ACM-ICPC Standard Code Library

李琦

2014/10/15

Just follow our heart

目录

数学

GCD

```
int gcd(int a,int b){
    if ( b==0 )return a;
    return gcd(b , a%b);
}
```

EXTGCD

```
int extgcd(int a,int b,int & x,int & y)
{
    int d = a;
    if ( b!=0 ){
        d = extgcd(b,a%b,y,x);
        y -= (a/b)*x;
    }else{
        x=1;y=0;
    }
    return d;
}
```

模线性方程

$$\mathrm{ax} + \mathrm{by} = \mathrm{gcd}(a,b) = d$$

通解为：$x = x1 + \left(\frac{b}{d}\right) * t$，$y = y1 - \left(\frac{a}{d}\right) * t$

模线性方程合并：$a_1 * x + a_2 * y == b2 - b1$

$c1 == c\ (mod\ lcm(a1,a2))$(c 前面方程求出的最小特解)

```
int main()
{
    while(~scanf("%d",&n)){
        LL a1,b1;
        bool ok = 1;
        scanf("%lld%lld",&a1,&b1);
        if(n == 1){
            printf("%lld\n",a1+b1);
```

```
            continue;
        }
        for(int i = 1 ; i < n ; i ++){
            LL a2,b2;
            scanf("%lld%lld",&a2,&b2);
            LL gg = gcd(a1,a2);
            if((b2 - b1) % gg != 0)ok = 0;
            if(ok){
                LL d = ext_gcd(a1,a2);
                x *= (b2-b1)/d;
                x = x - (x*d/a2)*(a2/d);
                if(x < 0) x += a2/d;
                LL c = a1 * x + b1;
                b1 = c;
                a1 = a1 / d * a2;
            }
        }
        if(ok)
            printf("%lld\n",b1);
        else
            printf("-1\n");
    }
    return 0;
}
```

逆元

拓展欧几里得求逆元

```
int mod_inverse(int a,int m)
{
    int x,y;
    extgcd(a,m,x,y);
    return ( m + x % m ) % m;
}
```

线性求逆元

```
int inv(int a) {
    //return fpow(a, MOD-2, MOD);
    return a == 1 ? 1 : (long long)(MOD - MOD / a) * inv(MOD % a) % MOD;
}
```

Pell 方程

假设对于 Pell 方程 x^2 - D*y^2 = 1

sqrt(D)=[a0,a1,...an,b1,b2...b(m-1),bm,b1,b2,...]

即以[b1,b2...bm]为循环节出现

p/q=[a0,a1,...an,b1,b2,....b(m-1)]

则一定有：bm-1 = 2*b1

且若 m 为偶数：x = p，y = q

若 m 为奇数：x = 2*p^2 + 1，y = p*q

如果我们求出 Pell 方程的最小正整数解后，就可以根据递推式求出所有的解。

$$x_n = x_{n-1}x_1 + dy_{n-1}y_1$$
$$y_n = x_{n-1}y_1 + y_{n-1}x_1$$

则根据上式我们可以构造矩阵，然后就可以快速幂了。

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} x_1 & dy_1 \\ y_1 & x_1 \end{bmatrix}^{k-1} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

这样就可以求出第 k 大的解。

素数测试

```
int power(int a, int e, int m){
   if (e == 0) return 1;
   if (e == 1) return a % m;
   int t = power(a, e/2, m);
   if (e % 2 == 1) return (t * t * a) % m;
   return (t * t) % m;
}
//int 范围内可测
bool isprime(int x){
   const int a[4] = {2, 3, 5, 7};
   for (int i = 0; i < 4; i++)
      if (power(a[i], x-1, x) != 1) return false;
   return true;
```

```
}
```

Millar-rabin

```
typedef long long LL;
LL mul(LL a,LL b,LL mod)
{
    LL ans=0;
    while (b){
        if (b&1) ans=(ans+a)%mod;
        a=(a<<1)%mod;
        b>>=1;
    }
    return ans;
}

bool test(LL n, LL b) {
    LL m = n - 1;
    LL counter = 0;
    while (~m & 1) {
        m >>= 1;
        counter ++;
    }
    LL ret = pow_mod(b, m, n);
    if (ret == 1 || ret == n - 1) {
        return true;
    }
    counter --;
    while (counter >= 0) {
        ret = multiply_mod(ret, ret, n);
        if (ret == n - 1) {
            return true;
        }
        counter --;
    }
    return false;
}

const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};

bool is_prime(LL n) {
    if (n < 2) {
        return false;
```

```
    }
    if (n < 4) {
        return true;
    }
    if (n == 3215031751LL) {
        return false;
    }
    for (int i = 0; i < 12 && BASE[i] < n; ++ i) {
        if (!test(n, BASE[i])) {
            return false;
        }
    }
    return true;
}
```

Pollar-Rho

```
typedef long long LL;

LL pollard_rho(LL n, LL seed) {
    LL x, y, head = 1, tail = 2;
    x = y = rand() % (n - 1) + 1;
    while (true) {
        x = multiply_mod(x, x, n);
        x = add_mod(x, seed, n);
        if (x == y) {
            return n;
        }
        LL d = gcd(abs(x - y), n);
        if (1 < d && d < n) {
            return d;
        }
        head ++;
        if (head == tail) {
            y = x;
            tail <<= 1;
        }
    }
}

vector <LL> divisors;

void factorize(LL n) {
```

```
    if (n > 1) {
        if (is_prime(n)) {
            divisors.push_back(n);
        } else {
            LL d = n;
            while (d >= n) {
                d = pollard_rho(n, rand() % (n - 1) + 1);
            }
            factorize(n / d);
            factorize(d);
        }
    }
}
```

欧拉函数

线性筛

```
void get_phi(int N)
{
    CLR(check,0);
    fai[1] = 1;
    int tot = 0;
    for(int i=2;i<N;i++){
        if(!check[i]){
            prime[tot ++] = i;
            phi[i] = i-1;
        }
        for(int j=0;j<tot;j++){
            if(i * prime[j] > N)break;
            check[i * prime[j]] = true;
            if(i % prime[j] == 0){
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            }else{
                phi[i * prime[j]] = phi[i] * (prime[j] - 1);
            }
        }
    }
}
```

单值

```
int phi(int x)
{
    int num = x;
    for(int i = 2 ; i * i <= x ; i ++){
        if(x % i == 0){
            num = (num / i) * (i-1);
            while(x % i == 0)x /= i;
        }
    }
    if(x != 1)num = (num/x) * (x-1);
    return num;
}
```

性质

1. 若(N%a==0 && (N/a)%a==0) 则有:E(N)=E(N/a)*a;
2. 若(N%a==0 && (N/a)%a!=0) 则有:E(N)=E(N/a)*(a-1);
3. 若N>2, 欧拉函数E(N)必定是偶数若gcd(a,b) = 1,则有E(a * b) = E(a) * E(b)
4. 若N>1,不大于N且与N互素的所有正整数的和是1/2 * N * E(N)
5. 因子和:若 k=p1^a1*p2^a2...*pi^ai
 F(k) = (p1^0+...+p1^a1)*(p2^0+...+p2^a2)*...*(pi^0 + ... + pi^ai)

$\sum_{d|n} \mu(d) = [n == 1]$

$\sum_{d|n} \varphi(d) = n$

6.
7. A^x = A^(x % Phi(C) + Phi(C)) (mod C) 其中phi是欧拉函数

莫比乌斯函数

$$f(n) = \sum_{d|n} \mu(d) F(n/d)$$

1、

f(n) = sigma(d|n, g(d))

g(n) = sigma(d|n, mu(d)*f(n/d))

2、

f(n) = sigma(n|d, g(d))

g(n) = sigma(n|d, mu(d/n)*f(d))

```
bool check[N];
int mu[N];
int prime[N];
void get_mu()
{
    CLR(check,0);
    mu[1] = 1;
    int tot = 0;
    for(int i=2;i<N;i++){
        if(!check[i]){
            prime[tot ++] = i;
            mu[i] = -1;
        }
        for(int j=0;j<tot;j++){
            if(i * prime[j] >= N)break;
            check[i * prime[j]] = true;
            if(i % prime[j] == 0){
                mu[i * prime[j]] = 0;
                break;
            }else{
                mu[i * prime[j]] = -mu[i];
            }
        }
    }
}
```

HDU4746 Mobius

题意：给 a 在 1-n，b 在 1-m 之间，问有多少对 a，b 使得 gcd(a,b)的因子数少于 p

```
typedef long long ll;
#define FILL(x) memset(x,0,sizeof(x))
#define CLR(a,b) memset(a,b,sizeof(a))
const int maxn = 500000+20;
bool check[maxn];
int mu[maxn],prime[maxn],num[maxn],sum[maxn];
int G[maxn][30];
```

```
int n,m,p;
void Mobius(int N)
{
    FILL(check);
    mu[1] = 1;
    num[1] = 0;
    int tot = 0;
    for(int i=2;i<=N;i++){
        if(!check[i]){
            prime[tot ++] = i;
            mu[i] = -1;
            num[i] = 1;
        }
        for(int j=0;j<tot;j++){
            if(i * prime[j] > N)break;
            check[i * prime[j]] = true;
            num[i * prime[j]] = num[i] + 1;
            if(i % prime[j] == 0){
                mu[i * prime[j]] = 0;
                break;
            }else{
                mu[i * prime[j]] = -mu[i];
            }
        }
    }
}
void init()
{
    FILL(G);
    for(int i=1;i<maxn;i++){
        for(int j=i;j<maxn;j+=i){
            G[j][num[i]] += mu[j/i];
        }
    }
    for(int i=1;i<maxn;i++)
        for(int j=0;j<30;j++){
            G[i][j] += G[i-1][j];
    }
    for(int i=0;i<maxn;i++)
        for(int j=1;j<30;j++){
            G[i][j] += G[i][j-1];
    }
}
void solve()
{
```

```
    if(n>m)swap(n,m);
    if(p>=19){
        ll ans = (ll)n * m;
        printf("%I64d\n",ans);
        return;
    }
    ll ans = 0;
    for(int i=1,last=i;i<=n;i=last+1){
        last = min(n/(n/i),m/(m/i));
        ans += (ll)(G[last][p] - G[i-1][p])*(n/i)*(m/i); //分块加速
    }
    printf("%I64d\n",ans);

}
```

Lucas 定理

```
int inv(int a) {
    //return fpow(a, MOD-2, MOD);
    return a == 1 ? 1 : (long long)(MOD - MOD / a) * inv(MOD % a) % MOD;
}
LL C(LL n,LL m)
{
    if(m < 0)return 0;
    if(n < m)return 0;
    return fac[n] * inv(fac[m]*fac[n-m] % MOD) % MOD;
}
LL n,m;
LL lucas(LL n,LL m,LL p)
{
    LL ret = 1;
    while(n && m){
        LL a = n % p,b = m % p;
        if(a < b)return 0;
        ret = ret * C(a,b) % p;
        n /= p;
        m /= p;
    }
    return ret;
}
```

n! mod p 的性质

```
int fact[MAX_P];//n<p 时候的表，利用周期性

void init()//预处理，o(p)
{
    fact[0] = 1;
    for(int i=1;i<MAX_P;i++){
        fact[i] = fact[i-1] * i % p;
    }
}

// 分解 n!=a*p^e，返回 a mod p，O(log p (n))
int mod_fact(int n,int p,int &e)
{
    e = 0;
    if(n==0)return 1;

    //计算 p 的倍数的部分
    int res = mod_fact(n / p , p , e);
    e += n/p;

    //由于(p-1)!=-1,因此(p-1)!^(n/p)只需要知道 n/p 的奇偶就可以计算了
    if(n/p %2 != 0)return res * (p - fact[n%p]) % p;
    return res * fact[n%p] %p;
}
```

拉格朗日插值法

```
double lagrangePolynomial(double x, double xs[],double ys[],int n)
{
    double ans = 0;
    for(int i=0;i<n;i++){
        double f = 1;
        for(int j=0;j<n;j++){
            if(i == j) continue;
                f=f*(x-xs[j])/(xs[i]-xs[j]);
        }
        ans += f * ys[i];
    }
    return ans;
}
```

牛顿迭代

```
x' = x - f(x) / f'(x)
```

高斯消元

```
int Gauss()
{
    int i,j,k,col,max_r;
    for(k=0,col=0;k<equ&&col<var;k++,col++){
        max_r = k;
        for(i=k+1;i<equ;i++)
            if(fabs(a[i][col])>fabs(a[max_r][col]))
                max_r = i;
        if(fabs(a[max_r][col])<eps)return 0; //无解,有自由变元
        if(k != max_r){
            for(j=col;j<var;j++)
                swap(a[k][j],a[max_r][j]);
            swap(x[k],x[max_r]);
        }
        x[k]/=a[k][col];
        for(j=col+1;j<var;j++)a[k][j]/=a[k][col];
        a[k][col] = 1;
        for(i=0;i<equ;i++)
            if(i!=k){
                x[i] -= x[k]*a[i][k];
                for(j=col+1;j<var;j++)a[i][j]-=a[k][j]*a[i][col];
                a[i][col]=0;
            }
    }
    return 1;
}
```

FFT

```
#define L(x) (1 << (x))
const double PI = acos(-1.0);
const int N = 17, Maxn = L(N + 1);
double ax[Maxn], ay[Maxn], bx[Maxn], by[Maxn];
struct FFT
{
    private :
    int revv(int x, int bits) {
```

```
    int ret = 0;
    for (int i = 0; i < bits; i++) {
        ret <<= 1;
        ret |= x & 1;
        x >>= 1;
    }
    return ret;
}
void fft(double * a, double * b, int n, bool rev)
{
    int bits = 0;
    while (1 << bits < n) ++bits;
    for (int i = 0; i < n; i++)
    {
        int j = revv(i, bits);
        if (i < j)
        {
            double t = a[i];
            a[i] = a[j];
            a[j] = t;
            t = b[i];
            b[i] = b[j];
            b[j] = t;
        }
    }
    for (int len = 2; len <= n; len <<= 1)
    {
        int half = len >> 1;
        double wmx = cos(2 * PI / len);
        double wmy = sin(2 * PI / len);
        if (rev)
            wmy = -wmy;
        for (int i = 0; i < n; i += len)
        {
            double wx = 1;
            double wy = 0;
            for (int j = 0; j < half; j++)
            {
                double cx = a[i + j];
                double cy = b[i + j];
                double dx = a[i + j + half];
                double dy = b[i + j + half];
                double ex = dx * wx - dy * wy;
                double ey = dx * wy + dy * wx;
                a[i + j] = cx + ex;
```

```
                b[i + j] = cy + ey;
                a[i + j + half] = cx - ex;
                b[i + j + half] = cy - ey;
                double wnx = wx * wmx - wy * wmy;
                double wny = wx * wmy + wy * wmx;
                wx = wnx;
                wy = wny;
            }
        }
    }
    if (rev)
    {
        for (int i = 0; i < n; i++)
        {
            a[i] /= n;
            b[i] /= n;
        }
    }
}
public:
    int solve(int a[],int na,int b[],int nb,long long ans[])
    {
        int len = max(na, nb), ln;
        for(ln=0; L(ln)<len; ++ln);
        len=L(++ln);
        for (int i = 0; i < len ; ++i)
        {
            if (i >= na) ax[i] = 0, bx[i] =0;
            else ax[i] = a[i], bx[i] = 0;
        }
        fft(ax, ay, len, 0);
        for (int i = 0; i < len; ++i)
        {
            if (i >= nb) bx[i] = 0, by[i] = 0;
            else bx[i] = b[i], by[i] = 0;
        }
        fft(bx, by, len, 0);
        for (int i = 0; i < len; ++i)
        {
            double cx = ax[i] * bx[i] - ay[i] * by[i];
            double cy = ax[i] * by[i] + ay[i] * bx[i];
            ax[i] = cx, ay[i] = cy;
        }
        fft(ax, ay, len, 1);
        for (int i = 0; i < len; ++i)
```

```
        ans[i] = ax[i] + 0.01;
    return len;
}
int solve(int a[], int na, long long ans[])
{
    int len = na, ln;
    for(ln = 0; L(ln) < na; ++ln);
    len=L(++ln);
    for(int i = 0;i < len; ++i)
    {
        if (i >= na) ax[i] = 0, ay[i] = 0;
        else ax[i] = a[i], ay[i] = 0;
    }
    fft(ax, ay, len, 0);
    for(int i=0;i<len;++i)
    {
        double cx = ax[i] * ax[i] - ay[i] * ay[i];
        double cy = 2 * ax[i] * ay[i];
        ax[i] = cx, ay[i] = cy;
    }
    fft(ax, ay, len, 1);

    for(int i=0;i<len;++i)
        ans[i] = ax[i] + 0.5;
    return len;
}
};
```

中国剩余定理

```
typedef int typec;

typec CRT_2(typec a,typec x,typec b,typec y)
{
    typec xx,yy,tmp;
    tmp = extgcd(a,b,xx,yy);
    typec c = y - x;
    while (c<0)
        c += a;
    if (c%tmp!=0) return -1;
    xx *= c/tmp;
    yy *= c/tmp;
```

```
    typec t = yy/(a/tmp);
    while (yy-t*(a/tmp)>0)
        t++;
    while (yy-(t-1)*(a/tmp)<=0)
        t--;
    return (t*(a/tmp)-yy)*b+y;
}
typec CRT(typec a[],typec r[],int n)
{
    int i;
    typec m = a[0]/__gcd(a[0],a[1])*a[1];
    typec ans = CRT_2(a[0],r[0],a[1],r[1])%m;
    for (i=2;i<n&&ans!=-1;i++)
    {
        ans = CRT_2(m,ans,a[i],r[i]);
        m*=a[i]/__gcd(m,a[i]);
        ans%=m;
    }
    return ans;
}
```

积分

自适应 Simpson

```
double F(double x1)
{
    return 4 * sqrt(a*a-x1*x1) * sqrt(b*b-x1*x1);
}
double simpson(double a,double b)
{
    double c = a + (b-a)/2;
    return (F(a) + 4*F(c) + F(b))*(b-a)/6;
}
double asr(double a,double b,double eps,double A)
{
    double c = a + (b-a)/2;
    double L = simpson(a,c);
    double R = simpson(c,b);
    if(fabs(L+R-A) <= 15*eps)return L+R+(L+R-A)/15;
    return asr(a,c,eps/2,L) + asr(c,b,eps/2,R);
}
double asr(double a,double b,double eps)
```

```
{
    return asr(a,b,eps,simpson(a,b));
}
```

龙贝格积分

```
double romberg(double (*f)(double), double l, double r) {
    const int N = 20;
    double a[N][N], p[N];

    p[0] = 1;
    for (int i = 1; i < N; i++)
        p[i] = p[i - 1] * 4;

    a[0][0] = (f(l) + f(r)) / 2;
    for (int i = 1, n = 2; i < N; i++, n <<= 1) {
        a[i][0] = 0;
        for (int j = 1; j < n; j += 2)
            a[i][0] += f((r - l) * j / n + l);
        a[i][0] += a[i - 1][0] * (n / 2);
        a[i][0] /= n;
    }
    for (int j = 1; j < N; j++)
        for (int i = 0; i < N - j; i++)
            a[i][j] = (a[i + 1][j - 1] * p[j] - a[i][j - 1]) / (p[j] - 1);
    return a[0][N - 1] * (r - l);
}
```

Baby-Step-Giant-Step

A^n=B(mod C)，求 n

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

typedef long long LL;

#define MAXN 131071
struct HashNode { LL data, id, next; };
```

```
HashNode hash[MAXN<<1];
bool flag[MAXN<<1];
LL top;

void Insert ( LL a, LL b )
{
    LL k = b & MAXN;
    if ( flag[k] == false )
    {
        flag[k] = true;
        hash[k].next = -1;
        hash[k].id = a;
        hash[k].data = b;
        return;
    }
    while( hash[k].next != -1 )
    {
        if( hash[k].data == b ) return;
        k = hash[k].next;
    }
    if ( hash[k].data == b ) return;
    hash[k].next = ++top;
    hash[top].next = -1;
    hash[top].id = a;
    hash[top].data = b;
}

LL Find ( LL b )
{
    LL k = b & MAXN;
    if( flag[k] == false ) return -1;
    while ( k != -1 )
    {
        if( hash[k].data == b ) return hash[k].id;
        k = hash[k].next;
    }
    return -1;
}

LL BabyStep_GiantStep ( LL A, LL B, LL C )
{
    top = MAXN;  B %= C;
    LL tmp = 1, i;
    for ( i = 0; i <= 100; tmp = tmp * A % C, i++ )
        if ( tmp == B % C ) return i;
```

```
    LL D = 1, cnt = 0;
    while( (tmp = gcd(A,C)) !=1 )
    {
        if( B % tmp ) return -1;
        C /= tmp;
        B /= tmp;
        D = D * A / tmp % C;
        cnt++;
    }

    LL M = (LL)ceil(sqrt(C+0.0));
    for ( tmp = 1, i = 0; i <= M; tmp = tmp * A % C, i++ )
        Insert ( i, tmp );

    LL x, y, K = fpow( A, M, C );
    for ( i = 0; i <= M; i++ )
    {
        ext_gcd ( D, C, x, y ); // D * X = 1 ( mod C )
        tmp = ((B * x) % C + C) % C;
        if( (y = Find(tmp)) != -1 )
            return i * M + y + cnt;
        D = D * K % C;
    }
    return -1;
}

int main()
{
    LL A, B, C;
    while( scanf("%I64d%I64d%I64d",&A,&C,&B ) !=EOF )
    {
        if ( !A && !B && !C ) break;
        memset(flag,0,sizeof(flag));
        LL tmp = BabyStep_GiantStep ( A, B, C );
        if ( tmp == -1 )puts("No Solution");
        else printf("%I64d\n",tmp);
    }
    return 0;
}
```

公式

A^x mod C

$$A^x = A^{(x \bmod Phi(C) + Phi(C))} \bmod C \ (x \geq Phi(C))$$

组合数性质

1. C(n,k)=C(n−1,k)+C(n−1,k−1) 1<=k<=n−1
2. 1*C(n,1)+2*C(n,2)+...+n*C(n,n)=n*2^(n−1) (n>=1)
3. 通过对等式 (1+x)^n=sigma(C(n,k)*x^k) k: 0−>n 两边就微分，可以得到 sigma(k^p * C(n,k)) k: 1−>n 的和$\sum_{k=1}^{n} C(n,k)^2 = C(2n,n)$
4. C(r,0)+C(r+1,1)+...+C(r+k,k) = C(r+k+1,k)

5, C(0,k)+C(1,k)+...+C(n−1,k)+C(n,k)=C(n+1,k+1)

概率公式

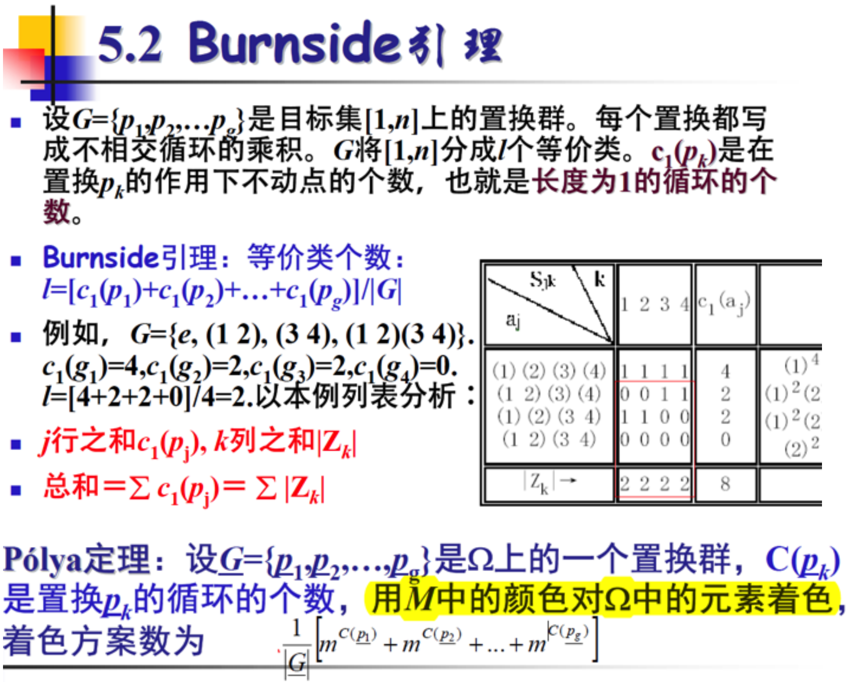
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(A_k|B) = \frac{P(B|A_k)P(A_k)}{\sum_j P(B|A_j)P(A_j)}$$

$$P((X|A) = x) = \frac{P(X = x, A)}{P(A)}$$

$$P(A) = \sum_k P(A|B_k)P(B_k)$$

Polya



$G=\{(v_1)(v_2)(v_3),(v_1v_2v_3),(v_3v_2v_1),(v_1)(v_2v_3),(v_2)(v_1v_3),(v_3)(v_1v_2)\}$

故不同的方案数为

$m=1/6*[3^3+2*3+3*3^2]=10$

博弈游戏

威佐夫博奕

//有两堆各若干个物品,两个人轮流从某一堆或同时从两堆中取同样多的物品, //规定每次至少取一个,多者不限,最后取光者得胜.

//(ak,bk)(ak ≤ bk ,k=0,1,2,...,n)表示奇异局势

//求法:

//ak =[k(1+√5)/2], bk= ak + k (k=0,1,2,...,n 方括号表示取整函数)

[判断] Gold=(1+sqrt(5.0))/2.0;

1321)假设(a,b)为第k种奇异局势(k=0,1,2...) 那么k=b-a;

2)判断其a==(int)(k*Gold),相等则为奇异局势 [注意]

采用适当的方法,可以将非奇异局势变为奇异局势. 假设面对的局势是(a,b)

若b=a,则同时从两堆中取走 a 个物体,就变为了奇异局势(0,0); 1.如果a=ak,

1.1 b>bk, 那么取走b – bk个物体,即变为奇异局势(ak, bk); 1.2 b<bk,则同时从两堆中拿走ak–a[b–ak]个物体,

变为奇异局势(a[b–ak],a[b–ak]+b–ak); 2.如果a=bk,
2.1 b>ak,则从第二堆中拿走多余的数量b–ak
2.2 b<ak,则:若b=aj(j<k)从第一堆中拿走多余的数量a– bj;(a>bj)
若b=bj(j<k)从第一堆中拿走多余的数量a–aj;(a>aj)

```
//THE code int main() {
int t,m,k,a,b; while(scanf("%d%d",&a,&b)!=EOF) {
if(a>b) swap(a,b); k=b-a; m=k*(1+sqrt(5.0))/2; if(m==a) puts("0"); else puts("1");
}
return 0; }
```

输出方案:

a=k*gold , b=a+k
k=a/gold+1; k=b/(gold+1)+1;

反 Nim 博弈(Anti-SG)

最后不能走的赢.
对于任意一个Anti-SG游戏,如果我们规定当局面中所有的单一游戏的SG值为0时,游 戏结束,则先手必胜当且仅当:
1、游戏的SG函数不为0且游戏中某个单一游戏的SG函数大于1;
2、游戏的 SG 函数为 0 且游戏中没有单一游戏的 SG 函数大于 1。

砍树博弈

一棵树 dfs 等价转换成一堆石子后,石子数可能为 0 多颗树同理...

```
// nim ^= dfs(root, root)
int dfs(int u,int f) { //从根节点开始搜
    int nim=0;
    for(int i=p[u];i!=-1;i=e[i].nxt) {
        int v=e[i].b;
        if(v==f) continue;
        nim^=(dfs(v,u)+1);
    }
    return nim;
}
```

无向图删边博弈

无向图,每次可以删除一条和 root 相连的边,并且去掉和 root 不相连的部分.
不能删了,就算输了.
[解析]
先用双连通缩点,然后就变成一棵树的删边游戏了.

1 若分量中边的条数为奇数,则该分量缩成一条新边.
2 所分量中边的条数为偶数,则该分量缩成一个点.
3 把桥边加回去.

常用算法

矩阵快速幂

```
#define CLR(a,b) memset(a,b,sizeof(a))
typedef long long ll;
typedef vector<ll> vec;
typedef vector<vec> mat;

mat mul(mat &A,mat &B)
{
    mat C(A.size(),vec(B[0].size()));
    for(int i=0;i<A.size();i++){
        for(int k=0;k<B.size();k++){
            for(int j=0;j<B[0].size();j++){
                C[i][j] = (C[i][j] + A[i][k] * B[k][j]) % mod;
            }
        }
    }
    return C;
}

mat pow(mat A,ll n){
    mat B(A.size(),vec(A.size()));
    for(int i=0;i<A.size();i++){
        B[i][i] = 1;
    }
    while(n > 0){
        if(n & 1) B = mul(B,A);
        A = mul(A, A);
        n >>= 1;
    }
    return B;
}
```

最长回文串 Manacher 算法

palindrome[i]是以i为对称中心的最长回文串长度

```
void manacher(char *text, int n) {
    palindrome[0] = 1;
    for (int i = 1, j = 0; i < n; ++ i) {
        if (j + palindrome[j] <= i) {
            palindrome[i] = 0;
        } else {
            palindrome[i] = min(palindrome[(j << 1) - i], j + palindrome[j] - i);
        }
        while (i - palindrome[i] >= 0 && i + palindrome[i] < n
                && text[i - palindrome[i]] == text[i + palindrome[i]]) {
            palindrome[i] ++;
        }
        if (i + palindrome[i] > j + palindrome[j]) {
            j = i;
        }
    }
}
```

直线下格点统计

计算

$$\sum_{0\leq i<n}\left\lfloor\frac{a+b\cdot i}{m}\right\rfloor$$

$(n,m>0,a,b\geq 0)$

```
typedef long long LL;

LL count(LL n, LL a, LL b, LL m) {
    if (b == 0) {
        return n * (a / m);
    }
    if (a >= m) {
        return n * (a / m) + count(n, a % m, b, m);
    }
    if (b >= m) {
        return (n - 1) * n / 2 * (b / m) + count(n, a, b % m, m);
```

```
    }
    return count((a + b * n) / m, (a + b * n) % m, m, b);
}
```

环状最长公共子串

```
int n, a[N << 1], b[N << 1];

bool has(int i, int j) {
    return a[(i - 1) % n] == b[(j - 1) % n];
}

const int DELTA[3][2] = {{0, -1}, {-1, -1}, {-1, 0}};

int from[N][N];

int solve() {
    memset(from, 0, sizeof(from));
    int ret = 0;
    for (int i = 1; i <= 2 * n; ++ i) {
        from[i][0] = 2;
        int left = 0, up = 0;
        for (int j = 1; j <= n; ++ j) {
            int upleft = up + 1 + !!from[i - 1][j];
            if (!has(i, j)) {
                upleft = INT_MIN;
            }
            int max = std::max(left, std::max(upleft, up));
            if (left == max) {
                from[i][j] = 0;
            } else if (upleft == max) {
                from[i][j] = 1;
            } else {
                from[i][j] = 2;
            }
            left = max;
        }
        if (i >= n) {
            int count = 0;
            for (int x = i, y = n; y;) {
                int t = from[x][y];
                count += t == 1;
                x += DELTA[t][0];
```

```
            y += DELTA[t][1];
        }
        ret = std::max(ret, count);
        int x = i - n + 1;
        from[x][0] = 0;
        int y = 0;
        while (y <= n && from[x][y] == 0) {
            y++;
        }
        for (; x <= i; ++ x) {
            from[x][y] = 0;
            if (x == i) {
                break;
            }
            for (; y <= n; ++ y) {
                if (from[x + 1][y] == 2) {
                    break;
                }
                if (y + 1 <= n && from[x + 1][y + 1] == 1) {
                    y ++;
                    break;
                }
            }
        }
    }
  }
  return ret;
}
```

双向广搜

```
struct State { }; //状态
queue<State>que[2];
bool vis[2];
bool flag;
void bfs(int d) {
    int size=que[d].size();
    while(size--) {
        //普通单广转移新状态v
        //状态出队
        if(vis[d][v])
            continue;
        if(vis[d^1][v]) { flag=true; return; }
        //新状态入队
```

```
    }
}
int dbfs() { //初始化
    int cnt=0;
    while(true) {
        cnt++;
        if(que[0].size()<que[1].size()) bfs(0);
        else bfs(1);
        if(flag) break;
    }
    return cnt;
}
```

Dancing Links-重复覆盖

```
LL L[M],R[M],U[M],D[M];
LL S[M];
bool hash1[M];
LL Col[M],Row[M];

void init()
{
    for (int i=0;i<=n;i++)
    {
        L[i]=i-1; R[i]=i+1;
        U[i]=D[i]=i;
    }
    L[0]=n;
    R[n]=0;
    int cnt=n+1;
    for (int i=0;i<n;i++)
    {
        int head=cnt,tail=cnt;
        for (int j=0;j<n;j++)
        {
            int c = j+1;
            if(g[i][j]==1)
            {
                S[c]++;
                Col[cnt]=c;
                Row[cnt]=i;
                U[D[c]]=cnt;
                D[cnt]=D[c];
                U[cnt]=c;
```

```
                D[c]=cnt;
                L[cnt]=tail; R[tail]=cnt;
                R[cnt]=head; L[head]=cnt;
                tail=cnt;
                cnt++;
            }
        }
    }
}
void remove(int &c) {
    for(int i = D[c]; i != c ; i = D[i]) {
        L[R[i]] = L[i];
        R[L[i]] = R[i];
    }
}
void resume(int &c) {
    for(int i = U[c]; i != c ; i = U[i]) {
        L[R[i]] = i;
        R[L[i]] = i;
    }
}
int h() {
    bool hash[N];
    memset(hash,false,sizeof(hash));
    int ret = 0;
    for(int c = R[0]; c != 0 ; c = R[c]) {
        if(!hash[c]) {
            ret ++;
            hash[c] = true;
            for(int i = D[c] ; i != c ; i = D[i]) {
                for(int j = R[i] ; j != i ; j = R[j]) {
                    hash[Col[j]] = true;
                }
            }
        }
    }
    return ret;
}
bool dfs(int deep,int lim) {
    if(deep + h() > lim) {
        return false;
    }
    if(R[0] == 0) {
        return true;
    }
```

```
    int idx , i , j , minnum = INF;
    for(i = R[0] ; i != 0 ; i = R[i]) {
        if(S[i] < minnum) {
            minnum = S[i];
            idx = i;
        }
    }
    for(i = D[idx]; i != idx; i = D[i]) {
        remove(i);
        for(j = R[i]; j != i ; j = R[j]) {
            remove(j);
        }
        if(dfs(deep+1,lim)) {
            return true;
        }
        for(j = L[i]; j != i ; j = L[j]) {
            resume(j);
        }
        resume(i);
    }
    return false;
}
```

Dancing-Links 精确覆盖

```
void remove(int &c) {
    L[R[c]] = L[c];
    R[L[c]] = R[c];
    for(int i = D[c]; i != c ; i = D[i]) {
        for(int j = R[i]; j != i ; j = R[j]) {
            U[D[j]] = U[j];
            D[U[j]] = D[j];
            --S[Col[j]];
        }
    }
}
void resume(int &c) {
    for(int i = U[c];i != c;i = U[i]) {
        for(int j = L[i]; j != i ; j = L[j]) {
            ++S[Col[j]];
            U[D[j]] = j;
            D[U[j]] = j;
        }
```

```
    }
    L[R[c]] = c;
    R[L[c]] = c;
}
bool dfs() {
    if(R[0] == 0) {
        return true;
    }
    int i , j;
    int idx,minnum = 999999;
    for(i = R[0];i != 0 ; i = R[i]) {
        if(S[i] < minnum) {
            minnum = S[i];
            idx = i;
        }
    }
    remove(idx);
    for(i = D[idx]; i != idx; i = D[i]) {
        ans[deep++] = Row[i];
        for(j = R[i]; j != i ; j = R[j]) {
            remove(Col[j]);
        }
        if(dfs()) {
            return true;
        }
        deep --;
        for(j = L[i]; j != i ; j = L[j]) {
            resume(Col[j]);
        }
    }
    resume(idx);
    return false;
}
```

数据结构

线段树

```
#define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1

const int maxn = 131072;
```

```
bool hash[maxn+1];
int cover[maxn << 2];
int XOR[maxn << 2];

void pushXOR(int rt)
{
   if(cover[rt] != -1)cover[rt] ^= 1;
   else XOR[rt] ^= 1;
}
void build(int l,int r,int rt) {
    if (l == r) {
       node[rt] = 1;
        return ;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    PushUP(rt);
}
void PushUP(int rt) {
    node[rt] = node[rt<<1] + node[rt<<1|1];
}
void PushDown(int rt) {
   if(cover[rt] != -1){
       cover[rt << 1] = cover[rt << 1 | 1] = cover[rt];
       XOR[rt << 1] = XOR[rt << 1 | 1] = 0;
       cover[rt] = -1;
   }
   if(XOR[rt]){
       pushXOR(rt<<1);
       pushXOR(rt<<1|1);
       XOR[rt] = 0;
   }
}
void update(int L,int R,int l,int r,int rt,int val) {
    if (L <= l && r <= R) {
       cover[rt] = val;
       XOR[rt] = 0;
        return ;
    }
    if (l == r) return ;
    PushDown(rt);
    int m = (l + r) >> 1;
    if (L <= m) update(L , R , lson, val);
   if (R > m) update(L , R , rson, val);
```

```
}
void update2(int L,int R,int l,int r,int rt) {
    if (L <= l && r <= R) {
        pushXOR(rt);
        return ;
    }
    if (l == r) return ;
    PushDown(rt);
    int m = (l + r) >> 1;
    if (L <= m) update2(L , R , lson);
    if (R > m) update2(L , R , rson);
}
void query(int l,int r,int rt) {
    if (cover[rt] == 1) {
        for (int it = l ; it <= r ; it ++) {
            hash[it] = true;
        }
        return ;
    } else if (cover[rt] == 0) return ;
    if (l == r) return ;
    PushDown(rt);
    int m = (l + r) >> 1;
    query(lson);
    query(rson);
}
```

路径压缩并查集

```
namespace ufset{
        const int N=1000;
        int fa[N],rank[N];

        void init() { for (int i=0;i<N;++i) fa[i]=i,rank[i]=0; }
        int find(int x){
            int r=x,y;
            while (fa[r]!=r) r=fa[r];
            while (fa[x]!=r) { y=fa[x],fa[x]=r,x=y;}
            return r;
        }
        void unionset(int x,int y){      // x,y roots
             if (rank[x]>rank[y]) fa[y]=x;
             else { fa[x]=y; if (rank[x]==rank[y]) ++rank[y]; }
        }
};
```

倍增 LCA

```
int head[N],idx,n;
int fa[N],deep[N];;
int f[N][M];
int ans;

struct node
{
    int v,w;
    int nxt;
}edge[N << 1];

void init()
{
    CLR(dp, 0x3f);
    CLR(head, -1);
    CLR(fa,-1);
    idx = 0;
}

void add_edge(int u,int v,int w)
{
    edge[idx].v = v;
    edge[idx].w = w;
    edge[idx].nxt = head[u];
    head[u] = idx ++;

    edge[idx].v = u;
    edge[idx].w = w;
    edge[idx].nxt = head[v];
    head[v] = idx++;
}

void dfs_deep(int s)
{
    for(int i = head[s]; ~i ; i = edge[i].nxt){
        int v = edge[i].v;
        if(v == fa[s])continue;
        fa[v] = s;
        deep[v] = deep[s] + 1;
        dfs_deep(v);
    }
}
```

```
void bz()  // 倍增祖先
{
    for(int i = 1 ; i <= n ; i++){
        f[i][0] = fa[i];
    }
    int i , j ;
    for(j = 1 ; j < M ; j ++)
    {
        for(i = 1 ; i <= n ; i ++)
        {
            f[i][j] = f[ f[i][j - 1] ][j - 1] ;
        }
    }
}

int LCA(int u , int v)
{
    if(deep[u] < deep[v]) swap(u , v) ;
    int d = deep[u] - deep[v] ;
    int i ;
    for(i = 0 ; i < M ; i ++)
    {
        if( (1 << i) & d )
        {
            u = f[u][i] ;
        }
    }
    if(u == v) return u ;
    for(i = M - 1 ; i >= 0 ; i --)
    {
        if(f[u][i] != f[v][i])
        {
            u = f[u][i] ;
            v = f[v][i] ;
        }
    }
    u = f[u][0] ;
    return u ;
}
```

树分治

HDU 4012

题意：给一棵树，各点有点权，问所有路径中点权 mod k 为 0 的有序端点对中字典序最小的

做法：点分治，分别统计

```
const int mod = 1000000+3;
#pragma comment(linker,"/STACK:102400000,102400000")
const int maxk = 1000000+20;
const int maxn = 100000+20;
int n,K;
struct edge{
   int x , next;
}e[maxn << 1];
int inv[maxk],val[maxn],pre[maxn],hash[maxk];
PII tds[maxn];
bool centroid[maxn];
int subtree_size[maxn];
int ecnt,tot;
vector<int> upd;

pair<int,int> ans;

void update(int x , int y)
{
   if (x > y) swap(x , y);
   ans = min(ans , make_pair(x , y));
}

//返回（最大子树的顶点数， 顶点编号）
pair<int,int> search_centroid(int v, int p, int t)
{
   pair<int,int> res = MP(INF,-1);
   int m = 0;
   subtree_size[v] = 1;
   for(int i=pre[v];~i;i=e[i].next){
      int w = e[i].x;
      if (w == p || centroid[w])continue;
      res = min(res, search_centroid(w, v, t));
      m = max(m, subtree_size[w]);
      subtree_size[v] += subtree_size[w];
   }
   m = max(m, t-subtree_size[v]);//the subtree of v or the subtree of v in another direction
   res = min(res, MP(m, v));
```

```
    return res;
}

//以 v 为根的子树中的所有顶点到中心的距离
void enumerate_paths(int v, int p, int d)
{
    tds[tot++] = MP(d, v);
    for(int i=pre[v];~i;i=e[i].next){
        int w = e[i].x;
        if(w == p || centroid[w])continue;
        enumerate_paths(w, v, (LL)d * val[w] % mod);
    }
}

void solve_subproblem(int v)
{
    int s = search_centroid(v, -1, subtree_size[v]).SE;
    centroid[s] = true;

    hash[1] = s;
    upd.PB(1);

    for(int i=pre[s];~i;i=e[i].next){
        int w = e[i].x;
        if(centroid[w])continue;
        tot = 0;
        enumerate_paths(w, s, val[w]);
        subtree_size[w] = tot;
        for(int j=0;j<tot;j++){
            int to = (LL)inv[tds[j].FI] * inv[val[s]] % mod * K % mod;
            if(hash[to] != 0x7F7F7F7F){
                update(tds[j].SE, hash[to]);
            }
        }
        for(int j=0;j<tot;j++){
            hash[tds[j].FI] = min(hash[tds[j].FI],tds[j].SE);
            upd.PB(tds[j].FI);
        }
    }

    while(!upd.empty()){
        hash[upd.back()] = 0x7F7F7F7F, upd.pop_back();
    }

    for(int i=pre[s];~i;i=e[i].next){
```

```
        if(centroid[e[i].x])continue;
        solve_subproblem(e[i].x);
    }

    centroid[s] = false;
}
```

计算几何

平面一条直线穿过最多圆个数

```
// template
typedef complex<double> pnt;
typedef pair<pnt,double> circle;
const int N = 1005;
const double eps = 1e-10;
const double pi = acos(-1.0);
inline bool eq(double a, double b){return abs(b-a) < eps;}
inline double fix(double arg) {
    while(arg > pi) arg -= 2*pi;
    while(arg <= -pi) arg += 2*pi;
    return arg;
}
circle num[N];
struct line {
    int id,c;
    double arg;
    line(){}
    line(int _id,int _c,double _arg) :
    id(_id) , c(_c), arg(_arg){
        //      cout<<"add: "<<id<<" "<<arg<<" "<<c<<endl;
    }
    bool operator < (const line& A) const{
        return eq(arg , A.arg) ? c > A.c : arg < A.arg;
    }
} Line[N<<2];
// cut line
#define ht first
#define rs second
inline void cut_line(const circle &A, const circle &B, int& ans, int& cnt,const int&
id) {
```

```
    double d = abs(A.ht - B.ht) ;
    if(d <= abs(A.rs - B.rs)) {
        if(d <= B.rs - A.rs ) {ans ++; return ;}
        if(eq(d , A.rs - B.rs) ) {
            Line[cnt++] = line(id, 1 , arg(B.ht - A.ht));
            Line[cnt++] = line(id,-1 , arg(B.ht - A.ht));
        }
        return ;
    }
    double t;
    t = acos((A.rs - B.rs)/ d);
    double ag = arg(B.ht - A.ht);
    Line[cnt++] = line(id, 1 , fix(ag-t));
    Line[cnt++] = line(id,-1 , fix(ag+t));
    if(d > A.rs + B.rs + eps) {
        double t = acos((A.rs + B.rs)/d);
        Line[cnt++] = line(id, 1 , fix(ag+t));
        Line[cnt++] = line(id,-1 , fix(ag-t));
    }
}
// solve
bool vis[N];
int work(int n, int len){
    int ans = 0, sum = 0;
    for(int i=0;i<n;i++){
        vis[i] = 0;
    }
    for(int i=0;i<len+len;i++) {
        int k = i%len;
        int id = Line[k].id;
        int c = Line[k].c;
        if(c == 1){
            assert(vis[id]==0);
            vis[id] = 1; sum ++;
        }
        else if(c == -1) {
            if(vis[id] == 1)
                sum --, vis[id] = 0;
        }
        if(sum > ans ) ans = sum;
    }
    return ans;
}
// main
int main(){
```

```
    int cas;
    cin >> cas;
    for(int oo=1; oo<= cas; oo++) {
        int n,len = 0;
        scanf("%d",&n);
        for(int i=0;i<n;i++) {
            int x,y,r;
            scanf("%d%d%d",&x,&y,&r);
            num[i] = make_pair(pnt(x,y) , r);
        }
        int ans = 0;
        for(int s = 0; s < n; s ++) {
            //          cout<<"start: "<<s<<endl;
            len = 0; int sum = 1;
            for(int i=0;i< n; i++) if(i!=s)
                cut_line(num[s] , num[i], sum, len, i);
            sort(Line, Line + len);
            sum += work(n,len);
            if(sum > ans) ans = sum;
        }
        printf("Case #%d: %d\n",oo, ans);
    }
}
```

基础定义

```
const double EPS = 1e-8;
const double PI = acos(-1.0);
template <class T> T sqr(T x){return x * x;}
```

点和向量

```
struct Point{
   double x,y;
   Point(){}
   Point(double x,double y) : x(x),y(y){}
};
inline double dist(const Point &a,const Point &b)
{
        return sqr(a.x-b.x)+sqr(a.y-b.y);
}
```

```
typedef Point Vec;
Vec operator + (Vec a,Vec b){return Vec(a.x+b.x,a.y+b.y);};
Vec operator - (Vec a,Vec b){return Vec(a.x-b.x,a.y-b.y);};
Vec operator * (Vec a,double p){return Vec(a.x * p,a.y *p);};
Vec operator / (Vec a,double p){return Vec(a.x / p,a.y / p);};
inline int sgn(double x){return (x>EPS) - (x<-EPS);}
bool operator < (Point a, Point b) { return sgn(a.x - b.x) < 0 || sgn(a.x - b.x) == 0 &&
a.y < b.y;}
bool operator == (Point a, Point b) { return sgn(a.x - b.x) == 0 && sgn(a.y - b.y) == 0;}

inline double dotDet(Vec a, Vec b) { return a.x * b.x + a.y * b.y;}
inline double crossDet(Vec a, Vec b) { return a.x * b.y - a.y * b.x;}
inline double dotDet(Point o, Point a, Point b) { return dotDet(a - o, b - o);}
inline double crossDet(Point o, Point a, Point b) { return crossDet(a - o, b - o);}
inline double vecLen(Vec x) { return sqrt(dotDet(x, x));}
inline Vec vecUnit(Vec x) { return x / vecLen(x);}
inline Vec normal(Vec x) { return Vec(-x.y, x.x) / vecLen(x);}
```

点在线段上

```
inline bool onSeg(Point x, Point a, Point b) { return sgn(crossDet(x, a, b)) == 0 &&
sgn(dotDet(x, a, b)) < 0;}
```

线段是否相交

```
int segIntersect(Point a, Point c, Point b, Point d) {
	Vec v1 = b - a, v2 = c - b, v3 = d - c, v4 = a - d;
	int a_bc = sgn(crossDet(v1, v2));
	int b_cd = sgn(crossDet(v2, v3));
	int c_da = sgn(crossDet(v3, v4));
	int d_ab = sgn(crossDet(v4, v1));
	if (a_bc * c_da > 0 && b_cd * d_ab > 0) return 1;
	if (onSeg(b, a, c) && c_da) return 2;
	if (onSeg(c, b, d) && d_ab) return 2;
	if (onSeg(d, c, a) && a_bc) return 2;
	if (onSeg(a, d, b) && b_cd) return 2;
	return 0;
}
```

直线相交

```
Point lineIntersect(Point P, Vec v, Point Q, Vec w) {
	Vec u = P - Q;
```

```
        double t = crossDet(w, u) / crossDet(v, w);
        return P + v * t;
}
```

点在线段上

```
inline bool InLine(const Point &a,const Point &b,const Point &c)
{
        return fabs(crossDet(b-a,c-a))<EPS && dotDet(a-c,b-c)<EPS;
}
```

线段与线段交点

```
inline void LineToLine(const Point &a,const Point &b,const Point &c,const Point &d)
{
        double s1=crossDet(c-a,b-a),s2=crossDet(b-a,d-a);
        if (s1*s2<-EPS) return;
        Point e=c+(d-c)*s1/(s1+s2);

        if (InLine(a,b,e) && InLine(c,d,e))
        {
                Add(e.x);
        }
}
```

中垂线

```
void midVerticalLine(Point a,Point b,Point &c,Point &d){
    c.x = (a.x + b.x)/2;
    c.y = (a.y + b.y)/2;
    d.x = c.x + (a.y - b.y)/2;
    d.y = c.y + (b.x - a.x)/2;
}
```

三维点模板

```
struct P3
{
        double x,y,z;

        P3(){}
        P3(double a,double b,double c){x=a;y=b;z=c;}
```

```
        inline void read(){scanf("%lf%lf%lf",&x,&y,&z);}
};
inline P3 operator +(const P3 &a,const P3 &b){return P3(a.x+b.x,a.y+b.y,a.z+b.z);}
inline P3 operator -(const P3 &a,const P3 &b){return P3(a.x-b.x,a.y-b.y,a.z-b.z);}
inline P3 operator *(const double &a,const P3 &b){return P3(a*b.x,a*b.y,a*b.z);}
inline P3 operator *(const P3 &b,const double &a){return P3(a*b.x,a*b.y,a*b.z);}
inline P3 operator /(const P3 &a,const double &b){return P3(a.x/b,a.y/b,a.z/b);}
```

三角剖分

三角剖分剖分线模板

```
struct Tinter
{
        double x,y,Area,mid;
        int delta;
        Tinter(){}
        Tinter(double xx,double yy,double mm,int dd,double aa)
        {
                x=xx;y=yy;mid=mm;
                delta=dd;Area=aa;
        }
};
inline bool operator <(const Tinter &a,const Tinter &b){return a.mid>b.mid+EPS;}
inline bool operator ==(const Tinter &a,const Tinter &b){return fabs(a.mid-b.mid)<EPS;}
```

Vector 多边形模板

```
struct Poly{
    vector<Point> pt;
    Poly() { pt.clear(); }
    ~Poly(){}
    Poly(vector<Point> &pt) : pt(pt){}
    Point operator [] (int x) const { return pt[x];}
    int size() { return pt.size();}
     double area() {
        double ret = 0.0;
        for (int i = 0, sz = pt.size(); i < sz; i++) {
              ret += crossDet(pt[i], pt[(i + 1) % sz]);
        }
        return fabs(ret / 2.0);
    }
```

```
};
```

多边形切割

```
 Poly cutPoly(Poly &poly, Point a, Point b) {
    Poly ret = Poly();
    int n = poly.size();
    for (int i = 0; i < n; i++) {
          Point c = poly[i], d = poly[(i + 1) % n];
          if (sgn(crossDet(a, b, c)) >= 0) ret.pt.push_back(c);
          if (sgn(crossDet(b - a, c - d)) != 0) {
                 Point ip = lineIntersect(a, b - a, c, d - c);
                 if (onSeg(ip, c, d)) ret.pt.push_back(ip);
          }
    }
    return ret;
 }
```

线段与多边形相交

```
 bool isIntersect(Point a, Point b, Poly &poly) {
    for (int i = 0, sz = poly.size(); i < sz; i++) {
          if (segIntersect(a, b, poly[i], poly[(i + 1) % sz])) return true;
    }
    return false;
 }

struct Circle {
        Point c;
        double r;
        Circle(){}
        Circle(Point c,double r):c(c),r(r){}
};
```

点在圆内

```
inline bool inCircle(Point a, Circle c) { return vecLen(c.c - a) < c.r;}
bool lineCircleIntersect(Point s, Point t, Circle C, vector<Point> &sol) {
        Vec dir = t - s, nor = normal(dir);
        Point mid = lineIntersect(C.c, nor, s, dir);
        double len = sqr(C.r) - dotDet(C.c - mid, C.c - mid);
         if (sgn(len) < 0) return 0;//不相交
```

```
        if (sgn(len) == 0) {
               sol.push_back(mid);//相切
               return 1;
        }
        Vec dis = vecUnit(dir);
        len = sqrt(len);
        sol.push_back(mid + dis * len);
        sol.push_back(mid - dis * len);
        return 2;//正常情况
}
```

线段与圆相交

```
bool segCircleIntersect(Point s, Point t, Circle C){
    vector<Point> tmp;
    tmp.clear();
    if (lineCircleIntersect(s ,t, C, tmp)){
        if(tmp.size() < 2)return false;
        for(int i=0,sz = tmp.size();i < sz;i++){
            if(onSeg(tmp[i],s,t))return true;
        }
    }
    return false;
}
```

点在多边形内

```
int ptInPoly( Point p, Poly &poly) {
    int wn = 0,sz = poly.size();
    for(int i=0;i<sz;i++){
        if(onSeg(p, poly[i], poly[(i+1) % sz])) return -1;//在边上
        int k = sgn(crossDet(poly[(i+1) % sz] - poly[i], p - poly[i]));
        int d1 = sgn(poly[i].y - p.y);
        int d2 = sgn(poly[(i+1) % sz].y - p.y);
        if (k > 0 && d1 <= 0 && d2 >0 ) wn++;
        if (k < 0 && d2 <= 0 && d1 > 0) wn--;
    }
    if (wn != 0) return 1;
    return 0;
}
```

空间四面体

```
#define MAX_N 10
struct Poly4{
   P3 p[4];
}a[MAX_N];
struct Sphere
{
	P3 o;
	double r;
	inline void read(){o.read();scanf("%lf",&r);}
}b[MAX_N];
int n,m;

vector <Tinter> inter;

vector<Poly> polies;
vector<Circle> circles;
vector<double> bak;
inline void Add(double x)
{
	bak.push_back(x);
}
```

圆与圆交点

```
inline void CircleIntersectCircle(const Circle &a,const Circle &b)
{
	double l=dist(a.c,b.c);
	double s=((a.r-b.r)*(a.r+b.r)/l+1)/2;
	double t=sqrt(-(l-sqr(a.r+b.r))*(l-sqr(a.r-b.r))/(l*l*4));
	double ux=b.c.x-a.c.x,uy=b.c.y-a.c.y;
	double ix=a.c.x+s*ux+t*uy,iy=a.c.y+s*uy-t*ux;
	double jx=a.c.x+s*ux-t*uy,jy=a.c.y+s*uy+t*ux;
	Add(ix);
	Add(jx);
}
```

线段与圆交点

```
inline void LineToCircle(const Point &a,const Point &b,const Circle &c)
{
	double h=fabs(crossDet(c.c-a,b-a))/vecLen(b-a);
```

```
        if (h>c.r+EPS) return;
        double lamda=dotDet(c.c-a,b-a);
        lamda/=dist(a,b);
        Point x=a+(b-a)*lamda;
        double d=sqrt( sqr(c.r)-sqr(h) );
        d/=vecLen((b-a));
        Point e=x+(b-a)*d;
        Point f=x-(b-a)*d;
        if (InLine(a,b,e))
                Add(e.x);
        if (InLine(a,b,f))
                Add(f.x);
        return;
}
```

圆和圆交点

```
inline void CircleToCircle()
{
   //求圆和圆的交点
        for (int i=0;i<circles.size();++i)
        {
                for (int j=i+1;j<circles.size();++j)
                if (dist(circles[i].c,circles[j].c)<=sqr(circles[i].r+circles[j].r))
                if (dist(circles[i].c,circles[j].c)>=sqr(circles[i].r-circles[j].r))
       {
          CircleIntersectCircle(circles[i],circles[j]);
       }
        }
}
```

圆和多边形交点

```
inline void CircleToPoly()
{
        for (int i=0;i<circles.size();++i)
                for (int j=0;j<polies.size();++j)
                        for (int v=0;v<polies[j].size();++v)

        LineToCircle(polies[j][v],polies[j][(v+1)%polies[j].size()],circles[i]);
}
```

多边形与多边形交点

```
inline void PolyToPoly()
{
	for (int i=0;i<polies.size();++i)
		for (int j=i+1;j<polies.size();++j)
			for (int u=0;u<polies[i].size();++u)
				for (int v=0;v<polies[j].size();++v)

	LineToLine(polies[i][u],polies[i][(u+1)%polies[i].size()],polies[j][v],polies[j
][(v+1)%polies[j].size()]);
}
```

直线 x = x0 与圆的交点

```
inline void Get(const Circle &c,double x,double &l,double &r)
{
   //直线 x = x0 与圆的交点
	double y=fabs(c.c.x-x);
	double d=sqrt(fabs( sqr(c.r)-sqr(y) ));
	l=c.c.y+d;
	r=c.c.y-d;
}
```

梯形/三角形边界与圆边界形成的弧面积

```
inline double arcArea(const Circle &a,double l,double x,double r,double y)
{
   //梯形/三角形边界与圆边界形成的弧的面积
	double len=sqrt(sqr(l-r) + sqr(x-y));
	double d=sqrt(sqr(a.r)-sqr(len)/4.0);
	double angle=atan(len/2.0/d);
	return fabs(angle*sqr(a.r)-d*len/2.0);
}
```

取剖分线段

```
inline void Get_Interval(const Circle &a,double l,double r)
{
	double L1,L2,R1,R2,M1,M2;
	Get(a,l,L1,L2);
	Get(a,r,R1,R2);
```

```
        Get(a,(l+r)/2.0,M1,M2);
        int D1=1,D2=-1;
        double A1=arcArea(a,l,L1,r,R1),A2=arcArea(a,l,L2,r,R2);
        inter.push_back( Tinter(L1,R1,M1,D1,A1) );
        inter.push_back( Tinter(L2,R2,M2,D2,A2) );
}
```

计算一段面积

```
inline double calcSlice(double xl,double xr)
{
   const int inf = 8;
        inter.clear();
        double lmost=-inf,rmost=inf;
        for (int i=0;i<polies.size();++i)
        {
int cc=0;
Tinter I[5];
for (int u=0;u<polies[i].size();++u)
{
        Point x=polies[i][u];
        Point y=polies[i][(u+1)%polies[i].size()];
        double l=min(x.x,y.x),r=max(x.x,y.x);
        if (l<=xl+EPS && xr<=r+EPS)
{
if (fabs(l-r)<EPS) continue;
Point d=y-x;
Point Left=x+d/d.x*(xl-x.x);
Point Right=x+d/d.x*(xr-x.x);
Point Mid=(Left+Right)/2;
                                I[cc++]=Tinter(Left.y,Right.y,Mid.y,1,0);
                        }
                }
                sort(I,I+cc);
                if (cc==2)
                {
                        I[1].delta=-1;
                        inter.push_back(I[0]);
                        inter.push_back(I[1]);
                        lmost=max(lmost,I[1].mid);
                        rmost=min(rmost,I[0].mid);
                }
        }
```

```
	for (int i=0;i<circles.size();++i)
	if (fabs(circles[i].c.x-xl)<circles[i].r+EPS &&
fabs(circles[i].c.x-xr)<circles[i].r+EPS)
		Get_Interval(circles[i],xl,xr);

	if (!inter.size()) return 0;
	double ans=0;
	sort(inter.begin(),inter.end());
	int cnt=0;
	for (int i=0;i<inter.size();++i)
	{
		if (cnt>0)
		{

	ans+=(fabs(inter[i-1].x-inter[i].x)+fabs(inter[i-1].y-inter[i].y))*(xr-xl)/2.0;
			ans+=inter[i-1].delta*inter[i-1].Area;
			ans-=inter[i].delta*inter[i].Area;
		}
		cnt+=inter[i].delta;
	}
	return ans;
}
```

极角排序

```
inline void ToHull(vector <Point> &a)
{
	sort(a.begin(),a.end());
	int hull[10],len,limit=1;
	hull[len=1]=0;
	for (int i=1;i<4;++i)
	{
		while (len>limit &&
crossDet(a[hull[len]]-a[hull[len-1]],a[i]-a[hull[len]])>=0) --len;
		hull[++len]=i;
	}
	limit=len;
	for (int i=2;i>=0;--i)
	{
		while (len>limit &&
crossDet(a[hull[len]]-a[hull[len-1]],a[i]-a[hull[len]])>=0) --len;
		hull[++len]=i;
	}
	vector <Point> b=a;
```

```
        a.resize(len-1);
        for (int i=0;i<len-1;++i)
                a[i]=b[hull[i+1]];
}
```

计算总面积

```
double calcArea(double z){
    //cout<<z<<endl;
    polies.clear();
    circles.clear();
    bak.clear();
    //与四面体的交面
    for(int i=0;i<n;i++){
        vector <Point> cross;
        for(int j=0;j<4;j++){
            for(int k=j+1;k<4;k++){
                if(sgn(a[i].p[j].z-a[i].p[k].z)){
                        double l = min(a[i].p[j].z,a[i].p[k].z);
                        double r = max(a[i].p[j].z,a[i].p[k].z);
                        if(l<=z+EPS && z<=r+EPS){
                            //线与平面相交，求交点
                            P3 d=a[i].p[k]-a[i].p[j];
                            d=d/d.z;
                            d=d*(z-a[i].p[j].z);
                            d=d+a[i].p[j];
                            cross.push_back(Point(d.x,d.y));
                        }
                }
            }
        }
        sort(cross.begin(),cross.end());
        cross.erase(unique(cross.begin(),cross.end()),cross.end());
        if (cross.size()>2)
                {
            if (cross.size()==4)
            ToHull(cross);
                        polies.push_back(cross);
                }
    }
    for(int i=0;i<m;i++){
        if (fabs(z-b[i].o.z)+EPS<b[i].r)
        {
            Point o(b[i].o.x,b[i].o.y);
```

```
            double r=sqrt( sqr(b[i].r)-sqr(z-b[i].o.z) );
            circles.push_back(Circle(o,r));
        }
    }

    for(int i=0;i<polies.size();i++){
        for(int j=0;j<polies[i].size();j++)
            Add(polies[i][j].x);
    }
    for(int i=0;i<circles.size();i++){
            Add(circles[i].c.x - circles[i].r);
            Add(circles[i].c.x);
            Add(circles[i].c.x + circles[i].r);
    }

         CircleToCircle();
         CircleToPoly();
         PolyToPoly();

    sort(bak.begin(),bak.end());
/*
   if(bak.size()>2){
        cout<<z<<":";
        for(int i=0;i<bak.size();i++)
        cout<<bak[i]<<" ";
    cout<<endl;
   }
*/
         double res=0;

         for (int i=0;i+1<bak.size();++i)
         if (fabs(bak[i+1]-bak[i])>EPS)
                  res+=calcSlice(bak[i],bak[i+1]);

         return res;
}

void solve(){
    int inf = 8;
    const int block = 4000;
    double ans = calcArea(-inf)+calcArea(inf);
    double h = (inf + inf)/(double)block;
    /*
    for(int i=0;i<=block;i++){
        ans += calcArea(-inf+i*h);
```

```
    }*/

    for (int i=1;i<block;i+=2)
        ans+=4*calcArea(-inf+i*h);
    for (int i=2;i<block;i+=2)
        ans+=2*calcArea(-inf+i*h);
    ans*=(h/3.0);

    printf("%.3lf\n",ans);
}
int main()
{
    while(cin>>n>>m){
        if(!(n||m))break;
        for(int i=0;i<n;i++)
            for(int j=0;j<4;j++)
                a[i].p[j].read();
        for(int i=0;i<m;i++){
            b[i].read();
        }
        solve();
    }
    return 0;
}
```

点在圆内

```
inline bool inCircle(Point a, Circle c) { return vecLen(c.c - a) < c.r;}
bool lineCircleIntersect(Point s, Point t, Circle C, vector<Point> &sol) {
        Vec dir = t - s, nor = normal(dir);
        Point mid = lineIntersect(C.c, nor, s, dir);
        double len = sqr(C.r) - dotDet(C.c - mid, C.c - mid);
         if (sgn(len) < 0) return 0;//不相交
         if (sgn(len) == 0) {
                sol.push_back(mid);//相切
                return 1;
        }
        Vec dis = vecUnit(dir);
        len = sqrt(len);
        sol.push_back(mid + dis * len);
        sol.push_back(mid - dis * len);
        return 2;//正常情况
}
```

线段与圆相交

```
bool segCircleIntersect(Point s, Point t, Circle C){
    vector<Point> tmp;
    tmp.clear();
    if (lineCircleIntersect(s ,t, C, tmp)){
        if(tmp.size() < 2)return false;
        for(int i=0,sz = tmp.size();i < sz;i++){
            if(onSeg(tmp[i],s,t))return true;
        }
    }
    return false;
}
```

多边形切割成三角形

```
vector<Poly> cutPolies(Point s, Point t, vector<Poly> polies) {
    vector<Poly> ret;
    ret.clear();
    for(int i=0,sz = polies.size();i < sz; i++){
        Poly tmp;
        tmp = cutPoly(polies[i] , s , t);
        if(tmp.size() >= 3 && tmp.area() > EPS ) ret.push_back(tmp);
        tmp = cutPoly(polies[i] , t , s);
        if(tmp.size() >= 3 && tmp.area() > EPS) ret.push_back(tmp);
    }
    return ret;
```

点在多边形内

```
int ptInPoly( Point p, Poly &poly) {
    int wn = 0,sz = poly.size();
    for(int i=0;i<sz;i++){
        if(onSeg(p, poly[i], poly[(i+1) % sz])) return -1;//在边上
        int k = sgn(crossDet(poly[(i+1) % sz] - poly[i], p - poly[i]));
        int d1 = sgn(poly[i].y - p.y);
        int d2 = sgn(poly[(i+1) % sz].y - p.y);
        if (k > 0 && d1 <= 0 && d2 >0 ) wn++;
        if (k < 0 && d2 <= 0 && d1 > 0) wn--;
    }
    if (wn != 0) return 1;
    return 0;
}
```

```
bool circlePoly( Circle C, Poly &poly ){
    int sz = poly.size();
    if(ptInPoly(C.c, poly))return true;
    for(int i=0;i<sz;i++){
        if(inCircle(poly[i], C))return true;
    }
    for(int i=0;i<sz;i++){
        if(segCircleIntersect(poly[i],poly[(i+1) % sz], C))return true;
    }
    return false;
}

vector<double> circleWithPolies( Circle C, vector<Poly> &polies ){
    vector<double> ret;
    ret.clear();
    int sz = polies.size();
    for(int i=0;i<sz;i++){
        if(circlePoly(C,polies[i]))
        ret.push_back(polies[i].area());
    }
    return ret;
}
```

三角剖分简化模板

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

const double EPS = 1e-8;
const double PI = acos(-1.0);
template <class T> T sqr(T x) {return x*x;}
struct Point{
    double x,y;
    Point(){};
    Point(double x,double y):x(x),y(y){};
};
inline double dis(const Point &a,const Point &b)
```

```
{
    return sqr(a.x-b.x)+sqr(a.y-b.y);
}
typedef Point Vec;
Vec operator+(Vec a,Vec b){
    return Vec(a.x+b.x,a.y+b.y);
}
Vec operator-(Vec a,Vec b){
    return Vec(a.x-b.x,a.y-b.y);
}
Vec operator*(Vec a,double p){
    return Vec(a.x*p,a.y*p);
}
Vec operator/(Vec a,double p){
    return Vec(a.x/p,a.y/p);
}
inline int sgn(double x){return (x>EPS)-(x<-EPS);}

inline double dotDet(Vec a,Vec b){
    return a.x*b.x + a.y*b.y;
}
inline double crossDet(Vec a,Vec b){
    return a.x* b.y - a.y * b.x;
}
inline double dotDet(Point o,Point a,Point b){
    return dotDet(a-o,b-o);
}
inline double crossDet(Point o,Point a,Point b){
    return crossDet(a-o,b-o);
}
inline double vecLen(Vec x){
    return sqrt(dotDet(x,x));
}
inline Vec vecUnit(Vec x){
    return x/vecLen(x);
}
inline Vec normal(Vec x){
    return Vec(-x.y,x.x);
}
inline bool onSeg(Point x,Point a,Point b){
    return sgn(crossDet(x,a,b)==0) &&  sgn(dotDet(x,a,b)<0);
}

struct Tinter{double x,y,Area,mid;int delta;Tinter(){};
             Tinter(double xx,double yy,double mm,int dd,double aa){
```

```
                x =xx;y=yy;mid=mm;
                delta = dd;Area = aa;
            }
};
inline bool operator<(const Tinter &a,const Tinter &b)
{
    return a.mid>b.mid +EPS;
}
inline bool operator==(const Tinter &a,const Tinter &b)
{
    return fabs(a.mid-b.mid)<EPS;
}

int segIntersect(Point a,Point c,Point b,Point d){
    Vec v1 = b-a,v2 = c-b,v3 = d-c,v4 = a-d;
    int a_bc = sgn(crossDet(v1,v2));
    int b_cd = sgn(crossDet(v2,v3));
    int c_da = sgn(crossDet(v3,v4));
    int d_ab = sgn(crossDet(v4,v1));
    if(a_bc * c_da >0 && b_cd * d_ab >0)return 1;
    if(onSeg(b,a,c)&& c_da) return 2;
    if(onSeg(c,b,d)&& d_ab) return 2;
    if(onSeg(d,c,a)&& a_bc) return 2;
    if(onSeg(a,d,b)&& b_cd) return 2;
    return 0;
}
Point lineIntersect(Point P,Vec v,Point Q,Vec w)
{
    Vec u = P-Q;
    double t = crossDet(w,u) / crossDet(v,w);
    return P + v * t;
}

struct Poly{
    vector<Point> pt;
    Poly(){pt.clear();}
    ~Poly(){}
    Poly(vector<Point>&pt):pt(pt){};
    Point operator[](int x)const {return pt[x];}
    int size(){return pt.size();}
};



vector<double> bak;
```

```
double ans[55];
vector<Poly> polies;
vector<Tinter> inter;



inline void Add(double x){
    bak.push_back(x);
}

inline bool InLine(const Point &a,const Point &b,const Point &c)
{
    return fabs(crossDet(b-a,c-a)<EPS && dotDet(a-c,b-c)<EPS);
}

inline void LineToLine(const Point &a,const Point &b,const Point &c,const Point &d)
{
    double s1 = crossDet(c-a,b-a),s2 = crossDet(b-a,d-a);
    if(s1*s2 < -EPS)return ;
    Point e = c+(d-c)*s1/(s1+s2);
    if(InLine(a,b,e) && InLine(c,d,e))
        Add(e.x);
}

inline void PolyToPoly()
{
    for(int i=0;i<polies.size();++i)
        for(int j=i+1;j<polies.size();++j)
        for(int u=0;u<polies[i].size();++u)
        for(int v=0;v<polies[j].size();++v)

LineToLine(polies[i][u],polies[i][(u+1)%polies[i].size()],polies[j][v],polies[j][(
v+1)%polies[j].size()]);
}

inline void calcSlice(double xl,double xr)
{
    const int inf = 1000;
    inter.clear();
    double lmost = -inf,rmost = inf;
    for(int i=0;i<polies.size();++i)
    {
        int cc = 0;
        Tinter I[5];
        for(int u=0;u<polies[i].size();++u)
```

```
        {
            Point x = polies[i][u];
            Point y = polies[i][(u+1)%3];
            double l=min(x.x,y.x),r=max(x.x,y.x);
            if(l<=xl+EPS && xr<=r+EPS){
                if(fabs(l-r)<EPS)continue;
                Point d = y-x;
                Point Left = x+d/d.x*(xl-x.x);
                Point Right = x+d/d.x*(xr-x.x);
                Point Mid = (Left+Right)/2.0;
                I[cc++] = Tinter(Left.y,Right.y,Mid.y,1,0);
            }
            }
                        sort(I,I+cc);
            if(cc==2){
                I[1].delta=-1;
                inter.push_back(I[0]);
                inter.push_back(I[1]);
                lmost = max(lmost,I[1].mid);
                rmost = min(rmost,I[0].mid);
        }
    }

    if(!inter.size())return ;
    sort(inter.begin(),inter.end());
    int cnt = 0;
    //cout<<inter.size()<<endl;
    for(int i=0;i<inter.size();++i)
    {
        if(cnt>0)
        {
            ans[cnt]                                                        +=
fabs((inter[i-1].x-inter[i].x)+fabs(inter[i-1].y-inter[i].y))*(xr-xl)/2.0;
        }
        cnt += inter[i].delta;
    }
    return ;
}
void calcArea()
{
    bak.clear();
    for(int i=0;i<polies.size();i++)
    for(int j=0;j<polies[i].size();j++)
        Add(polies[i][j].x);
    PolyToPoly();
```

```
    sort(bak.begin(),bak.end());
    for(int i=0;i<bak.size();i++)
        if(fabs(bak[i+1]-bak[i]>EPS))
            calcSlice(bak[i],bak[i+1]);
}
int main()
{
    int T;
    cin>>T;
    while(T--)
    {
        polies.clear();
        memset(ans,0,sizeof(ans));
        int n;
        cin>>n;
        for(int i=0;i<n;i++){
            Point p[3];
            for(int i=0;i<3;i++)
                cin>>p[i].x>>p[i].y;
            Vec a = p[2]-p[0];
            Vec b = p[1]-p[0];
            if(fabs(a.x*b.y-b.x*a.y)<EPS){
                continue;
            }
            vector<Point> v;
            for(int i=0;i<3;i++)
            v.push_back(p[i]);
            Poly tra(v);
            polies.push_back(tra);
        }
        calcArea();
        for(int i=1;i<=n;i++)
            printf("%.8lf\n",ans[i]);
    }
    return 0;
}
```

其他

大数模板

```
#pragma comment(linker, "/STACK:1024000000,1024000000")
```

```
#include <stdio.h>
#include <string.h>
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>
#include <set>
#include <map>
#include <string>
#include <math.h>
#include <stdlib.h>
#include <time.h>
using namespace std;

/*
 * 完全大数模板
 * 输出 cin>>a
 * 输出 a.print();
 * 注意这个输入不能自动去掉前导 0 的，可以先读入到 char 数组，去掉前导 0，再用构造函数。
 */
#define MAXN 9999
#define MAXSIZE 1010
#define DLEN 4

class BigNum
{
public:
    int a[500];  //可以控制大数的位数
    int len;
public:
    BigNum(){len=1;memset(a,0,sizeof(a));}  //构造函数
    BigNum(const int);     //将一个 int 类型的变量转化成大数
    BigNum(const char*);   //将一个字符串类型的变量转化为大数
    BigNum(const BigNum &); //拷贝构造函数
    BigNum &operator=(const BigNum &); //重载赋值运算符，大数之间进行赋值运算
    friend istream& operator>>(istream&,BigNum&); //重载输入运算符
    friend ostream& operator<<(ostream&,BigNum&); //重载输出运算符

    BigNum operator+(const BigNum &)const;  //重载加法运算符，两个大数之间的相加运算
    BigNum operator-(const BigNum &)const;  //重载减法运算符，两个大数之间的相减运算
    BigNum operator*(const BigNum &)const;  //重载乘法运算符，两个大数之间的相乘运算
    BigNum operator/(const int &)const;     //重载除法运算符，大数对一个整数进行相除运算

    BigNum operator^(const int &)const;     //大数的 n 次方运算
    int operator%(const int &)const;       //大数对一个 int 类型的变量进行取模运算
```

```
    bool operator>(const BigNum &T)const;   //大数和另一个大数的大小比较
    bool operator>(const int &t)const;      //大数和一个 int 类型的变量的大小比较

    void print();       //输出大数
};
BigNum::BigNum(const int b)   //将一个 int 类型的变量转化为大数
{
    int c,d=b;
    len=0;
    memset(a,0,sizeof(a));
    while(d>MAXN)
    {
        c=d-(d/(MAXN+1))*(MAXN+1);
        d=d/(MAXN+1);
        a[len++]=c;
    }
    a[len++]=d;
}
BigNum::BigNum(const char *s)  //将一个字符串类型的变量转化为大数
{
    int t,k,index,L,i;
    memset(a,0,sizeof(a));
    L=strlen(s);
    len=L/DLEN;
    if(L%DLEN)len++;
    index=0;
    for(i=L-1;i>=0;i-=DLEN)
    {
        t=0;
        k=i-DLEN+1;
        if(k<0)k=0;
        for(int j=k;j<=i;j++)
            t=t*10+s[j]-'0';
        a[index++]=t;
    }
}
BigNum::BigNum(const BigNum &T):len(T.len)  //拷贝构造函数
{
    int i;
    memset(a,0,sizeof(a));
    for(i=0;i<len;i++)
        a[i]=T.a[i];
}
BigNum & BigNum::operator=(const BigNum &n)  //重载赋值运算符，大数之间赋值运算
{
```

```
    int i;
    len=n.len;
    memset(a,0,sizeof(a));
    for(i=0;i<len;i++)
        a[i]=n.a[i];
    return *this;
}
istream& operator>>(istream &in,BigNum &b)
{
    char ch[MAXSIZE*4];
    int i=-1;
    in>>ch;
    int L=strlen(ch);
    int count=0,sum=0;
    for(i=L-1;i>=0;)
    {
        sum=0;
        int t=1;
        for(int j=0;j<4&&i>=0;j++,i--,t*=10)
        {
            sum+=(ch[i]-'0')*t;
        }
        b.a[count]=sum;
        count++;
    }
    b.len=count++;
    return in;
}
ostream& operator<<(ostream& out,BigNum& b)  //重载输出运算符
{
    int i;
    cout<<b.a[b.len-1];
    for(i=b.len-2;i>=0;i--)
    {
        printf("%04d",b.a[i]);
    }
    return out;
}
BigNum BigNum::operator+(const BigNum &T)const   //两个大数之间的相加运算
{
    BigNum t(*this);
    int i,big;
    big=T.len>len?T.len:len;
    for(i=0;i<big;i++)
    {
```

```
        t.a[i]+=T.a[i];
        if(t.a[i]>MAXN)
        {
            t.a[i+1]++;
            t.a[i]-=MAXN+1;
        }
    }
    if(t.a[big]!=0)
       t.len=big+1;
    else t.len=big;
    return t;
}
BigNum BigNum::operator-(const BigNum &T)const  //两个大数之间的相减运算
{
    int i,j,big;
    bool flag;
    BigNum t1,t2;
    if(*this>T)
    {
        t1=*this;
        t2=T;
        flag=0;
    }
    else
    {
        t1=T;
        t2=*this;
        flag=1;
    }
    big=t1.len;
    for(i=0;i<big;i++)
    {
        if(t1.a[i]<t2.a[i])
        {
            j=i+1;
            while(t1.a[j]==0)
                j++;
            t1.a[j--]--;
            while(j>i)
                t1.a[j--]+=MAXN;
            t1.a[i]+=MAXN+1-t2.a[i];
        }
        else t1.a[i]-=t2.a[i];
    }
    t1.len=big;
```

```
    while(t1.a[len-1]==0 && t1.len>1)
    {
        t1.len--;
        big--;
    }
    if(flag)
        t1.a[big-1]=0-t1.a[big-1];
    return t1;
}
BigNum BigNum::operator*(const BigNum &T)const  //两个大数之间的相乘
{
    BigNum ret;
    int i,j,up;
    int temp,temp1;
    for(i=0;i<len;i++)
    {
        up=0;
        for(j=0;j<T.len;j++)
        {
            temp=a[i]*T.a[j]+ret.a[i+j]+up;
            if(temp>MAXN)
            {
                temp1=temp-temp/(MAXN+1)*(MAXN+1);
                up=temp/(MAXN+1);
                ret.a[i+j]=temp1;
            }
            else
            {
                up=0;
                ret.a[i+j]=temp;
            }
        }
        if(up!=0)
          ret.a[i+j]=up;
    }
    ret.len=i+j;
    while(ret.a[ret.len-1]==0 && ret.len>1)ret.len--;
    return ret;
}
BigNum BigNum::operator/(const int &b)const  //大数对一个整数进行相除运算
{
    BigNum ret;
    int i,down=0;
    for(i=len-1;i>=0;i--)
    {
```

```
        ret.a[i]=(a[i]+down*(MAXN+1))/b;
        down=a[i]+down*(MAXN+1)-ret.a[i]*b;
    }
    ret.len=len;
    while(ret.a[ret.len-1]==0 && ret.len>1)
        ret.len--;
    return ret;
}
int BigNum::operator%(const int &b)const   //大数对一个 int 类型的变量进行取模
{
    int i,d=0;
    for(i=len-1;i>=0;i--)
        d=((d*(MAXN+1))%b+a[i])%b;
    return d;
}
BigNum BigNum::operator^(const int &n)const  //大数的 n 次方运算
{
    BigNum t,ret(1);
    int i;
    if(n<0)exit(-1);
    if(n==0)return 1;
    if(n==1)return *this;
    int m=n;
    while(m>1)
    {
        t=*this;
        for(i=1;(i<<1)<=m;i<<=1)
           t=t*t;
        m-=i;
        ret=ret*t;
        if(m==1)ret=ret*(*this);
    }
    return ret;
}
bool BigNum::operator>(const BigNum &T)const    //大数和另一个大数的大小比较
{
    int ln;
    if(len>T.len)return true;
    else if(len==T.len)
    {
        ln=len-1;
        while(a[ln]==T.a[ln]&&ln>=0)
          ln--;
        if(ln>=0 && a[ln]>T.a[ln])
           return true;
```

```
        else
            return false;
    }
    else
        return false;
}
bool BigNum::operator>(const int &t)const  //大数和一个 int 类型的变量的大小比较
{
    BigNum b(t);
    return *this>b;
}
void BigNum::print()   //输出大数
{
    int i;
    printf("%d",a[len-1]);
    for(i=len-2;i>=0;i--)
      printf("%04d",a[i]);
    printf("\n");
}
bool ONE(BigNum a)
{
    if(a.len == 1 && a.a[0] == 1)return true;
    else return false;
}
BigNum A,B,X,Y;
char str1[10010],str2[10010],str3[10010],str4[10010];


int a[1010],b[1010],x[1010],y[1010];
int c[1010];
int main()
{
    //freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
    int T;
    int n;
    int iCase = 0;
    scanf("%d",&T);
    while(T--)
    {
        iCase++;
        scanf("%d",&n);
        cin>>A>>X>>B>>Y;
        printf("Case %d: ",iCase) ;
        A = A-1;
```

```
X = X-1;
B = B-1;
Y = Y-1;
for(int i = 0;i < n;i++)
{
    if(A.a[0]%2 == 0)a[i] = 0;
    else a[i] = 1;
    if(B.a[0]%2 == 0)b[i] = 0;
    else b[i] = 1;
    if(X.a[0]%2 == 0)x[i] = 0;
    else x[i] = 1;
    if(Y.a[0]%2 == 0)y[i] = 0;
    else y[i] = 1;
    A = A/2;
    B = B/2;
    X = X/2;
    Y = Y/2;
}
bool flag = false;
for(int k = 0;k <= n;k++)
{
    x[n] = x[0];
    y[n] = y[0];
    for(int i = 0;i < n;i++)
    {
        x[i] = x[i+1];
        y[i] = y[i+1];
    }
    for(int i = 0;i < n;i++)
    {
        if(a[i] == x[i])c[i] = 0;
        else c[i] = 1;
    }
    bool fff = true;
    for(int i = 0;i < n;i++)
        if(b[i]^c[i] != y[i])
        {
            fff = false;
            break;
        }
    if(fff)flag = true;
    if(flag)break;

}
if(flag)printf("Yes\n");
```

```
        else printf("No\n");
    }
    return 0;
}
```

io 优化

```
char buf[MAXN];
gets(buf);
int v;
char *p=strtok(buf," ");
while(p) {
    sscanf(p,"%d",&v);
    p=strtok(NULL," ");
}

//ACMonster's IO
int get() {
    char c;
    while(c=getchar(),(c<'0'||c>'9')&&(c!='-'));
    bool flag=(c=='-');
    if(flag)
        c=getchar();
    int x=0;
    while(c>='0'&&c<='9') {
        x=x*10+c-'0';
        c=getchar();
    }
    return flag?-x:x;
}
void output(long long x) {
    if(x<0) {
        putchar('-');
        x=-x;
    }
    int len=0,data[20];
    while(x) {
        data[len++]=x%10;
        x/=10;
    }
    if(!len)
        data[len++]=0;
    while(len--)
```

```
        putchar(data[len]+'0');
    putchar('\n');
}

inline int readint() {
    char c=getchar();
    while(!isdigit(c))
        c=getchar();
    int x=0;
    while(isdigit(c)) {
        x=x*10+c-'0';
        c=getchar();
    }
    return x;
}
char buf[20];
inline void writeint(int x) {
    if(x==0) {
        putchar('0');
        return;
    }
    if(x<0) {
        putchar('-');
        x=-x;
    }
    int bas=0;
    while(x) {
        buf[bas++]=x%10+'0';
        x/=10;
    }
    while(k--)
        putchar(buf[bas]);
}

template<class T> inline T& RDD(T &x) {
    char c;
    for (c = getchar(); c < '-'; c = getchar());
    if (c == '-') {
        x = '0' - getchar();
        for (c = getchar(); '0' <= c && c <= '9'; c = getchar()) x = x * 10 + '0'
- c;
    } else {
        x = c - '0';
        for (c = getchar(); '0' <= c && c <= '9'; c = getchar()) x = x * 10 + c -
'0';
```

```
    }
    return x;
}
```

用位运算生成下一个含有 k 个 1 的二进制数

```
b = x & -x;
t = x + b;
c = t ^ x;
m = (c >> 2) / b;
r = t | m; //最终结果
```

枚举子集

```
for (int x = S; x; x = (x-1)&S)
```

预处理子集和

```
for (k = 0 ; k < m ; ++ k)
    for (j = 0 ; j < 1 << m ; ++ j)
        if (j >> k & 1) {
            A[j] += A[j ^ (1 << k)];
```