目录

> *该板子补充红书上没有或者替换成自己熟悉的板子*
> *记得打表*
> *记得前缀和/积（积可逆元处理）*
> *记得map处理直线斜率*

1. 数学

1.1. gcd

> $gcd(a,b) = gcd(b,a\%b)$ *等式(1)*

```
ll gcd(ll a,ll b){
    return b?gcd(b,a%b):a;
}

或者
inline ll gcd(ll a,ll b){
    if(!b) return a;
    while(b^=a^=b^=a%=b);//先做一次取模，然后交换两个元素
    return a;
}
//--------------素数筛法--------------
//埃筛
int primes[N], cnt;     // primes[]存储所有素数
bool st[N];         // st[x]存储x是否被筛掉

void get_primes(int n) // 时间复杂度大约O(nloglogn)
{
    for (int i = 2; i <= n; i ++ )
    {
        if (st[i]) continue;
        primes[cnt ++ ] = i;
        for (int j = i + i; j <= n; j += i)
            st[j] = true;
    }
}

// 欧拉筛 / 线性筛
int primes[N], cnt;     // primes[]存储所有素数
bool st[N];         // st[x]存储x是否被筛掉

void get_primes(int n) // O(n)
{
    for (int i = 2; i <= n; i ++ )
    {
        if (!st[i]) primes[cnt ++ ] = i;
        for (int j = 0; j < cnt && primes[j] <= n / i; j ++ )
        {
            st[primes[j] * i] = true;
            if (i % primes[j] == 0) break;
        }
```

```
    }
}
```

1.2. exgcd

> *求解 $ax+by=c$的x的最小正整数解*

- 首先求解$ax+by=gcd(a,b)(2)$
- 由等式(1)可得$bx1+a\%b*y1=gcd(b,a\%b)=ay1+b*(-\lfloor\frac{a}{b}\rfloor+x1)*y1\ (3)$
- 联立等式(2), (3)得到$x=y_1, y=-\lfloor\frac{a}{b}\rfloor+x1$
- 将所得式子不断递归至$b=0$, 返回最后结果

```
ll exgcd(ll a, ll b, ll &x, ll &y)
{
    if(b == 0)
    {
        x = 1, y = 0;
        return a;
    }
    ll d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}
//------------------自己的板子--------------------
ll exgcd(ll a, ll b, ll& x, ll& y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    ll ans = exgcd(b, a % b, x, y);
    ll tmp = x;
    x = y;
    y = tmp - a / b * y;
    return ans;
}
```

- 设$d=gcd(a,b)$, 等式两边则同时乘上$\frac{c}{d}$则可以获得一组特解
  $x1=x0*\frac{c}{d}$，$y1=y0*\frac{c}{d}$
- 每次将x0减少x，y0则相应增加$\frac{ax}{b}$, 若需要保证y依然为正整数，则$\frac{ax}{b}=\frac{a}{d}, x=\frac{b}{d}$
- 保证$0<x1<x$时，即可得到答案。
- 若$x<0$，则将x取绝对值
- 若$x0<0$，则将x0加上x即可。

```
d = exgcd(a, b, x, y);
if(c% d== 0)
{
    x *= c / d;
    t = b / d;
    t = abs(t);
    x = (x % t + t) % t;
    printf("%d\n", x);
}
```

1.3. 求解线性同余方程

- 给出n个同余方程$N \equiv a_1 \mod p_1, N \equiv a_n \mod p_n$
- 求x的最小正整数解，p_i不互质
- 显然可以化为$k1 * p1 + a1 = k2 * p2 + a2 -> k1 * p1 + (-k2 * p2) = a2 - a1$
- 设$a = p1, b = p2, x = k1, y = (-k2), c = a2 - a1$方程可写为$ax + by = c$
- 则问题转变为上一个题目
- 那么将x化为原方程的最小正整数解，$x = (x * (\frac{c}{d})\%(\frac{b}{d}) + (\frac{b}{d}))\%(\frac{b}{d})$
- 所以$N = a * (x + z * (\frac{b}{d})) + r1 -> N = (\frac{ab}{d}) * z + (a * x + r1)$
- 现在只有z是未知数，所以变成$N \equiv (a * x + r1) mod (\frac{ab}{d}))$的问题
- 继续合并同余方程即可

```
ll M[55], A[55];
ll China(int r){
    ll dm, i, a, b, x, y, d;
    ll c, c1, c2;
    a=M[0];
    c1=A[0];
    for(i = 1; i < r; i++){
        b = M[i];
        c2 = A[i];
        exgcd(a, b ,d, x, y);
        c = c2 - c1;
        if(c % d)
            return -1;//c一定是d的倍数，如果不是，则肯定无解
        dm = b / d;
        x = ((x * (c / d)) % dm + dm) % dm;//保证x为最小正数//c/dm是余数，系数扩大
        c1 = a * x + c1;
        a = a * dm;
    }
    if(c1 == 0){//余数为0，说明M[]是等比数列。且余数都为0
    c1 = 1;
    for(i = 0; i < r; i++)
        c1 = c1 * M[i] / gcd(c1, M[i]);
    }
    return c1;
}
```

- 中国剩余定理

```cpp
//n个方程：x=a[i](mod m[i]) (0<=i<n)
LL china(int n, LL *a, LL *m){
    LL M = 1, ret = 0;
    for(int i = 0; i < n; i ++) M *= m[i];
    for(int i = 0; i < n; i ++){
        LL w = M / m[i];
        ret = (ret + w * inv(w, m[i]) * a[i]) % M;
    }
    return (ret + M) % M;
}
//------------------我的模板--------------------
ll exgcd(ll a, ll b, ll& x, ll& y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    ll ans = exgcd(b, a % b, x, y);
    ll tmp = x;
    x = y;
    y = tmp - a / b * y;
    return ans;
}
ll m[10], a[10]; // 模数m, 余数a, X % m = a
bool solve(ll& m0, ll& a0, ll m, ll a) {
    ll x, y;
    ll gcd = exgcd(m0, m, x, y);
    if (abs(a - a0) % gcd) return false;
    x *= (a - a0) / gcd;
    x %= m / gcd;
    a0 = (x * m0 + a0);
    m0 *= m / gcd;
    a0 %= m0;
    if (a0 < 0) a0 += m0;
    return true;
}
// 无解返回false，有解返回true
// 解的形式最后为a0 + m0 * t (0 <= a0 < m0)
// 解为 (a0 % m0 + m0) % m0
bool MLES(ll& m0, ll& a0, int n) {
    bool flag = true;
    m0 = 1;
    a0 = 0;
    for (int i = 1; i <= n; i++) {
        if (!solve(m0, a0, m[i], a[i])) {
            flag = false;
            break;
        }
    }
    return flag;
}
```

- 进阶: 给出k个模方程组：x mod ai = ri。求x的最小正值。如果不存在这样的x，那么输出-1.

```
#include<cstdio>
#include<algorithm>
using namespace std;
typedef long long LL;
typedef pair<LL, LL> PLL;
LL a[100000], b[100000], m[100000];
LL gcd(LL a, LL b){
    return b ? gcd(b, a%b) : a;
}
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;
}
PLL linear(LL A[], LL B[], LL M[], int n) {
    //求解A[i]x = B[i] (mod M[i]),总共n个线性方程组
    LL x = 0, m = 1;
    for(int i = 0; i < n; i ++) {
        LL a = A[i] * m, b = B[i] - A[i]*x, d = gcd(M[i], a);
        if(b % d != 0)  return PLL(0, -1);//答案，不存在，返回-1
        LL t = b/d * inv(a/d, M[i]/d)%(M[i]/d);
        x = x + m*t;
        m *= M[i]/d;
    }
    x = (x % m + m ) % m;
    return PLL(x, m);//返回的x就是答案，m是最后的lcm值
}
int main(){
    int n;
    while(scanf("%d", &n) != EOF){
        for(int i = 0; i < n; i ++){
            a[i] = 1;
            scanf("%d%d", &m[i], &b[i]);
        }
        PLL ans = linear(a, b, m, n);
        if(ans.second == -1) printf("-1\n");
        else printf("%I64d\n", ans.first);
    }
}
```

1.4. 欧拉定理(扩展， 不适用于矩阵)

- $a^b \equiv a^{b \mod \varphi(p)} \mod p$ (a与p互质)
- $a^b \equiv a^{b \mod \varphi(p)+\varphi(p)} \mod p$ (a与p不互质)

```
ll ph(ll x)
{
    ll res = x, a = x;
    for(ll i = 2; i * i <= x; i++){
        if(a % i == 0){
            res = res / i * (i - 1);
            while(a % i == 0) a /= i;
        }
    }
    if(a > 1) res = res / a * (a - 1);
    return res;
}
ll quick_pow(ll a, ll b, ll mod)
{
    ll ans = 1;
    while(b)
    {
        if(b & 1) ans = (ans * a) % mod;
        a = (a * a) % mod;
        b >>= 1;
    }
    return ans;
}
ll ph_pow(ll a, ll b, ll mod)
{
    if(gcd(b, mod) == 1){
        return quick_pow(a, b % (mod - 1), mod);
    } else {
        ll phi = ph(mod);
        return quick_pow(a, b % (phi) + phi, mod);
    }
}
// ---------------自己的模板-----------------
int prime[100001],mark[1000001];//prime是素数数组，mark为标记不是素数的
数组
int tot,phi[100001];//phi为ϕ(),tot为1~i现求出的素数个数
void getphi(int N){
    phi[1]=1;//ϕ(1)=1
    for(int i=2;i<=N;i++){//从2枚举到N
        if(!mark[i]){//如果是素数
            prime[++tot]=i;//那么进素数数组，指针加1
            phi[i]=i-1;//根据性质1所得
        }
        for(int j=1;j<=tot;j++){//从现求出素数枚举
            if(i*prime[j]>N) break;//如果超出了所求范围就没有意义了
            mark[i*prime[j]]=1;//标记i*prime[j]不是素数
            if(i%prime[j]==0){//应用性质2
```

```
                phi[i*prime[j]]=phi[i]*prime[j];break;
            }
            else phi[i*prime[j]]=phi[i]*phi[prime[j]];//应用性质3
        }
    }
}
// 低内存筛法(埃筛)
// ull phi[MAXN];//phi为ϕ(),tot为1~i现求出的素数个数
// void getphi(int n){
//     for (int i = 0; i <= n; i++)
//         phi[i] = i;
//     for (int i = 2; i <= n; i++) {
//         if (!(phi[i] != i)) {
//             for (int j = i; j <= n; j += i)
//                 phi[j] = phi[j] / i * (i - 1);
//         }
// }
// }
//--------------求一个数的欧拉函数--------------
// 运用唯一分解定理
ll getphi(ll n) {
    ll ans = n;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            ans = ans - ans / i; // ans = ans * (i - 1) / i
            n /= i;
            while (n % i == 0) {
                n /= i;
            }
        }
    }
    if (n > 1) {
        ans = ans - ans / n;
    }
    return ans;
}
```

1.5. 矩阵

- 斐波那契数列求解 $F[n]=F[n-1]+F[n-2]$
- 则可以看作矩阵 $\begin{bmatrix} F[n] \\ F[n-1] \end{bmatrix}=\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}*\begin{bmatrix} F[n-1] \\ F[n-2] \end{bmatrix}$
- 设 $A=\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ $B=\begin{bmatrix} 1 \\ 1 \end{bmatrix}$, 则计算F[n]就是计算 $A^{n-1}*B$, 再取出矩阵第一项即可。
- 上述相对应的操作在矩阵上均可运用。

```
struct Matrix{
    int n, m;
    ll a[maxn][maxn];
    void clear()
    {
```

```
        n = m = 0;
        memset(a, 0, sizeof(a));
    }
    void init (int x)
    {
        n = m = maxn;
        memset(a, 0, sizeof(a));
        if(x)
            for(int i = 0; i < 3; i++)
                a[i][i] = 1;
    }
    Matrix operator *(const Matrix &b) const
    {
        Matrix tmp;
        tmp.clear();
        tmp.n = n;
        tmp.m = b.m;
        for(int i = 0; i < n; i++)
            for(int j = 0; j < b.m; j++)
                for(int k = 0; k < m; k++)
                    (tmp.a[i][j] += a[i][k] * b.a[k][j] % mod) %=
mod;
        return tmp;
    }
    Matrix qpow(ll b)
    {
        Matrix ans, p = *this;
        ans.n = p.n;
        ans.init(1);
        while (b){
            if (b & 1) ans = ans * p;
            p = p * p;
            b >>= 1;
        }
        return ans;
    }
};
```

矩阵快速幂

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 1e5 + 10;
const int mod = 1e9 + 7;
ll n, ax, ay, bx, by, a0, b0;
struct Matrix
{
    ll m[5][5]; // 矩阵
}I;
Matrix mul(Matrix a, Matrix b) {
```

```
    Matrix ans;
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            ans.m[i][j] = 0;
            for (int p = 0; p < 5; p++) {
                ans.m[i][j] = (ans.m[i][j] + a.m[i][p] * b.m[p][j]
% mod) % mod;
            }
        }
    }
    return ans;
}
Matrix qpow(Matrix a, ll b) {
    Matrix ans = I;
    while (b) {
        if (b & 1) ans = mul(ans, a);
        a = mul(a, a);
        b >>= 1;
    }
    return ans;
}
int main() {
    Matrix a, b;
    memset(a.m, 0, sizeof a.m);
    memset(b.m, 0, sizeof b.m);
    memset(I.m, 0, sizeof I.m); // 初始化
    I.m[0][0] = I.m[1][1] = I.m[2][2] = I.m[3][3] = I.m[4][4] = 1;
// 根据矩阵大小设定单位矩阵
    a.m[0][0] = a0 % mod, a.m[0][1] = b0 % mod, a.m[0][2] = a0 * b0
% mod, a.m[0][3] = 1; // 设定初始的矩阵，是行矩阵
    b.m[0][0] = ax % mod, b.m[0][2] = ax * by % mod, b.m[1][1] = bx
% mod;
    b.m[1][2] = ay * bx % mod, b.m[2][2] = ax * bx % mod, b.m[3][0]
= ay % mod;
    b.m[3][1] = by % mod, b.m[3][2] = ay * by % mod, b.m[3][3] = 1;
    b.m[2][4] = 1, b.m[4][4] = 1; // 设定 base矩阵，后面进行快速幂运算
    b = qpow(b, n); // 对base矩阵进行快速幂运算
    Matrix ans = mul(a, b); // 用一个ans矩阵记录最后的答案矩阵，切记初始矩
阵右乘base矩阵，行乘列
    cout << ans.m[0][4] << endl; // 输出答案矩阵中的对应答案即可
    return 0;
}
```

1.6. 广义斐波那契数列循环节

```
typedef long long ll;
typedef __int128 i16;
const int N = 1e5 + 7;
int pr[N], pn;
bitset<N> np;
void init() {
    for(int i = 2; i < N; i++) {
        if(!np[i]) pr[pn++] = i;
        for(int j = 0; j < pn && i * pr[j] < N; j++) {
            np[i * pr[j]] = 1;
            if(i % pr[j] == 0) break;
        }
    }
}
ll getpp(int p) {
    ll ret = 1;
    for(int i = 0; i < pn && pr[i] * pr[i] <= p; i++) {
        if(p % pr[i] == 0) {
            ret *= (pr[i] - 1ll) * (pr[i] + 1);
            p /= pr[i];
            while(p % pr[i] == 0) p /= pr[i], ret *= pr[i];
        }
    }
    if(p != 1) ret *= (p - 1ll) * (p + 1);
    return ret;
}
//使用方式
scanf("%s%d", s, &mod);
int len = strlen(s);
ll ord = getpp(mod), r = 0;//ord为循环节长度
for(int i = 0; i < len; i++) {
    r = ((i16)r * 10 + s[i] - '0') % ord;//简化r的大小
}
```

1.7. 整除分块

正常整除分块为

$ans = (ans + ((n/l) * (r-l+1) mod m) mod m$

```
for(int l = 1, r; l <= n; l = r + 1){
    r = (n / (n / l));
    ans += (r - l + 1) * 1LL * (n / l);
}
// -------------自己的模板-------------
#include<bits/stdc++.h>
using namespace std;
long long n,ans;
int main()
{
```

```
  scanf("%lld",&n);
  for(int l=1,r;l<=n;l=r+1)
     {
        r=n/(n/l); // 有时 r 需要与上界进行 min 运算,参考洛谷P2261
        ans+=(r-l+1)*(n/l);
        cout<<l<<' '<<r<<' '<<ans<<endl;
     }
  printf("%lld",ans);
}
```

加速整除分块$\sum_{i=1}^{n}\lfloor\frac{n}{i}\rfloor=2*\sum_{i=1}^{\sqrt{n}}\lfloor\frac{n}{i}\rfloor-\lfloor\sqrt{n}\rfloor^2$

```
int sn = sqrt(n);
ans1 = -sn * sn;
for(int l = 1, r; l <= sn; l = r + 1){
    r = min(n / (n / l), sn);
    ans1 += 2 * (r - l + 1) * 1LL * (n / l);
}
```

求$\sum_{i=1}^{n}\lfloor n/i\rfloor*i^2 mod 1e9+7$

```
#include <cstdio>
typedef long long LL;
const LL mod = 1e9 + 7;
const LL inv = 166666668;
const LL inv2 = 500000004;
LL f(LL x)
{
    return ((((x * (x + 1)) % mod) * (2 * x + 1) % mod) % mod) *
inv % mod;
}
LL call(LL n, LL m)
{
    LL ans = 0, ans1 = 0, t1 = ((((n * n) % mod) * (n + 1)) % mod)
* inv2 % mod
    LL t2 = ((((m * m) % mod) * (m + 1)) % mod) * inv2 % mod;
    for(LL l = 1, r; l <= n; l = r + 1){
        r = n / (n / l);
        LL tmp = (f(r) - f(l - 1) + mod) % mod;
        ans = (ans + ((n / l) * tmp) % mod) % mod;
    }
    ans = (t1 + mod - ans) % mod;
    for(LL l = 1, r; l <= m; l = r + 1){
        r = m / (m / l);
        LL tmp = (f(r) - f(l - 1) + mod) % mod;
        ans1 = (ans1 + ((m / l) * tmp) % mod) % mod;
    }
    ans1 = (t2 + mod - ans1) % mod;
    return (ans * ans1) % mod;
```

```
}
int main()
{
    LL n, m;
    while(~scanf("%lld %lld", &n, &m)){
        printf("%lld\n", call(n, m));
    }
    return 0;
}
```

1.8. 莫比乌斯反演

这里的ans后面少了mu函数，代码是对的

```
#include<cstdio>
#include<iostream>
#include<cstring>
using namespace std;
const long long MAXN=1e6+5;
//线性筛法求莫比乌斯函数
bool check[MAXN+10];
long long prime[MAXN+10];
int mu[MAXN+10];
void Moblus()
{
    memset(check,false,sizeof(check));
    mu[1] = 1;
    long long tot = 0;
    for(long long i = 2; i <= MAXN; i++)
    {
        if( !check[i] ){
            prime[tot++] = i;
            mu[i] = -1;
        }
        for(long long j = 0; j < tot; j++)
        {
            if(i * prime[j] > MAXN) break;
            check[i * prime[j]] = true;
            if( i % prime[j] == 0){
                mu[i * prime[j]] = 0;
                break;
            }else{
                mu[i * prime[j]] = -mu[i];
            }
        }
    }
```

```
}

int main()
{
    Moblus();
    int t;
    scanf("%d",&t);
    while(t--)
    {
        long long  n;
        scanf("%lld",&n);
        long long ans=3;
        for(long long i=1;i<=n;i++) ans+=mu[i]*(n/i)*(n/i)*(n/i+3);
        printf("%lld\n",ans);
    }
}
```

1.9. Lucas

```
//Lucas定理实现C(n,m)%p的代码：p为素数
LL exp_mod(LL a, LL b, LL p)
{ //快速幂取模
    LL res = 1;
    while(b != 0)
    {
        if(b&1) res = (res * a) % p;
        a = (a*a) % p;
        b >>= 1;
    }
    return res;
}
LL Comb(LL a, LL b, LL p)
{ //求组合数C(a,b)%p
    if(a < b)   return 0;
    if(a == b)  return 1;
    if(b > a - b)   b = a - b;
    LL ans = 1, ca = 1, cb = 1;
    for(LL i = 0; i < b; ++i)
    {
        ca = (ca * (a - i))%p;
        cb = (cb * (b - i))%p;
    }
    ans = (ca*exp_mod(cb, p - 2, p)) % p;
    return ans;
}
LL Lucas(LL n,LL m,LL p)
{ //Lucas定理求C(n,m)%p
     LL ans = 1;
     while(n&&m&&ans)
    {
        ans = (ans*Comb(n%p, m%p, p)) % p;
```

```
        n /= p;
        m /= p;
    }
    return ans;
}
```

- 组合数取模

```
typedef long long LL;
const LL maxn(1000005), mod(1e9 + 7);
LL Jc[maxn];
void calJc()    //求maxn以内的数的阶乘
{
    Jc[0] = Jc[1] = 1;
    for(LL i = 2; i < maxn; i++)
        Jc[i] = Jc[i - 1] * i % mod;
}
```

1.10. 逆元

- 费马小定理求逆元

```
LL pow(LL a, LL n, LL p)    //快速幂 a^n % p
{
    LL ans = 1;
    while(n)
    {
        if(n & 1) ans = ans * a % p;
        a = a * a % p;
        n >>= 1;
    }
    return ans;
}
LL niYuan(LL a, LL b)   //费马小定理求逆元
{
    return pow(a, b - 2, b);
}
LL C(LL a, LL b)    //计算C(a, b)
{
    return Jc[a] * niYuan(Jc[b], mod) % mod

        * niYuan(Jc[a - b], mod) % mod;

}
```

- 拓展欧几里得算法求逆元

```
void exgcd(LL a, LL b, LL &x, LL &y)    //拓展欧几里得算法
{
```

```
    if(!b) x = 1, y = 0;
    else
    {
        exgcd(b, a % b, y, x);
        y -= x * (a / b);
    }
}
LL niYuan(LL a, LL b)   //求a对b取模的逆元
{
    LL x, y;
    exgcd(a, b, x, y);
    return (x + b) % b;
}
```

- 线性求逆元

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#define maxn 3000006
#define LL long long
using namespace std;
int n,p,inv[maxn];
int main(){
    scanf("%d%d",&n,&p);
    inv[1]=1;
    for(int i=2;i<=n;i++)inv[i]=(p-(LL)p/i)*inv[p%i]%p%p;
    for(int i=1;i<=n;i++)printf("%d\n",inv[i]);
    return 0;
}
```

1.11. BM算法

- 给出前k项，即可求出第n项，只能用于符合递推方程形式的方程

```
#include <cstdio>
#include <cstring>
#include <cmath>
#include <algorithm>
#include <vector>
#include <string>
#include <map>
#include <set>
#include <cassert>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
```

```
#define se second
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
const ll mod=1000000007;
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0);
for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
// head
int _;
ll n;
namespace linear_seq {
    const int N=10010;
    ll res[N],base[N],_c[N],_md[N];
    vector<int> Md;
    void mul(ll *a,ll *b,int k) {
        rep(i,0,k+k) _c[i]=0;
        rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=
(_c[i+j]+a[i]*b[j])%mod;
        for (int i=k+k-1;i>=k;i--) if (_c[i])
            rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-
_c[i]*_md[Md[j]])%mod;
        rep(i,0,k) a[i]=_c[i];
    }
    int solve(ll n,VI a,VI b) { // a 系数 b 初值
b[n+1]=a[0]*b[n]+...
        //        printf("%d\n",SZ(b));
        ll ans=0,pnt=0;
        int k=SZ(a);
        assert(SZ(a)==SZ(b));
        rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
        Md.clear();
        rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
        rep(i,0,k) res[i]=base[i]=0;
        res[0]=1;
        while ((1ll<<pnt)<=n) pnt++;
        for (int p=pnt;p>=0;p--) {
            mul(res,res,k);
            if ((n>>p)&1) {
                for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
                rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-
res[k]*_md[Md[j]])%mod;
            }
        }
        rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
        if (ans<0) ans+=mod;
        return ans;
    }
    VI BM(VI s) {
        VI C(1,1),B(1,1);
        int L=0,m=1,b=1;
        rep(n,0,SZ(s)) {
```

```
            ll d=0;
            rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
            if (d==0) ++m;
            else if (2*L<=n) {
                VI T=C;
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                L=n+1-L; B=T; b=d; m=1;
            } else {
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                ++m;
            }
        }
        return C;
    }
    int gao(VI a,ll n) {
        VI c=BM(a);
        c.erase(c.begin());
        rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
        return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
    }
};
int main() {
    for (scanf("%d",&_);_;_--) {
        scanf("%lld",&n);
        vector<int>a;
        a.push_back(1);
        a.push_back(3);
        a.push_back(5);
        a.push_back(7);
        printf("%d\n",linear_seq::gao(a,n-1));
    }
}
```

1.12. 二次剩余

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define re register
#define gc getchar
#define pc putchar
#define puts put_s
#define cs const

namespace IO{
    namespace READONLY{
        cs int Rlen=1<<18|1;
```

```
        char buf[Rlen],*p1,*p2;
        char obuf[Rlen],*p3=obuf;
        char ch[23];
    }
    inline char get_char(){
        using namespace READONLY;
        return (p1==p2)&&(p2=
(p1=buf)+fread(buf,1,Rlen,stdin),p1==p2)?EOF:*p1++;
    }
    inline void put_char(cs char &c){
        using namespace READONLY;
        *p3++=c;
        if(p3==obuf+Rlen)fwrite(obuf,1,Rlen,stdout),p3=obuf;
    }
    inline void put_s(cs char *s){
        for(;*s;++s)pc(*s);
        pc('\n');
    }
    inline void FLUSH(){
        using namespace READONLY;
        fwrite(obuf,1,p3-obuf,stdout);
        p3=obuf;
    }

    inline ll getint(){
        re ll num;
        re char c;
        while(!isdigit(c=gc()));num=c^48;
        while(isdigit(c=gc()))num=(num+(num<<2)<<1)+(c^48);
        return num;
    }
    inline void outint(ll a){
        using namespace READONLY;
        if(a==0)pc('0');
        if(a<0)pc('-'),a=-a;
        while(a)ch[++ch[0]]=a-a/10*10,a/=10;
        while(ch[0])pc(ch[ch[0]--]^48);
    }
}
using namespace IO;

namespace Linear_sieves{
    cs int P=300005;
    int prime[P],pcnt;
    bool mark[P];

    inline void init(int len=P-5){
        mark[1]=true;
        for(int re i=2;i<=len;++i){
            if(!mark[i])prime[++pcnt]=i;
            for(int re j=1;j<=pcnt&&i*prime[j]<=len;++j){
                mark[i*prime[j]]=true;
```

```
                if(i%prime[j]==0)break;
            }
        }
    }
}

namespace Find_root{
    #define complex COMPLEX
    using namespace Linear_sieves;
    inline ll mul(cs ll &a,cs ll &b,cs ll &mod){
        return (a*b-(ll)((long double)a/mod*b)*mod+mod)%mod;
    }
    inline ll quickpow(ll a,ll b,cs ll &mod,ll res=1){
        for(;b;b>>=1,a=mul(a,a,mod))if(b&1)res=mul(res,a,mod);
        return res;
    }

    inline ll ex_gcd(cs ll &a,cs ll &b,ll &x,ll &y){
        if(!b){
            y=0;
            x=1;
            return a;
        }
        ll t=ex_gcd(b,a-a/b*b,y,x);
        y-=(a/b)*x;
        return t;
    }
    inline ll inv(cs ll a,cs ll mod){
        ll x,y;
        ll t=ex_gcd(a,mod,x,y);
        return (x%mod+mod)%mod;
    }

    ll W,Mod;
    class complex{
        public:
            ll x,y;
            complex(cs ll &_x=0,cs ll &_y=0):x(_x),y(_y){}
            inline friend complex operator*(cs complex &a,cs
complex &b){
                return complex(

(mul(a.x,b.x,Mod)+mul(mul(a.y,b.y,Mod),W,Mod))%Mod,
                    (mul(a.x,b.y,Mod)+mul(a.y,b.x,Mod))%Mod);
            }
    };

    complex quickpow(complex a,ll b){
        complex res(1,0);
        for(;b;b>>=1,a=a*a)if(b&1)res=res*a;
        return res;
    }
```

```
    inline bool isprime(ll x){
        if(x<=P-5)return !mark[x];

 if(x%2==0||x%3==0||x%5==0||x%7==0||x%31==0||x%24251==0)return
false;
        re ll t=x-1,s;
        t>>=(s=__builtin_ctzll(t));
        for(int re i=1;i<=5;++i){
            re ll p=prime[rand()%pcnt+1];
            re ll num=quickpow(p,t,x),pre=num;
            for(int re j=0;j<s;++j){
                num=mul(num,num,x);
                if(num==1&&pre!=x-1&&pre!=1)return false;
                pre=num;
                if(num==1)break;
            }
            if(num^1)return false;
        }
        return true;
    }

    inline ll Pollard_rho(ll x){
        if(x%2==0)return 2;
        if(x%3==0)return 3;
        if(x%5==0)return 5;
        if(x%7==0)return 7;
        if(x%31==0)return 31;
        if(x%24251==0)return 24251;
        re ll n=0,m=0,t=1,q=1,c=rand()%(x-2)+2;
        for(int re k=2;;k<<=1,m=n,q=1){
            for(int re i=1;i<=k;++i){
                n=(mul(n,n,x)+c)%x;
                q=mul(q,abs(n-m),x);
            }
            if((t=__gcd(q,x))>1)return t;
        }
    }

    ll fact[60],cntf;
    inline void sieves(ll x){
        if(x==1)return ;
        if(isprime(x)){fact[++cntf]=x;return;}
        re ll p=x;
        while(p==x)p=Pollard_rho(p);
        sieves(p);
        while(x%p==0)x/=p;
        sieves(x);
    }

    inline ll solve_2k(ll a,ll k){
        if(a%(1<<k)==0)return 0;
```

```
        a%=(1<<k);
        re ll t=0,res=1;
        a>>=(t=__builtin_ctzll(a));
        if((a&7)^1)return -1;
        if(t&1)return -1;
        k-=t;
        for(int re i=4;i<=k;++i){
            res=(res+(a%(1<<i)-res*res)/2)%(1<<k);
        }
        res%=1<<k;
        if(res<0)res+=1<<k;
        return res<<(t>>1);
    }

    inline ll solve_p(ll a,ll p){
        a%=p;
        if(quickpow(a,(p-1)>>1,p)==p-1)return -1;
        re ll b;
        Mod=p;
        while(true){
            b=rand()%p;
            W=(mul(b,b,p)-a+p)%p;
            if(quickpow(W,(p-1)>>1,p)==p-1)break;
        }
        re ll ans=quickpow(complex(b,1),(p+1)>>1).x;
        return min(ans,p-ans);
    }

    inline ll solve_pk(ll a,ll k,ll p,ll mod){
        if(a%mod==0)return 0;
        re ll t=0,hmod=1;
        while(a%p==0)a/=p,++t,hmod*=(t&1)?p:1;
        if(t&1)return -1;
        k-=t;
        mod/=hmod*hmod;
        re ll res=solve_p(a,p);
        if(res==-1)return -1;
        complex tmp(res,1);
        W=a;
        Mod=mod;
        tmp=quickpow(tmp,k);
        res=mul(tmp.x,inv(tmp.y,Mod),Mod);
        return res*hmod;
    }

    ll remain[20],mod[20],p;
    inline ll CRT(){
        re ll ans=0;
        for(int re i=1;i<=cntf;++i){
            ans=
(ans+mul(mul(p/mod[i],inv(p/mod[i],mod[i]),p),remain[i],p))%p;
        }
```

```
        return ans;
    }

    inline ll solve(ll a,ll pmod){
        a%=pmod;
        cntf=0;
        p=pmod;
        sieves(pmod);
        if(cntf>1)sort(fact+1,fact+cntf+1);
        if(cntf>1)cntf=unique(fact+1,fact+cntf+1)-fact-1;
        for(int re i=1;i<=cntf;++i){
            re ll now=0,rmod=1;

 while(pmod%fact[i]==0)pmod/=fact[i],++now,rmod*=fact[i];
            mod[i]=rmod;
            if(fact[i]==2)remain[i]=solve_2k(a,now);
            else remain[i]=solve_pk(a,now,fact[i],rmod);
            if(remain[i]==-1)return -1;
        }
        return CRT();
    }

    #undef complex
}

int T;
signed main(){
    srand(time(0));
    Linear_sieves::init();
    T=getint();
    const ll p = 1e9+7;
    while(T--){
        re ll b=getint(),c=getint(),ans;
        ans=Find_root::solve(b * b - 4 * c,4*p);
        if(ans == -1){
            b += p;
            ans=Find_root::solve(b * b - 4 * c,4*p);
            if(ans == -1){
                printf("-1 -1\n");
                continue;
            }
        }
        ll x = ans + b;
        x >>= 1;
        x %= p;
        ll y = (b - x + p) % p;
        if(x > y){
            swap(x, y);
        }
        printf("%lld %lld\n", x, y);
    }
    FLUSH();
```

```
    return 0;
}
```

1.13. k次幂和

```
#include<cstdio>
#include<cstdlib>
#include<algorithm>
using namespace std;
typedef long long ll;
const int P=1000000009;
const int INV2=500000005;
const int SQRT5=383008016;
const int INVSQRT5=276601605;
const int A=691504013;
const int B=308495997;
const int N=100005;
ll n,K;
ll fac[N],inv[N];
ll pa[N],pb[N];
inline void Pre(int n)
{
    fac[0]=1;
    for (int i=1;i<=n;i++)
        fac[i]=fac[i-1]*i%P;
    inv[1]=1;
    for (int i=2;i<=n;i++)
        inv[i]=(P-P/i)*inv[P%i]%P;
    inv[0]=1;
    for (int i=1;i<=n;i++)
        inv[i]=inv[i]*inv[i-1]%P;
    pa[0]=1;
    for (int i=1;i<=n;i++)
        pa[i]=pa[i-1]*A%P;
    pb[0]=1;
    for (int i=1;i<=n;i++)
        pb[i]=pb[i-1]*B%P;
}
inline ll C(int n,int m)
{
    return fac[n]*inv[m]%P*inv[n-m]%P;
}
inline ll Pow(ll a,ll b)
{
    ll ret=1;
    for (;b;b>>=1,a=a*a%P)
        if (b&1)
            ret=ret*a%P;
        return ret;
}
inline ll Inv(ll x)
```

```
{
    return Pow(x,P-2);
}
inline void Solve()
{
    ll Ans=0;
    for (int j=0;j<=K;j++){
        ll t=pa[K-j]*pb[j]%P,tem;
        tem=t==1?n%P:t*(Pow(t,n)-1+P)%P*Inv(t-1)%P;
        if (~j&1)
            Ans+=C(K,j)*tem%P,Ans%=P;
        else
            Ans+=P-C(K,j)*tem%P,Ans%=P;
    }
    Ans=Ans*Pow(INVSQRT5,K)%P;
    printf("%lld\n",Ans);
}
int main()
{
    int T;
    freopen("t.in","r",stdin);
    freopen("t.out","w",stdout);
    Pre(100000);
    scanf("%d",&T);
    while (T--){
        scanf("%lld%lld",&n,&K);
        Solve();
    }
    return 0;
}
```

1.14. 杜教筛

> *筛μ和φ的板子,根据内存来限制先预处理多少*

```
#include<bits/stdc++.h>
#include<tr1/unordered_map>
#define N 6000010
using namespace std;
template<typename T>inline void read(T &x)
{
    x=0;
    static int p;p=1;
    static char c;c=getchar();
    while(!isdigit(c)){if(c=='-')p=-1;c=getchar();}
    while(isdigit(c)) {x=(x<<1)+(x<<3)+(c-48);c=getchar();}
    x*=p;
}
bool vis[N];
int mu[N],sum1[N],phi[N];
long long sum2[N];
```

```
int cnt,prim[N];
tr1::unordered_map<long long,long long>w1;
tr1::unordered_map<int,int>w;
void get(int maxn)
{
    phi[1]=mu[1]=1;
    for(int i=2;i<=maxn;i++)
    {
        if(!vis[i])
        {
            prim[++cnt]=i;
            mu[i]=-1;phi[i]=i-1;
        }
        for(int j=1;j<=cnt&&prim[j]*i<=maxn;j++)
        {
            vis[i*prim[j]]=1;
            if(i%prim[j]==0)
            {
                phi[i*prim[j]]=phi[i]*prim[j];
                break;
            }
            else mu[i*prim[j]]=-mu[i],phi[i*prim[j]]=phi[i]*
(prim[j]-1);
        }
    }
    for(int i=1;i<=maxn;i++)sum1[i]=sum1[i-1]+mu[i],sum2[i]=sum2[i-
1]+phi[i];
}
int djsmu(int x)
{
    if(x <= 6000000){
        return sum[x];
    }
    if(w[x]){
        return w[x];
    }
    int ans = 1;
    for(int l = 2, r; l >= 0 && l <= x; l = r + 1)
    {
        r = x / (x / l);
        ans -= (r - l + 1) * djsmu(x / l);
    }
    return w[x]=ans;
}
long long djsphi(long long x)
{
    if(x<=6000000)return sum2[x];
    if(w1[x])return w1[x];
    long long ans=x*(x+1)/2;
    for(long long l=2,r;l<=x;l=r+1)
    {
        r=x/(x/l);
```

```
        ans-=(r-l+1)*djsphi(x/l);
    }
    return w1[x]=ans;
}
int main()
{
    int t,n;
    read(t);
    get(6000000);
    while(t--)
    {
        read(n);
        printf("%lld %d\n",djsphi(n),djsmu(n));
    }
    return 0;
}
```

1.15. 一元三次方程

> *求解形如$ax^3+bx^2+cx+d=0$只有单实数根情况的解*

```
double calc(double a, double b,double c, double d)
{
    double p = (3 * a * c - b * b) / 3 / (a * a);
    double q = (27 * a * a * d - 9 * a * b * c + 2 * b * b * b) /
27 / (a * a * a);
    double tq = q / 2, tp = p / 3;
    return pow(-tq + sqrt(tq * tq + tp * tp * tp), 1.0 / 3) -
pow(tq + sqrt(tq * tq + tp * tp * tp), 1.0 / 3) - b / 3 / a;
}
```

1.16. 大整数幂和乘法

```
ll qmul(ll a,ll b,ll c)
{
    ll ret=0;
    while (b)
    {
        if (b & 1)
            ret = (ret + a) % c;
        b >>= 1;
        a = (a << 1) % c;
    }
    return ret;
}
ll qpow(ll a,ll b,ll c)
{
    ll ret = 1;
    // 有时候可以在这a = a % c
    while (b)
    {
```

```
        if (b & 1)
            ret = qmul(ret, a, c);
            a = qmul(a, a, c);
        b >>= 1;
       // a = qmul(a, a, c);
    }
    return ret % c;
}
```

1.17. 高斯消元解线性方程组

```
// a[N][N]是增广矩阵
int gauss()
{
    int c, r;
    for (c = 0, r = 0; c < n; c ++ )
    {
        int t = r;
        for (int i = r; i < n; i ++ )   // 找到绝对值最大的行
            if (fabs(a[i][c]) > fabs(a[t][c]))
                t = i;

        if (fabs(a[t][c]) < eps) continue;

        for (int i = c; i <= n; i ++ ) swap(a[t][i], a[r][i]);
 // 将绝对值最大的行换到最顶端
        for (int i = n; i >= c; i -- ) a[r][i] /= a[r][c];      //
将当前上的首位变成1
        for (int i = r + 1; i < n; i ++ )       // 用当前行将下面所有
的列消成0
            if (fabs(a[i][c]) > eps)
                for (int j = n; j >= c; j -- )
                    a[i][j] -= a[r][j] * a[i][c];

        r ++ ;
    }

    if (r < n)
    {
        for (int i = r; i < n; i ++ )
            if (fabs(a[i][n]) > eps)
                return 2; // 无解
        return 1; // 有无穷多组解
    }

    for (int i = n - 1; i >= 0; i -- )
        for (int j = i + 1; j < n; j ++ )
            a[i][n] -= a[i][j] * a[j][n];

    return 0; // 有唯一解
```

```
}
```

2. 二、数据结构

2.1. kd树

> *x维数为2的KDtree模板*

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const LL INF = 0x3f3f3f3f3f3f3f3f;
const int MAXN = 1e5 + 10;

struct Node {
    int lson, rson;
    LL Min[2], Max[2], x[2];
    int id;
} kdt[MAXN << 1], tmp;

int root, cmp_x;
LL ans, xx0, xx1;

bool cmp (const Node &a, const Node &b) {
    return a.x[cmp_x] < b.x[cmp_x] || (a.x[cmp_x] == b.x[cmp_x] &&
a.x[cmp_x^1] < b.x[cmp_x^1]);
}

// 更新每个结点的边界信息
void pushUp(int u, int v) {
    for (int i = 0; i < 2; i++) kdt[u].Min[i] = min(kdt[u].Min[i],
kdt[v].Min[i]);
    for (int i = 0; i < 2; i++) kdt[u].Max[i] = max(kdt[u].Max[i],
kdt[v].Max[i]);
}

int kdtBuild(int l, int r, int X) {
    int mid = (l + r) >> 1;
    kdt[mid].lson = kdt[mid].rson = 0;
    cmp_x = X;
    nth_element(kdt + l + 1, kdt + mid + 1, kdt + r + 1, cmp);
    // 将编号为mid的元素放在中间，比它小的放在前面，比它大的放后面
    kdt[mid].Min[0] = kdt[mid].Max[0] = kdt[mid].x[0];
    kdt[mid].Min[1] = kdt[mid].Max[1] = kdt[mid].x[1];
    if (l != mid) kdt[mid].lson = kdtBuild(l, mid - 1, X ^ 1);
    if (r != mid) kdt[mid].rson = kdtBuild(mid + 1, r, X ^ 1);
    if (kdt[mid].lson) pushUp(mid, kdt[mid].lson);
```

```
    if (kdt[mid].rson) pushUp(mid, kdt[mid].rson);
    return mid;
}

// 插入新的结点
void kdtInsert(int now) {
    int X = 0, p = root;
    while (true) {
        pushUp(p, now);
        if (kdt[now].x[X] < kdt[p].x[X]) {
            if (!kdt[p].lson) {
                kdt[p].lson = now;
                return;
            }
            else p = kdt[p].lson;
        }
        else {
            if (!kdt[p].rson) {
                kdt[p].rson = now;
                return;
            }
            else p = kdt[p].rson;
        }
    }
}

// 点(x,y)在结点id的边界范围内能得到的最大距离上界
LL getMaxDis(int id, LL x0, LL x1) {
    LL res = 0;
    if (x0 < kdt[id].Min[0]) res += (kdt[id].Min[0] - x0) *
(kdt[id].Min[0] - x0);
    if (x0 > kdt[id].Max[0]) res += (kdt[id].Max[0] - x0) *
(kdt[id].Max[0] - x0);
    if (x1 < kdt[id].Min[1]) res += (kdt[id].Min[1] - x1) *
(kdt[id].Min[1] - x1);
    if (x1 > kdt[id].Max[1]) res += (kdt[id].Max[1] - x1) *
(kdt[id].Max[1] - x1);
    return res;
}

LL dist(int id, LL x0, LL x1) {
    return (kdt[id].x[0] - x0)
    * (kdt[id].x[0] - x0) + (kdt[id].x[1] - x1) * (kdt[id].x[1] -
x1);
}

void kdtQuery(int p) {
    LL dl = INF, dr = INF, d;
    d = dist(p, xx0, xx1);
    if (kdt[p].x[0] == xx0 && kdt[p].x[1] == xx1) d = INF;
      // 查询(x,y)时要将(x,y)到自己的距离设为INF
    ans = min(ans, d);
```

```
    if (kdt[p].lson) dl = getMaxDis(kdt[p].lson, xx0, xx1);
    if (kdt[p].rson) dr = getMaxDis(kdt[p].rson, xx0, xx1);
    if (dl < dr) {
        if (dl < ans) kdtQuery(kdt[p].lson);
        if (dr < ans) kdtQuery(kdt[p].rson);
    }
    else {
        if (dr < ans) kdtQuery(kdt[p].rson);
        if (dl < ans) kdtQuery(kdt[p].lson);
    }
}

LL answer[MAXN];

int main() {
    //freopen("in.txt", "r", stdin);
    int T;
    scanf("%d", &T);
    while (T--) {
        int n;
        scanf("%d", &n);
        for (int i = 1; i <= n; i++) {
            scanf("%I64d%I64d", &kdt[i].x[0], &kdt[i].x[1]);
            kdt[i].id = i;
        }
        root = kdtBuild(1, n, 0);
        for (int i = 1; i <= n; i++) {
            ans = INF;
            xx0 = kdt[i].x[0]; xx1 = kdt[i].x[1];
            kdtQuery(root);
            answer[kdt[i].id] = ans;
           // printf("---%d\n", ans);
        }
        for (int i = 1; i <= n; i++)
            printf("%I64d\n", answer[i]);
    }
    return 0;
}
```

3. dp

3.1. 区间dp

```
#include <cstdio>
#include <queue>
#include <cstring>
#include <algorithm>
using namespace std;
#define mst(a,b) memset((a),(b),sizeof(a))
#define rush() int T;scanf("%d",&T);while(T--)
```

```
typedef long long ll;
const int maxn = 205;
const ll mod = 1e9+7;
const ll INF = 1e18;
const double eps = 1e-9;
int n,x;
int sum[maxn];
int dp[maxn][maxn];
int main()
{
    while(~scanf("%d",&n))
    {
        sum[0]=0;
        mst(dp,0x3f);
        for(int i=1;i<=n;i++)
        {
            scanf("%d",&x);
            sum[i]=sum[i-1]+x;
            dp[i][i]=0;
        }
        for(int len=2;len<=n;len++)
        for(int i=1;i<=n;i++)
        {
            int j=i+len-1;
            if(j>n) continue;
            for(int k=i;k<j;k++)
            {
                dp[i][j]=min(dp[i][j],dp[i][k]+dp[k+1][j]+sum[j]-
sum[i-1]);
            }
        }
        printf("%d\n",dp[1][n]);
    }
    return 0;
}
```

3.2. 状压dp

```
#include<iostream>
#include<vector>
#include<queue>
#include<cstdio>
#include<cstring>
using namespace std;
int RD(){
    int out = 0,flag = 1;char c = getchar();
    while(c < '0' || c >'9'){if(c == '-')flag = -1;c = getchar();}
    while(c >= '0' && c <= '9'){out = out * 10 + c - '0';c =
getchar();}
    return flag * out;
    }
```

```
const int maxn = 2048;
int num,m,numd;
struct Node{
    int dp,step;
    };
int vis[maxn];
int map[maxn][maxn];
void BFS(int n){
    queue<Node>Q;
    Node fir;fir.step = 0,fir.dp = n;//初始状态入队
    Q.push(fir);
    while(!Q.empty()){//BFS
        Node u = Q.front();
        Q.pop();
        int pre = u.dp;
        for(int i = 1;i <= m;i++){//枚举每个操作
            int now = pre;
            for(int j = 1;j <= num;j++){
                if(map[i][j] == 1){
                    if( (1 << (j - 1)) & now){
                        now = now ^ (1 << (j - 1));//对状态进行操作
                        }
                    }
                else if(map[i][j] == -1){
                    now = ( (1 << (j - 1) ) | now);//对状态进行操作
                    }
                }
            fir.dp = now,fir.step = u.step + 1;//记录步数
            if(vis[now] == true){
                continue;
                }
            if(fir.dp == 0){//达到目标状态
                vis[0] = true;//相当于一个标记flag
                cout<<fir.step<<endl;//输出
                return ;//退出函数
                }
            Q.push(fir);//新状态入队
            vis[now] = true;//表示这个状态操作过了（以后在有这个状态就不用
试了）
            }
        }
    }
int main(){
    num = RD();m = RD();
    int n = (1 << (num)) - 1;
    for(int i = 1;i <= m;i++){
        for(int j = 1;j <= num;j++){
            map[i][j] = RD();
            }
        }
    BFS(n);
    if(vis[0] == false)
```

```
        cout<<-1<<endl;
    return 0;
}
```

3.3. 数位dp

```
typedef long long ll;
int a[20];
ll dp[20][state];//不同题目状态不同
ll dfs(int pos,/*state变量*/,bool lead/*前导零*/,bool limit/*数位上界变量*/)
//不是每个题都要判断前导零
{
    //递归边界，既然是按位枚举，最低位是0，那么pos==-1说明这个数我枚举完了
    if(pos==-1) return 1;/*这里一般返回1，表示你枚举的这个数是合法的，
    那么这里就需要你在枚举时必须每一位都要满足题目条件
    ，也就是说当前枚举到pos位，
    一定要保证前面已经枚举的数位是合法的。不过具体题目不同或者写法不同的话不一定要返回1 */
    //第二个就是记忆化(在此前可能不同题目还能有一些剪枝)
    if(!limit && !lead && dp[pos][state]!=-1) return dp[pos][state];
    /*常规写法都是在没有限制的条件记忆化，这里与下面记录状态是对应，
    具体为什么是有条件的记忆化后面会讲*/
    int up=limit?a[pos]:9;//根据limit判断枚举的上界up;这个的例子前面用213讲过了
    ll ans=0;
    //开始计数
    for(int i=0;i<=up;i++)//枚举，然后把不同情况的个数加到ans就可以了
    {
        if() ...
        else if()...
        ans+=dfs(pos-1,/*状态转移*/,lead && i==0,limit && i==a[pos])
        //最后两个变量传参都是这样写的
        /*这里还算比较灵活，不过做几个题就觉得这里也是套路了
        大概就是说，我当前数位枚举的数是i，然后根据题目的约束条件分类讨论
        去计算不同情况下的个数，还有要根据state变量来保证i的合法性，比如题目
        要求数位上不能有62连续出现,那么就是state就是要保存前一位pre,然后分类，
            前一位如果是6那么这意味就不能是2，这里一定要保存枚举的这个数是合法*/
    }
    //计算完，记录状态
    if(!limit && !lead) dp[pos][state]=ans;
    /*这里对应上面的记忆化，在一定条件下时记录，
    保证一致性，当然如果约束条件不需要考虑lead，这里就是lead就完全不用考虑了*/

    return ans;
}
ll solve(ll x)
{
```

```
    int pos=0;
    while(x)//把数位都分解出来
    {
        a[pos++]=x%10;
        //个人老是喜欢编号为[0,pos),看不惯的就按自己习惯来，反正注意数位边界
就行
        x/=10;
    }
    return dfs(pos-1/*从最高位开始枚举*/,/*一系列状态 */,true,true);
    //刚开始最高位都是有限制并且有前导零的，显然比最高位还要高的一位视为0嘛
}
int main()
{
    ll le,ri;
    while(~scanf("%lld%lld",&le,&ri))
    {
        //初始化dp数组为-1,这里还有更加优美的优化,后面讲
        printf("%lld\n",solve(ri)-solve(le-1));
    }
}
```

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <vector>
#include <cmath>
#include <algorithm>
#include <string>
#include <cstring>
#include <map>
using namespace std;
typedef long long ll;
int a[20];
ll dp[20][2];
ll dfs(int pos, int pre, int state, bool limit)//不是每个题都要判断前导
零
{
    if(pos==-1) return 1;
    if(!limit && dp[pos][state]!=-1) return dp[pos][state];
    int up=limit?a[pos]:9;
    ll ans=0;
    for(int i=0;i<=up;i++)
    {
        if(i == 4) continue;
        else if(pre == 6 && i == 2) continue;
        ans+=dfs(pos-1, i, i == 6 ? 1 : 0, limit && i==a[pos]);
    }
    if(!limit) dp[pos][state]=ans;
    return ans;
}
```

```
ll solve(ll x)
{
    int pos=0;
    while(x)
    {
        a[pos++]=x%10;
        x/=10;
    }
    return dfs(pos - 1, 0, 0, true);
}
int main()
{
    ll le,ri;
    while(~scanf("%lld%lld",&le,&ri) && (le || ri))
    {
        memset(dp, -1, sizeof(dp));
        printf("%lld\n",solve(ri)-solve(le-1));
    }
    return 0;
}
```

4. 计算几何

4.1. 红书

4.2. 三维向量旋转

```
#include<bits/stdc++.h>
#define eps 1e-10
#define pi acos(-1.0)

using namespace std;
int dcmp(double x){if (fabs(x)<eps)return 0;else return x<0?-1:1;}
struct Point3
{
    double x,y,z;
    Point3(double x=0,double y=0,double z=0):x(x),y(y),z(z){};
};
typedef Point3 Vector3;
Vector3 operator + (Vector3 a,Vector3 b){return
Vector3(a.x+b.x,a.y+b.y,a.z+b.z);}
Vector3 operator - (Vector3 a,Vector3 b){return Vector3(a.x-
b.x,a.y-b.y,a.z-b.z);}
Vector3 operator * (Vector3 a,double b){return
Vector3(a.x*b,a.y*b,a.z*b);}
Vector3 operator / (Vector3 a,double b){return
Vector3(a.x/b,a.y/b,a.z/b);}
bool operator == (Vector3 a,Vector3 b){return a.x==b.x && a.y==b.y
&& a.z==b.z;}
```

```
double Dot3(Vector3 a,Vector3 b){return a.x*b.x+a.y*b.y+a.z*b.z;}
 //点积
double Length3(Vector3 a){return sqrt(Dot3(a,a));}
Vector3 Cross3(Vector3 a,Vector3 b) //叉积
{return Vector3(a.y*b.z-a.z*b.y,a.z*b.x-a.x*b.z,a.x*b.y-a.y*b.x);}
double Angle3(Vector3 a,Vector3 b){return
acos(Dot3(a,b)/Length3(a)/Length3(b));}
// 范围[0,180]
//点到线段的距离
double DistanceToSeg3(Point3 p,Point3 a,Point3 b)
{
    if(a==b) return Length3(p-a);
    Vector3 v1=b-a,v2=p-a,v3=p-b;
    if(dcmp(Dot3(v1,v2))<0) return Length3(v2);
    else if(dcmp(Dot3(v1,v3))>0) return Length3(v3);
    else return Length3(Cross3(v1,v2))/Length3(v1);
}
//点p绕向量ov旋转ang角度，旋转方向是向量ov叉乘向量op
Point3 rotate3(Point3 p,Vector3 v,double ang)
{
    double ret[3][3],a[3];
    v = v / Length3(v);
    ret[0][0] = (1.0 - cos(ang)) * v.x * v.x + cos(ang);
    ret[0][1] = (1.0 - cos(ang)) * v.x * v.y - sin(ang) * v.z;
    ret[0][2] = (1.0 - cos(ang)) * v.x * v.z + sin(ang) * v.y;
    ret[1][0] = (1.0 - cos(ang)) * v.y * v.x + sin(ang) * v.z;
    ret[1][1] = (1.0 - cos(ang)) * v.y * v.y + cos(ang);
    ret[1][2] = (1.0 - cos(ang)) * v.y * v.z - sin(ang) * v.x;
    ret[2][0] = (1.0 - cos(ang)) * v.z * v.x - sin(ang) * v.y;
    ret[2][1] = (1.0 - cos(ang)) * v.z * v.y + sin(ang) * v.x;
    ret[2][2] = (1.0 - cos(ang)) * v.z * v.z + cos(ang);
    for (int i=0;i<3;i++) a[i]=ret[i][0]*p.x+ret[i][1]*p.y+ret[i]
[2]*p.z;
    return Point3(a[0],a[1],a[2]);
}
Point3
face,head,st,en,a,vx=Point3(1,0,0),vy=Point3(0,1,0),vz=Point3(0,0,1
);
int main()
{
    int T,n;
    scanf("%d",&T);
    while (T--)
    {
        double ans=1e8,ang,dx,dy,dz,d; char x[3];
        face=Point3(1,0,0); head=Point3(0,0,1);
        scanf("%lf%lf%lf",&st.x,&st.y,&st.z);
        scanf("%lf%lf%lf",&en.x,&en.y,&en.z);
        scanf("%d",&n);
        while (n--)
        {
            scanf("%lf %s %lf",&d,x,&ang);
```

```
            dx=d*cos(Angle3(vx,face));
            dy=d*cos(Angle3(vy,face));
            dz=d*cos(Angle3(vz,face));
            a=st+Point3(dx,dy,dz);
            ans=min(ans,DistanceToSeg3(en,st,a));
            st=a;
            if (x[0]=='U')
            {
                Vector3 v=Cross3(face,head);
                face=rotate3(face,v,ang);
                head=rotate3(head,v,ang);
            }else
            if (x[0]=='D')
            {
                Vector3 v=Cross3(head,face);
                face=rotate3(face,v,ang);
                head=rotate3(head,v,ang);
            }else
            if (x[0]=='L')
            {
                face=rotate3(face,head,ang);
            }else
            if (x[0]=='R')
            {
                Vector3 v=head*(-1);
                face=rotate3(face,v,ang);
            }
        }
        printf("%.2f\n",ans);
    }
    return 0;
}
```

4.3. 直线交点个数

```
#include<iostream>
#include<cstring>
#include<string>
#include<queue>
#include<stack>
#include<algorithm>
#include<stdio.h>
#include<map>
#include<set>
using namespace std;
typedef long long ll;
typedef pair<ll,ll>P;
typedef pair<pair<ll,ll>,ll>Pi;
const int maxn=100010;
map<pair<ll,ll>,ll>mp1;
//两个参数a,b，代表形如a*x+b*y=c（c任意）的直线有多少个
```

```
map<pair<pair<ll,ll>,ll>,ll>mp2;
//三个参数a,b,c,代表形如a*x+b*y=c的直线有多少个，即相同直线有多少个
ll cnt,ans,n;
int main()
{
    ios::sync_with_stdio(0);
    int T;
    cin>>T;
    while(T--){
        ans=cnt=0;
        cin>>n;
        mp1.clear(),mp2.clear();
        for(int i=1;i<=n;i++){
            ll x1,x2,y1,y2;
            cin>>x1>>y1>>x2>>y2;
            ll a=x1-x2,b=y1-y2,c=x1*y2-x2*y1;
            ll g=__gcd(a,b);
            a/=g;b/=g;c/=g;
            mp1[P(a,b)]++;
            mp2[Pi(P(a,b),c)]++;
            ans+=i-1+mp2[Pi(P(a,b),c)]-mp1[P(a,b)];
            //假设与前i-1条边都相交，需要减去与他平行而不重合的线段
        }
        cout<<ans<<endl;
    }
    return 0;
}
```

4.4. 三点共线组数

```
#include <cstdio>
#include <map>
#include <cmath>
#include <algorithm>
#include <tuple>
#define mt(a,b,c,g) make_tuple(a/g,b/g,c/g)
using namespace std;
const int maxn = 1005;
typedef tuple<int, int, int> abc;
map<abc, int> mp;
struct Point{
    int x, y;
}point[maxn];
int f[maxn][4];
void init()
{
    for(int i = 0; i < maxn; i++){
        f[i][0] = 1;
        for(int j = 1; j < 4; j++){
            if(j > i){
                f[i][j] = 0;
```

```
            } else {
                f[i][j] = f[i - 1][j - 1] + f[i - 1][j];
            }
        }
    }
}
int count(int n)
{
    int ans = 0, a, b, c, g;
    Point tmp, tmp1;
    mp.clear();
    for(int i = 0; i < n; i++){
        tmp = point[i];
        for(int j = i + 1; j < n; j++){
            tmp1 = point[j];
            a = tmp.x - tmp1.x;
            b = tmp.y - tmp1.y;
            c = tmp1.y * tmp.x - tmp.y * tmp1.x;
            g = __gcd(a, b);
            mp[mt(a, b, c, g)]++;
        }
    }
    for(auto i : mp){
        ans += f[((int)sqrt(i.second * 8 + 1) + 1) >> 1][3];
    }
    return ans;
}
int main()
{
    int n;
    init();
    while(~scanf("%d", &n)){
        for(int i = 0; i < n; i++){
            scanf("%d %d", &point[i].x, &point[i].y);
        }
        printf("%d\n", count(n));
    }
    return 0;
}
```

4.5. 极角排序

```
struct point{
    double x, y;
    int i;
    point(){
    }
    point(double a, double b): x(a), y(b){
    }
    friend point operator - (const point &a, const point &b){
        return point(a.x - b.x, a.y - b.y);
```

```
    }
    friend bool operator == (const point &a, const point &b){
        return cmp(a.x - b.x) == 0 && cmp(a.y - b.y) == 0;
    }
    double operator *(const point &b)const
    {
        return x*b.x + y*b.y;
    }
}p[maxn], c;
double det(const point &a, const point &b)
{
    return a.x * b.y - a.y * b.x;
}
double dist(point a, point b)
{
    return sqrt((a-b)*(a-b));
}
bool cmp1(point a, point b)
{
    //c为参考点
    double tmp = det((a-c), (b-c));
    if(cmp(tmp) == 0)
        return dist(c,a) < dist(c,b);
    else if(cmp(tmp) < 0)return false;
    else return true;
}
```

4.6 最小球覆盖

模拟退火

```
#include<cstdio>
#include<cmath>
#include<algorithm>
using namespace std;
const double eps=1e-3;
int n;
struct Point{
    double x;
    double y;
    double z;
    Point(double _x=0.0,double _y=0.0,double _z=0.0)
{x=_x;y=_y;z=_z;}
};
Point points[105];
inline double dis(Point a,Point b)
{
    double norm=(a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y)+(a.z-b.z)*
(a.z-b.z);
    return sqrt(norm);
```

```
}
double solve()
{
    double T=1000.0; //初始温度
    double rate=0.99999; //温度下降系数
    Point ans_p;    //初始圆心
    int cur;
    Point max_p; //距离圆心最远的点
    double ans=1e99;
    while(T>eps) //模拟降温
    {
        double max_dis=0.0; //选定圆心到最远点距离
        for(int i=1;i<=n;++i)
        { //最远的点即points[cur]
            if(dis(ans_p,points[i])>max_dis){
                max_dis=dis(ans_p,points[i]);
                cur=i;
            }
        }
        ans=min(ans,max_dis);
        ans_p.x+=(points[cur].x-ans_p.x)*(T/1000.0);
        ans_p.y+=(points[cur].y-ans_p.y)*(T/1000.0);
        ans_p.z+=(points[cur].z-ans_p.z)*(T/1000.0);
        T*=rate;
    }
    return ans;
}
int main()
{
    scanf("%d", &n);

        for(int i=1;i<=n;++i)
        {

 scanf("%lf%lf%lf",&points[i].x,&points[i].y,&points[i].z);
        }
        printf("%.15f\n",solve());

    return 0;
}
```

三分套三分套三分

```
/*
    三分求球的最小覆盖
*/
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cmath>
```

```
using namespace std;
typedef long long ll;
const ll mod=9999973;
const double eps=1e-4;
int n;
double x[105],y[105],z[105];
double dis3(double a,double b,double c){
    double ans=0;
    for(int i=1;i<=n;i++){
        ans=max(ans,(x[i]-a)*(x[i]-a)+(y[i]-b)*(y[i]-b)+(z[i]-c)*
(z[i]-c));
    }
    return ans;
}
double dis2(double a,double b){
    double l=-100000;
    double r=100000;
    double ans=0;
    while(r-l>=eps){
        double rmid=(r+l)/2;
        double lmid=(l+rmid)/2;
        if(dis3(a,b,lmid)<dis3(a,b,rmid)){
            r=rmid;
        }
        else l=lmid;
    }
    return dis3(a,b,l);
}
double dis(double a){
    double l=-100000;
    double r=100000;
    while(r-l>=eps){
        double rmid=(r+l)/2;
        double lmid=(l+rmid)/2;
        if(dis2(a,lmid)<dis2(a,rmid)){
            r=rmid;
        }
        else l=lmid;
    }
    return dis2(a,l);
}
int main(){
   // int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)scanf("%lf%lf%lf",&x[i],&y[i],&z[i]);
    double l=-100000;
    double r=100000;
    while(r-l>=eps){
        double rmid=(r+l)/2;
        double lmid=(l+rmid)/2;
        if(dis(lmid)<dis(rmid)){
            r=rmid;
```

```
        }
        else l=lmid;
    }
    printf("%.8f\n",sqrt(dis(l)));
    return 0;
}
```

最小圆覆盖

使用的增量法
假设圆O是前i - 1个点的最小覆盖圆，加入第i个点，如果在园内或边上则什么都不做，否则，新得到的最小覆盖圆肯定经过第i个点。
然后以第i个点为基础(半径为0)，重复以上过程一次加入第j个点，若第j个点在圆外，则最小覆盖圆必经过第j个点。
重复以上步骤。(因为最多需要三个点来确定这个最小覆盖圆，所以重复三次)
遍历完所有点之后，所得到的圆就是覆盖所有点的最小圆。

时间复杂度O(n)，空间复杂度O(n)。

```
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>

using namespace std;

int n;
double r;

struct point {
  double x, y;
} p[100005], o;

inline double sqr(double x) { return x * x; }

inline double dis(point a, point b) {
  return sqrt(sqr(a.x - b.x) + sqr(a.y - b.y));
}

inline bool cmp(double a, double b) { return fabs(a - b) < 1e-8; }

point geto(point a, point b, point c) {
  double a1, a2, b1, b2, c1, c2;
  point ans;
  a1 = 2 * (b.x - a.x), b1 = 2 * (b.y - a.y),
  c1 = sqr(b.x) - sqr(a.x) + sqr(b.y) - sqr(a.y);
  a2 = 2 * (c.x - a.x), b2 = 2 * (c.y - a.y),
  c2 = sqr(c.x) - sqr(a.x) + sqr(c.y) - sqr(a.y);
  if (cmp(a1, 0)) {
    ans.y = c1 / b1;
    ans.x = (c2 - ans.y * b2) / a2;
```

```
  } else if (cmp(b1, 0)) {
    ans.x = c1 / a1;
    ans.y = (c2 - ans.x * a2) / b2;
  } else {
    ans.x = (c2 * b1 - c1 * b2) / (a2 * b1 - a1 * b2);
    ans.y = (c2 * a1 - c1 * a2) / (b2 * a1 - b1 * a2);
  }
  return ans;
}

int main() {
  scanf("%d", &n);
  for (int i = 1; i <= n; i++) scanf("%lf%lf", &p[i].x, &p[i].y);
  for (int i = 1; i <= n; i++) swap(p[rand() % n + 1], p[rand() % n
+ 1]);
  o = p[1];
  for (int i = 1; i <= n; i++) {
    if (dis(o, p[i]) < r || cmp(dis(o, p[i]), r)) continue;
    o.x = (p[i].x + p[1].x) / 2;
    o.y = (p[i].y + p[1].y) / 2;
    r = dis(p[i], p[1]) / 2;
    for (int j = 2; j < i; j++) {
      if (dis(o, p[j]) < r || cmp(dis(o, p[j]), r)) continue;
      o.x = (p[i].x + p[j].x) / 2;
      o.y = (p[i].y + p[j].y) / 2;
      r = dis(p[i], p[j]) / 2;
      for (int k = 1; k < j; k++) {
        if (dis(o, p[k]) < r || cmp(dis(o, p[k]), r)) continue;
        o = geto(p[i], p[j], p[k]);
        r = dis(o, p[i]);
      }
    }
  }
  printf("%.10lf\n%.10lf %.10lf", r, o.x, o.y);
  return 0;
}
```

4.7 二维的基本模板和凸包

```
int sgn(double x) {
    if (fabs(x) < eps) return 0;
    if (x < 0) return -1;
    else return 1;
}
int n;
struct Point
{
    double x, y;
    Point(){}
```

```
    Point(double _x, double _y) {
        x = _x;
        y = _y;
    }
    void input() {
        scanf("%lf%lf", &x, &y);
    }
    void output() {
        printf("%.2f %.2f\n", x, y);
    }
    bool operator == (Point b) const {
        return sgn(x - b.x) == 0 && sgn(y - b.y) == 0;
    }
    bool operator < (Point b) const {
        return sgn(x - b.x) == 0 ? sgn(y - b.y) < 0 : x < b.x;
    }
    Point operator - (const Point &b) const {
        return Point(x - b.x, y - b.y);
    }
    // 叉积
    double operator ^ (const Point &b) const {
        return x * b.y - y * b.x;
    }
    // 点积
    double operator * (const Point &b) const {
        return x * b.x + y * b.y;
    }
    // 返回长度
    double len() {
        return hypot(x, y); // 库函数
    }
    // 返回长度平方
    double len2() {
        return x * x + y * y;
    }
    // 返回两点距离
    double distance(Point p) {
        return hypot(x - p.x, y - p.y);
    }
    Point operator + (const Point &b) const {
        return Point(x + b.x, y + b.y);
    }
    Point operator * (const double &k) const {
        return Point(x * k, y * k);
    }
    Point operator / (const double &k) const {
        return Point(x / k, y / k);
    }
    // 计算 pa 和 pb 的夹角
    double rad(Point a, Point b) {
        Point p = *this;
```

```
        return fabs(atan2(fabs((a - p) ^ (b - p)), (a - p) ^ (b -
p)));
    }
    // 化为长度为 r 的向量
    Point trunc(double r) {
        double l = len();
        if (!sgn(l)) return *this;
        r /= l;
        return Point(x * r, y * r);
    }
    // 逆时针旋转90度
    Point rotleft() {
        return Point(-y, x);
    }
    // 顺时针旋转90度
    Point rotright() {
        return Point(y, -x);
    }
    // 绕着 p 点逆时针旋转 angle
    Point rotate(Point p, double angle) {
        Point v = (*this) - p;
        double c = cos(angle), s = sin(angle);
        return Point(p.x + v.x * c - v.y * s, p.y + v.x * s + v.y *
c);
    }
};
struct Line
{
    Point s, e;
    Line() {}
    Line(Point _s, Point _e) {
        s = _s;
        e = _e;
    }
};
struct polygon
{
    int n;
    Point p[MAXN];
    Line l[MAXN];
    void input(int _n) {
        n = _n;
        for (int i = 0; i < n; i++) {
            p[i].input();
        }
    }
    void add(Point q) {
        p[n++] = q;
    }
    void getline() {
        for (int i = 0; i < n; i++) {
            l[i] = Line(p[i], p[(i + 1) % n]);
```

```
        }
    }
    struct cmp
    {
        Point p;
        cmp(const Point &p0) {
            p = p0;
        }
        bool operator () (const Point &aa, const Point &bb) {
            Point a = aa, b = bb;
            int d = sgn((a - p) ^ (b - p));
            if (d == 0) {
                return sgn(a.distance(p) - b.distance(p)) < 0;
            }
            return d > 0;
        }
    };
    double getcircumference() { // 周长
        double sum = 0;
        for (int i = 0; i < n; i++) {
            sum += p[i].distance(p[(i + 1) % n]);
        }
        return sum;
    }
    double getarea() { // 多边形面积
        double sum = 0;
        for (int i = 0; i < n; i++) {
            sum += (p[i] ^ p[(i + 1) % n]);
        }
        return fabs(sum);
    }
};
Point p[MAXN];
void Andrew(polygon &convex) { // 凸包
    sort(p, p + n);
    convex.n = n;
    for (int i = 0; i < min(n, 2); i++) {
        convex.p[i] = p[i];
    }
    if (convex.n == 2 && (convex.p[0] == convex.p[1])) convex.n--;
    if (n <= 2) return;
    int &top = convex.n;
    top = 1;
    for (int i = 2; i < n; i++) {
        while (top && sgn((convex.p[top] - p[i]) ^ (convex.p[top -
1] - p[i])) <= 0) {
            top--;
        }
        convex.p[++top] = p[i];
    }
    int tmp = top;
    convex.p[++top] = p[n - 2];
```

```
    for (int i = n - 3; i >= 0; i--) {
        while (top != tmp && sgn((convex.p[top] - p[i]) ^
(convex.p[top - 1] - p[i])) <= 0){
            top--;
        }
        convex.p[++top] = p[i];
    }
    if (convex.n == 2 && (convex.p[0] == convex.p[1])) {
        convex.n--;
    }
}
```

圆的相关模板

```
#include <bits/stdc++.h>
using namespace std;
const double PI = acos(-1);
const double eps = 1e-7;
struct Point {
    double x, y;
    Point(double x = 0, double y = 0):x(x), y(y) {};
};
struct Circle {
    Point c;
    double r;
    Circle(Point c, double r):c(c), r(r) {};
    Point getPoint(double a) {
        return Point(c.x + cos(a) * r, c.y + sin(a) * r);
    }
};
int dcmp(double x) {
    if (fabs(x) < eps) return 0;
    else return x < 0 ? -1 : 1;
}
int getTangents(Circle A, Circle B, Point* a, Point* b) { // 求两个
圆的切线数量和切点坐标
    int cnt = 0;
    if (A.r < B.r) {
    swap(A, B);
    swap(a, b);
}
    int d2 = (A.c.x - B.c.x) * (A.c.x - B.c.x) + (A.c.y - B.c.y) *
(A.c.y - B.c.y);
    int rdiff = A.r - B.r;
    int rsum = A.r + B.r;
    if (d2 < rdiff * rdiff) return 0; // 内含
    double base = atan2(B.c.y - A.c.y, B.c.x - A.c.x);
    if (d2 == 0 && A.r == B.r) return -1; // 重合，无数条
    if (d2 == rdiff * rdiff) { // 内切
        a[cnt] = A.getPoint(base);
        b[cnt] = B.getPoint(base);
```

```
        cnt++;
        return 1;
    }
    // 有外公切线
    double ang = acos((A.r - B.r) / sqrt(d2));
    a[cnt] = A.getPoint(base + ang);
    b[cnt] = B.getPoint(base + ang);
    cnt++;
    a[cnt] = A.getPoint(base - ang);
    b[cnt] = B.getPoint(base - ang);
    cnt++;
    if (d2 == rsum * rsum) { // 一条内公切线
        a[cnt] = A.getPoint(base);
        b[cnt] = B.getPoint(PI + base);
        cnt++;
    }
    else if (d2 > rsum * rsum) { // 两条
        double ang = acos((A.r + B.r) / sqrt(d2));
        a[cnt] = A.getPoint(base + ang);
        b[cnt] = B.getPoint(PI + base + ang);
        cnt++;
        a[cnt] = A.getPoint(base - ang);
        b[cnt] = B.getPoint(PI + base - ang);
        cnt++;
    }
    return cnt;
}
int main() {
    Point a[100], b[100], c(0, 0), d(0, 4);
    Circle A(c, 2), B(d, 1);
    cout << getTangents(A, B, a, b) << endl;
    // cout << dcmp(a[0].x) << " " << a[0].y << endl;
    // cout << dcmp(b[0].x) << " " << b[0].y << endl;
    for (int i = 0; i < 4; i++) {
        cout << dcmp(a[i].x) << " " << a[i].y << endl; // 在各个圆上
的切点坐标
        cout << dcmp(b[i].x) << " " << b[i].y << endl;
    }
}
```

5. 博弈

5.1. bash博弈

```
/*一堆n个物品，两个人轮流从中取出1~m个，
最后取光者胜（不能继续取的人输）。
同余定理：n=k*(m+1)+r，先者拿走r个，那么后者无论拿走多少个
先者只要的数目使和为m+1，那么先手必赢。反之若n=k*(m+1)，那么先手无论怎样都会
输。*/
if (n % (m + 1))  return false;
else  return true;
```

5.2. 威佐夫博弈

```
/*有两堆各若干物品，两个人轮流从任意一堆中至少
取出一个或者从两堆中取出同样多的物品，
规定每次至少取一个，至多不限，最后取光者胜。
这里的必输局势：（0，0）、（1，2）、（3，5）、
（4，7）、（6，10）、（8，13）、（9，15）、
（11，18）、（12，20）。从这些必输局势可以发现，
每组的第一个是前面没有出现的最小正整数，
ak=[k*(1+√5)/2]，bk=ak+k，k=0,1,2,3...
所以，先求出差值，差值*黄金分割比 == 最小值的话后手赢，否者先手赢。*/
double r = (sqrt(5) + 1) / 2;
int d = abs(a - b) * r;
if (d != min(a, b))  return true;
else  false;
```

5.3. 斐波那契博弈

```
/*一堆石子有n个，两人轮流取，先取者第一次可以去任意多个，
但是不能取完，以后每次取的石子数不能超过上次取子数的2倍。取完者胜。
同样是一个规律：先手胜当且仅当n不是斐波那契数。*/
f[0] = f[1] = 1;
for (int i = 0; f[i - 1] < n; i++)
{
    f[i] = f[i - 1] + f[i - 2];
    if (f[i] == n)  return true;
}
return false;
```

5.4. nim博弈

```
/*有n堆物品，两人轮流取，每次取某堆中不少于1个，最后取完者胜。*/
int res = 0;
for (int i = 1; i <= n; i++)
    res ^= arr[i];
if (res)  return true;
else  return false;
```

5.5. 摸鱼&&摸鱼plus

```
//对5取模，1或4的时候先手必胜
#include <cstdio>

const int maxn = 1005;
bool dp[maxn][maxn];
bool vis[maxn][maxn];

bool call(int i, int j)
{
    if(vis[i][j]){
        return dp[i][j];
    }
    vis[i][j] = true;
    bool tmp = false;
    for(int k = j - 1; k >= 2 * j - i - 58 && k >= 0; k--){
        tmp |= call(j, k);
    }
    dp[i][j] = !tmp;
    return dp[i][j];
}
void init()
{
    for(int i = 1; i < maxn; i++){
        dp[i][0] = vis[i][0] = true;
        for(int j = 1; j < maxn; j++){
            dp[i][j] = vis[i][j] = false;
        }
    }
}
int main()
{
    int t, n;
    while(~scanf("%d", &t)){
        init();
        while(t--){
            //scanf("%d", &n);
            int p = 0;
            for(int i = 1; i < 1000; i++){
                call(i, i - 1) ? printf("%d hyy\n", i - p) , p = i:
printf("");
            }
            getchar();

        }
    }
    return 0;
}


//plus，找规律
```

```cpp
#include <cstdio>
#include <cmath>
typedef long long LL;
int log2(LL m)
{
    return (int)(log(m) / log(2));
}
bool call(LL n, LL m)
{
    int x = log2(m);
    LL t = 1ll << x;
    //printf("%d\n", x);
    int p = 0;
    LL tp = t;
    while(m > (t + 1)){
        tp >>= 1;
        t += tp;
        p++;
    }
    //printf("%d\n", p);
    LL mod = (1ll << (x + 1)) - 2 + (!p ? m : (m << 1) + (1ll << (x
- p + 1)) - 2);
    LL tmp = n % mod, tmp1 = 1 + m;
    //printf("%lld\n", mod);
    if(tmp == 1){
        return true;
    }
    for(int i = 0; i < x; i++){
        //printf("%lld\n", tmp1);
        if(tmp == tmp1){
            return true;
        }
        tmp1 += (1ll << (i + 1));
    }
    //printf("h%lld %lld\n", tmp, tmp1);
    if(tmp1 == tmp || tmp1 + m == tmp){
        return true;
    }
    return false;
}

int main()
{
    int t;
    LL n, m;
    //freopen("摸鱼plus随机.in","r", stdin);
    //freopen("摸鱼plus随机.out","w", stdout);
    while(~scanf("%d", &t)){
        while(t--){
            scanf("%lld %lld", &n, &m);
            printf("%s\n", call(n, m + 2) ? "hyy" : "wqy");
```

```
        }
    }
    return 0;
}
```

5.6. SG函数

```
//如果只涉及一个nim游戏，那么最后判断时不用与其他情况进行异或，如果有多个min游戏，也就是多个博弈图是，需要异或判断
int SG(int x) {
    if (sg[x] != -1) return sg[x];
    //memset(vis, 0, sizeof vis);
    vector<int> vis(100 + 10, 0);
    //set<int> con
    for (int i = 0; i < k; i++) {
        if (s[i] <= x) {
            vis[SG(x - s[i])] = 1;
        }
    }
    for (int i = 0; ; i++) {
        if (!vis[i]) { // 如果使用set，在这里直接!con.count(i)即可
            sg[x] = i;
            return i;
        }
    }
}
```

6. 图论

7. 黑科技

7.1. 高精

```
struct Wint:vector<int>
{
    Wint(int n=0)
    {
        push_back(n);
        check();
    }
    Wint& check()
    {
        while(!empty()&&!back())pop_back();
        if(empty())return *this;
        for(int i=1; i<size(); ++i)
        {
```

```
            (*this)[i]+=(*this)[i-1]/10;
            (*this)[i-1]%=10;
        }
        while(back()>=10)
        {
            push_back(back()/10);
            (*this)[size()-2]%=10;
        }
        return *this;
    }
};
istream& operator>>(istream &is,Wint &n)
{
    string s;
    is>>s;
    n.clear();
    for(int i=s.size()-1; i>=0; --i)n.push_back(s[i]-'0');
    return is;
}
ostream& operator<<(ostream &os,const Wint &n)
{
    if(n.empty())os<<0;
    for(int i=n.size()-1; i>=0; --i)os<<n[i];
    return os;
}
bool operator!=(const Wint &a,const Wint &b)
{
    if(a.size()!=b.size())return 1;
    for(int i=a.size()-1; i>=0; --i)
        if(a[i]!=b[i])return 1;
    return 0;
}
bool operator==(const Wint &a,const Wint &b)
{
    return !(a!=b);
}
bool operator<(const Wint &a,const Wint &b)
{
    if(a.size()!=b.size())return a.size()<b.size();
    for(int i=a.size()-1; i>=0; --i)
        if(a[i]!=b[i])return a[i]<b[i];
    return 0;
}
bool operator>(const Wint &a,const Wint &b)
{
    return b<a;
}
bool operator<=(const Wint &a,const Wint &b)
{
    return !(a>b);
}
bool operator>=(const Wint &a,const Wint &b)
```

```
{
    return !(a<b);
}
Wint& operator+=(Wint &a,const Wint &b)
{
    if(a.size()<b.size())a.resize(b.size());
    for(int i=0; i!=b.size(); ++i)a[i]+=b[i];
    return a.check();
}
Wint operator+(Wint a,const Wint &b)
{
    return a+=b;
}
Wint& operator-=(Wint &a,Wint b)
{
    if(a<b)swap(a,b);
    for(int i=0; i!=b.size(); a[i]-=b[i],++i)
        if(a[i]<b[i])
        {
            int j=i+1;
            while(!a[j])++j;
            while(j>i)
            {
                --a[j];
                a[--j]+=10;
            }
        }
    return a.check();
}
Wint operator-(Wint a,const Wint &b)
{
    return a-=b;
}
Wint operator*(const Wint &a,const Wint &b)
{
    Wint n;
    n.assign(a.size()+b.size()-1,0);
    for(int i=0; i!=a.size(); ++i)
        for(int j=0; j!=b.size(); ++j)
            n[i+j]+=a[i]*b[j];
    return n.check();
}
Wint& operator*=(Wint &a,const Wint &b)
{
    return a=a*b;
}
Wint divmod(Wint &a,const Wint &b)
{
    Wint ans;
    for(int t=a.size()-b.size(); a>=b; --t)
    {
        Wint d;
```

```
        d.assign(t+1,0);
        d.back()=1;
        Wint c=b*d;
        while(a>=c)
        {
            a-=c;
            ans+=d;
        }
    }
    return ans;
}
Wint operator/(Wint a,const Wint &b)
{
    return divmod(a,b);
}
Wint& operator/=(Wint &a,const Wint &b)
{
    return a=a/b;
}
Wint& operator%=(Wint &a,const Wint &b)
{
    divmod(a,b);
    return a;
}
Wint operator%(Wint a,const Wint &b)
{
    return a%=b;
}
```

8. tips

> *唯一分解定理：n的因子个数$Num(n) = (1 + a_1) * (1 + a_2) * \ldots * (1 + a_n)$*
> *a > b 且 gcd(a, b) == 1, 有$(gcd(a^n - b^n, a^m - b^m)) = a^{gcd(n,m)} - b^{gcd(n,m)}$*
> *$\mu(n)$——莫比乌斯函数*
> *$\varphi(n)$——欧拉函数。表示不大于n且与n互质的正整数个数，十分常见的数论函数。*
> *用数学式子表示即：$\varphi(n) = \sum_{i=1}^{n}[gcd(n, i) == 1]$*
> *$d(n)$——约数个数。表示n的约数的个数。用式子表示为:$d(n) = \sum_{d|n} 1$,也可以写作:*
> *$d(n) = \sum_{d=1}^{n}[d|n]$*
> *$\sigma(n)$——约数和函数。即n的各个约数之和。表示为$\sigma(n) = \sum_{d|n} d = \sum_{d=1}^{n}[d|n] * d$*
> *img*
> *img*
> *img*
> *img*
> *img*
> *img*
> *img*
> *img*
> *img*
> *img*
>
> - *__builtin_popcount(n)计算二进制有多少个1*

- $d(ij) = \sum_{x|i} \sum_{y|j} [gcd(x, y) = 1]$
- $\sum_{i=1}^{n} \lfloor \frac{n}{i} \rfloor = 2 * \sum_{i=1}^{\sqrt{n}} \lfloor \frac{n}{i} \rfloor - \lfloor \sqrt{n} \rfloor^2$