   int prime[N + 10], cnt;
7   LL miu[N + 10], phi[N + 10];
8
9   void init() {
10      phi[1] = miu[1] = 1;
11      is[1] = is[0] = 1;
12      for (int i = 2; i <= N; i++) {
13          if (!is[i]) prime[++cnt] = i, miu[i] = -1, phi[i] = prime[cnt] - 1;
14          for (int j = 1; j <= cnt; j++) {
15              if (i * prime[j] >= N) break;
16              is[i * prime[j]] = 1;
17              if (i % prime[j] == 0) {
18                  miu[i * prime[j]] = 0;
19                  phi[i * prime[j]] = phi[i] * prime[j];
20                  break;
21              } else miu[i * prime[j]] = -miu[i], phi[i * prime[j]] = phi[i] * (prime[j]
                    - 1);
22          }
23      }
24      for (int i = 1; i <= N; i++)miu[i] += miu[i - 1], phi[i] += phi[i - 1];
25  }
26  unordered_map<int, LL> ansmiu, ansphi;
27  LL S_phi(int n) {
28      if (n <= N)return phi[n];
29      if (ansphi[n])return ansphi[n];
30      LL ans = 0;
31      for (int l = 2, r; l <= n; l = r + 1)
32          r = n / (n / l), ans += (r - l + 1) * S_phi(n / l);
33      return ansphi[n] = 1ll * (LL) n * (n + 1) / 2 - ans;//杜教筛公式
34  }
35  LL S_miu(int n) {
```

```
36      if (n <= N)return miu[n];
37      if (ansmiu[n])return ansmiu[n];
38      LL ans = 0;
39      for (int l = 2, r; l <= n; l = r + 1)
40          r = n / (n / l), ans += (r - l + 1) * S_miu(n / l);
41      return ansmiu[n] = 1ll - ans;//杜教筛公式
42  }
43  int n;
44  int main() {
45      init();
46      int T;
47      cin >> T;
48      while (T--) {
49          scanf("%d", &n);
50          printf("%lld %lld\n", S_phi(n), S_miu(n));
51      }
52      return 0;
53  }
```

1.18 min-25 筛

1.18.1 求区间质数和

```
1  #include<iostream>
2  #include<cstdio>
3  #include<cstdlib>
4  #include<cstring>
5  #include<algorithm>
6  #include<cmath>
7  using namespace std;
8
9  #define ll long long
10 #define ld long double
11 #define mem(Arr,x) memset(Arr,x,sizeof(Arr))
12
13 const int maxN=1010000;
14 const int inf=2147483647;
15
16 ll nn,srt;
17 bool notprime[maxN];
18 int pcnt,P[maxN];
19 ll num,Num[maxN+maxN],Id[maxN+maxN];
20 ld G[maxN+maxN],Sum[maxN];
21
22 ld Calc(ll n);
23 void Init();
24 int GetId(ll x);
25
26 int main(){
27     Init();
28     ll L,R;scanf("%lld%lld",&L,&R);
29     printf("%.0LF\n",Calc(R)-Calc(L-1));return 0;
30 }
31
```

```
ld Calc(ll n){
    if (n==0) return 0;
    srt=sqrt(n);nn=n;num=0;mem(Id,0);mem(G,0);mem(Num,0);
    for (ll i=1,j;i<=n;i=j+1){
        j=n/i;Num[++num]=j;G[num]=(j<=1)?(0):((ld)(j-1)*(ld)(j+2)/2.0);
        if (j<=srt) Id[j]=num;
        else Id[i+maxN]=num;
        j=n/j;
    }
    for (int j=1;j<=pcnt;j++){
        for (int i=1;(i<=num)&&(1ll*P[j]*P[j]<=Num[i]);i++){
            G[i]=G[i]-1ll*P[j]*(G[GetId(Num[i]/P[j])]-Sum[j-1]);
        }
    }
    return G[1];
}

void Init(){
    notprime[1]=1;
    for (int i=1;i<maxN;i++){
        if (notprime[i]==0) P[++pcnt]=i,Sum[pcnt]=Sum[pcnt-1]+i;
        for (int j=1;(j<=pcnt)&&(1ll*i*P[j]<maxN);j++){
            notprime[i*P[j]]=1;if (i%P[j]==0) break;
        }
    }
    return;
}

int GetId(ll x){
    if (x<=srt) return Id[x];
    else return Id[nn/x+maxN];
}
```

1.18.2 求区间质数个数

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

#define ll long long
#define mem(Arr,x) memset(Arr,x,sizeof(Arr))

const int maxN=1001000;
const int inf=2147483647;

ll n,srt;
ll G[maxN];
ll num,Num[maxN+maxN],Id[maxN+maxN];
bool notprime[maxN];
int pcnt,P[maxN];
```

```
20
21 void Init(){
22     notprime[1]=1;
23     for (int i=2;i<maxN;i++){
24         if (notprime[i]==0) P[++pcnt]=i;
25         for (int j=1;(j<=pcnt)&&(1ll*i*P[j]<maxN);j++){
26             notprime[i*P[j]]=1;
27             if (i%P[j]==0) break;
28         }
29     }
30     return;
31 }
32 int GetId(ll x){
33     if (x<=srt) return Id[x];
34     else return Id[n/x+maxN];
35 }
36 ll S(ll n,int j);
37
38 int main(){
39     Init();
40     scanf("%lld",&n);srt=sqrt(n);
41     for (ll i=1,j;i<=n;i=j+1){
42         j=n/(n/i);Num[++num]=n/i;G[num]=n/i-1;
43         if (n/i<=srt) Id[n/i]=num;
44         else Id[i+maxN]=num;
45     }
46     for (int j=1;j<=pcnt;j++)
47         for (int i=1;(i<=num)&&(1ll*P[j]*P[j]<=Num[i]);i++)
48             G[i]=G[i]-G[GetId(Num[i]/P[j])]+(j-1);
49     printf("%lld\n",G[1]);
50 }
```

1.19 线性基

```
1 #define size 60
2 struct LinearBase
3 {
4     LL v[size+7],p[size+7];
5     int cnt=0;
6     bool now=0,flag;
7     void clear(){
8         memset(v,0,sizeof(v));
9         memset(p,0,sizeof(p));
10        cnt=0;flag=now=0;
11    }
12    void ins(LL a){
13        now=0;
14        for(int i=size;i>=0;i--)
15            if(a&(1ll<<i)){
16                if(!v[i]){v[i]=a;break;}
17                a^=v[i];
18            }
19        if(!a)flag=1;
20    }
```

```
21      bool find(LL a){
22          if(!a&&flag)return 1;
23          for(int i=size;i>=0;i--)
24              if(a&(1ll<<i)){
25                  if(!v[i])return 0;
26                  a^=v[i];
27              }
28          return 1;
29      }
30      LL getmax(){
31          LL anss=0;
32          for(int i=size;i>=0;i--)if((anss^v[i])>anss)anss^=v[i];
33          return anss;
34      }
35      LL getmin(){
36          if(flag)return 0;
37          for(int i=0;i<=size;i++)if(!v[i])return v[i];
38      }
39      void rebuild(){
40          now=1,cnt=0;
41          memset(p,0,sizeof(p));
42          for(int i=size;i>=0;i--)
43              for(int j=i-1;j>=0;j--)
44                  if(v[i]&(1ll<<j))v[i]^=v[j];
45          for(int i=0;i<=size;i++)if(v[i])p[cnt++]=v[i];
46      }
47      LL kth_min(LL k){
48          if(!now)rebuild();
49          LL res=0;
50          if(flag)k--;
51          if(k>=(1LL<<cnt))return -1;
52          for(int i=size;i>=0;i--)if(k&(1LL<<i))res^=p[i];
53          return res;
54      }
55      LL kth_max(LL k){
56          if(!now)rebuild();
57          LL t=(1LL<<cnt);
58          if(flag)t--;
59          return kth_min(t-k+1);
60      }
61      LinearBase operator+(const LinearBase &a){
62          LinearBase c=a;
63          for(int i=0;i<=size;i++)if(v[i])c.ins(v[i]);
64          return c;
65      }
66      friend LinearBase operator*(const LinearBase &a,const LinearBase &b){
67          LinearBase all,c,d;
68          all.clear(),c.clear(),d.clear();
69          for(int i=size;i>=0;i--)all.v[i]=a.v[i];
70          for(int i=size;i>=0;i--){
71              if(b.v[i]){
72                  LL v=b.v[i],k=1LL>>i;
73                  bool can=1;
74                  for(int j=size;j>=0;j--)
```

```
                    if(v&(1ll<<j))
                        if(all.v[j])v^=all.v[i],k^=d.v[j];
                        else{can=0,all.v[j]=v,d.v[j]=k;break;}
                if(can){
                    LL v=0;
                    for(int j=size;j>=0;j--)if(k&(1ll<<j))v^=b.v[j];
                    c.ins(v);
                }
            }
        }
        return all;
    }
};
int main()
{
    return 0;
}
```

1.20 类欧几里得

```
const LL p=998244353;
LL i2=499122177,i6=166374059;//2^(-1),6^(-1)
struct data{
    data(){f=g=h=0;}
    LL f,g,h;
};
data calc(LL n,LL a,LL b,LL c){
    LL ac=a/c,bc=b/c,m=(a*n+b)/c,n1=n+1,n21=n*2+1;
    data d;
    if(!a){
        d.f=bc*n1%p;
        d.g=bc*n%p*n1%p*i2%p;
        d.h=bc*bc%p*n1%p;
        return d;
    }
    if(a>=c||b>=c){
        d.f=n*n1%p*i2%p*ac%p+n1*bc%p;
        d.g=ac*n%p*n1%p*n21%p*i6%p+bc*n%p*n1%p*i2%p;
        d.h=ac*ac%p*n%p*n1%p*n21%p*i6%p+bc*bc%p*n1%p+ac*bc%p*n%p*n1%p;
        d.f%=p,d.g%=p,d.h%=p;
        data e=calc(n,a%c,b%c,c);
        d.h+=e.h+2*bc%p*e.f%p+2*ac%p*e.g%p;
        d.g+=e.g,d.f+=e.f;
        d.f%=p,d.g%=p,d.h%=p;
        return d;
    }
    data e=calc(m-1,c,c-b-1,a);
    d.f=n*m%p-e.f,d.f=(d.f%p+p)%p;
    d.g=m*n%p*n1%p-e.h-e.f,d.g=(d.g*i2%p+p)%p;
    d.h=n*m%p*(m+1)%p-2*e.g-2*e.f-d.f,d.h=(d.h%p+p)%p;
    return d;
}
```