目录

[计算几何 4](#_Toc370974339)

[多边形与圆面积 4](#_Toc370974340)

[多边形与圆面积 4](#_Toc370974341)

[半平面交（nlogn） 5](#_Toc370974342)

[二维计算几何操作 8](#_Toc370974343)

[三维计算几何操作 13](#_Toc370974344)

[三维几何操作合并 17](#_Toc370974345)

[三维旋转操作 18](#_Toc370974346)

[三维凸包（n^2） 19](#_Toc370974347)

[三维凸包求重心 22](#_Toc370974348)

[随机增量最小覆盖圆 23](#_Toc370974349)

[两圆面积交 25](#_Toc370974350)

[圆的面积模板（n^2logn） 26](#_Toc370974351)

[最小覆盖球 30](#_Toc370974352)

[最大空凸包 32](#_Toc370974353)

[Voronoi 35](#_Toc370974354)

[三角形的心 42](#_Toc370974355)

[四边形费马点 43](#_Toc370974356)

[最近点对 47](#_Toc370974357)

[最远点对 49](#_Toc370974358)

[经纬度求球面最短距离 51](#_Toc370974359)

[长方体表面两点最短距离 51](#_Toc370974360)

[Farmland 52](#_Toc370974361)

[图论 54](#_Toc370974362)

[最大团 54](#_Toc370974363)

[极大团计数 55](#_Toc370974364)

[2-SAT 56](#_Toc370974365)

[KM 57](#_Toc370974366)

[无向图最小割 59](#_Toc370974367)

[弦图相关 62](#_Toc370974368)

[弦图完美消除序列 63](#_Toc370974369)

[带花树 65](#_Toc370974370)

[最小树形图 67](#_Toc370974371)

[动态最小生成树 71](#_Toc370974372)

[Hopcroft 75](#_Toc370974373)

[割点缩块 77](#_Toc370974374)

[割边缩块 79](#_Toc370974375)

[K短路（可重复） 79](#_Toc370974376)

[K短路（不可重复） 84](#_Toc370974377)

[数学 87](#_Toc370974378)

[Miller-Rabin 87](#_Toc370974379)

[启发式分解 88](#_Toc370974380)

[N次剩余 91](#_Toc370974381)

[2次剩余 93](#_Toc370974382)

[线性筛法 94](#_Toc370974383)

[Pell方程 95](#_Toc370974384)

[皮克公式 96](#_Toc370974385)

[蔡勒公式 96](#_Toc370974386)

[莫比乌斯函数以及gcd=1的对数 96](#_Toc370974387)

[牛顿迭代 97](#_Toc370974388)

[FFT 98](#_Toc370974389)

[FFT(integer) 101](#_Toc370974390)

[Romberg&Simpson 105](#_Toc370974391)

[多项式求根（求导二分） 106](#_Toc370974392)

[线性规划 108](#_Toc370974393)

[数据结构 112](#_Toc370974394)

[回文串 112](#_Toc370974395)

[后缀数组（DC3） 113](#_Toc370974396)

[后缀数组(nlogn) 115](#_Toc370974397)

[后缀自动机 117](#_Toc370974398)

[扩展KMP 118](#_Toc370974399)

[动态树 119](#_Toc370974400)

[KD-Tree 124](#_Toc370974401)

[AC自动机 127](#_Toc370974402)

[左偏树 129](#_Toc370974403)

[杂 130](#_Toc370974404)

[字符串最小表示 130](#_Toc370974405)

[曼哈顿最小生成树 130](#_Toc370974406)

[表达式计算 134](#_Toc370974407)

[DancingLinks 136](#_Toc370974408)

[最长公共子序列 138](#_Toc370974409)

[高精度计算 141](#_Toc370974410)

[图同构hash 146](#_Toc370974411)

[双人零和矩阵游戏（公式） 146](#_Toc370974412)

[综合 146](#_Toc370974413)

[多边形内点的计数 150](#_Toc370974414)

[基本形公式 150](#_Toc370974415)

[树的计数 152](#_Toc370974416)

[代数 152](#_Toc370974417)

[三角公式 153](#_Toc370974418)

[积分表 153](#_Toc370974419)

[Java IO&vimrc 155](#_Toc370974420)

# 计算几何

## 多边形与圆面积交

//BEGIN

//intersection of a circle and a simple-polygon

struct point {

double x, y;

point() {}

point(double \_x, double \_y): x(\_x), y(\_y) {}

point operator +(const point &a) const { return point(x + a.x, y + a.y); }

point operator -(const point &a) const { return point(x - a.x, y - a.y); }

double len() const { return sqrt(x \* x + y \* y); }

void output() { printf("%.15f %.15f\n", x, y); }

} ORI;

const double eps = 1e-8;

const double PI = acos(-1.);

double r;

const int maxn = 110000;

int n;

point info[maxn];

inline int Sign(double x) {

if (x > eps) return 1;

if (x < -eps) return -1;

return 0;

}

double dot(const point &a, const point &b) {

return a.x \* b.x + a.y \* b.y;

}

double cross(const point &a, const point &b) {

return a.x \* b.y - a.y \* b.x;

}

//用有向面积，划分成一个三角形和圆的面积的交

double area2(point pa, point pb) {

if (pa.len() < pb.len()) swap(pa, pb);

if (pb.len() < eps) return 0;

double a, b, c, B, C, sinB, cosB, sinC, cosC, S, h, theta;

a = pb.len();

b = pa.len();

c = (pb - pa).len();

//sinB = abs(cross(pb, pb-pa)) / a / c;

cosB = dot(pb, pb - pa) / a / c;

B = acos(cosB);

//sinC = abs(cross(pa, pb)) / a / b;

cosC = dot(pa, pb) / a / b;

C = acos(cosC);

//printf("area2( %.4f, %.4f, %.4f )\n", a, b, C/PI\*180);

if (a > r) {

S = (C/2)\*r\*r;

h = a\*b\*sin(C)/c;

if (h < r && B < PI/2) S -= (acos(h/r)\*r\*r - h\*sqrt(r\*r-h\*h));

} else if (b > r) {

theta = PI - B - asin(sin(B)/r\*a);

S = .5\*a\*r\*sin(theta) + (C-theta)/2\*r\*r;

} else {

S = .5\*sin(C)\*a\*b;

}

//printf("res = %.4f\n", S);

return S;

}

double area() {

double S = 0;

for (int i = 0; i < n; ++i) {

S += area2(info[i], info[i + 1]) \* Sign(cross(info[i], info[i + 1]));

}

return fabs(S);

}

//END

## 半平面交（nlogn）

#define LL long long

#define eps 1e-10

#define inf 10000

#define zero(a) fabs(a)<eps

#define N 20005

struct Point{

double x,y;

}p[N\*2];

struct Segment{

Point s,e;

double angle;

void get\_angle(){angle=atan2(e.y-s.y,e.x-s.x);}

}seg[N];

int m;

//叉积为正说明，p2在p0-p1的左侧

double xmul(Point p0,Point p1,Point p2){

return (p1.x-p0.x)\*(p2.y-p0.y)-(p2.x-p0.x)\*(p1.y-p0.y);

}

Point Get\_Intersect(Segment s1,Segment s2){

double u=xmul(s1.s,s1.e,s2.s),v=xmul(s1.e,s1.s,s2.e);

Point t;

t.x=(s2.s.x\*v+s2.e.x\*u)/(u+v);t.y=(s2.s.y\*v+s2.e.y\*u)/(u+v);

return t;

}

bool cmp(Segment s1,Segment s2){

//先按极角排序

if(s1.angle>s2.angle) return true;

//极角相等，内侧的在前

else if(zero(s1.angle-s2.angle)&&xmul(s2.s,s2.e,s1.e)>-eps) return true;

return false;

}

void HalfPlaneIntersect(Segment seg[],int n){

sort(seg,seg+n,cmp);

int tmp=1;

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

if(!zero(seg[i].angle-seg[tmp-1].angle))

seg[tmp++]=seg[i];

n=tmp;

Segment deq[N];

deq[0]=seg[0];deq[1]=seg[1];

int head=0,tail=1;

for(int i=2;i<n;i++){

while(head<tail&&xmul(seg[i].s,seg[i].e,Get\_Intersect(deq[tail],deq[tail-1]))<-eps) tail--;

while(head<tail&&xmul(seg[i].s,seg[i].e,Get\_Intersect(deq[head],deq[head+1]))<-eps) head++;

deq[++tail]=seg[i];

}

while(head<tail&&xmul(deq[head].s,deq[head].e,Get\_Intersect(deq[tail],deq[tail-1]))<-eps) tail--;

while(head<tail&&xmul(deq[tail].s,deq[tail].e,Get\_Intersect(deq[head],deq[head+1]))<-eps) head++;

if(head==tail) return;

m=0;

for(int i=head;i<tail;i++)

p[m++]=Get\_Intersect(deq[i],deq[i+1]);

if(tail>head+1)

p[m++]=Get\_Intersect(deq[head],deq[tail]);

}

double Get\_area(Point p[],int &n){

double area=0;

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

area+=xmul(p[0],p[i],p[i+1]);

return fabs(area)/2.0;

}

int main(){

int n;

while(scanf("%d",&n)!=EOF){

seg[0].s.x=0;seg[0].s.y=0;seg[0].e.x=10000;seg[0].e.y=0;seg[0].get\_angle();

seg[1].s.x=10000;seg[1].s.y=0;seg[1].e.x=10000;seg[1].e.y=10000;seg[1].get\_angle();

seg[2].s.x=10000;seg[2].s.y=10000;seg[2].e.x=0;seg[2].e.y=10000;seg[2].get\_angle();

seg[3].s.x=0;seg[3].s.y=10000;seg[3].e.x=0;seg[3].e.y=0;seg[3].get\_angle();

for(int i=0;i<n;i++){

scanf("%lf%lf%lf%lf",&seg[i+4].s.x,&seg[i+4].s.y,&seg[i+4].e.x,&seg[i+4].e.y);

seg[i+4].get\_angle();

}

HalfPlaneIntersect(seg,n+4);

printf("%.1f\n",Get\_area(p,m)); //m<3 表示无解

}

return 0;

}

## 二维计算几何操作

const double eps = 1e-8;

const double pi = acos(-1.0);

const double inf = 1e5;

const int maxn = 100;

inline int Sign(double a) {

return a < -eps ? -1 : a > eps;

}

inline double Arc\_Sin(double a) {

if (Sign(a + 1) <= 0) return -pi / 2;

if (Sign(a - 1) >= 0) return pi / 2;

return asin(a);

}

inline double Arc\_Cos(double a) {

if (Sign(a + 1) <= 0) return pi;

if (Sign(a - 1) >= 0) return 0;

return acos(a);

}

inline double Sqr(double a) {

return a \* a;

}

inline double Sqrt(double a) {

return a <= 0 ? 0 : sqrt(a);

}

struct Point {

double x, y;

Point() {

}

Point(double x, double y) : x(x), y(y) {

}

void Input() {

scanf("%lf %lf", &x, &y);

}

double Length() const {

return Sqrt(Sqr(x) + Sqr(y));

}

Point Rotate(double a) const {

return Point(x \* cos(a) - y \* sin(a), x \* sin(a) + y \* cos(a));

}

Point Unit() const;

};

Point operator + (const Point &a, const Point &b) {

return Point(a.x + b.x, a.y + b.y);

}

Point operator - (const Point &a, const Point &b) {

return Point(a.x - b.x, a.y - b.y);

}

Point operator \* (const Point &a, double b) {

return Point(a.x \* b, a.y \* b);

}

Point operator / (const Point &a, double b) {

return Point(a.x / b, a.y / b);

}

Point Point::Unit() const {

return \*this / Length();

}

double Det(const Point &a, const Point &b) {

return a.x \* b.y - a.y \* b.x;

}

double Dot(const Point &a, const Point &b) {

return a.x \* b.x + a.y \* b.y;

}

double Dist(const Point &a, const Point &b, const Point &c) {

return abs(Det(a - c, b - c) / (a - b).Length());

}

double Angle(const Point &a, const Point &b) {

return Arc\_Cos(Dot(a, b) / a.Length() / b.Length());

}

bool Line\_Intersect(const Point &a, const Point &b, const Point &c, const Point &d, Point &e) {

double s1 = Det(c - a, d - a);

double s2 = Det(d - b, c - b);

if (!Sign(s1 + s2)) return 0;

e = (b - a) \* (s1 / (s1 + s2)) + a;

return 1;

}

int Side(const Point &a, const Point &b, const Point &c) {

return Sign(Det(c - a, b - a));

}

bool In\_The\_Seg(const Point &a, const Point &b, const Point &c) {

if (Sign(Dist(a, b, c))) return 0;// Not needed when you make sure it does technically.

return Sign(Dot(a - c, b - c)) <= 0;

}

bool Seg\_Intersect(const Point &a, const Point &b, const Point &c, const Point &d, Point &e) {

double s1 = Det(c - a, d - a);

double s2 = Det(d - b, c - b);

if (!Sign(s1 + s2)) return 0;

e = (b - a) \* (s1 / (s1 + s2)) + a;

return In\_The\_Seg(a, b, e) && In\_The\_Seg(c, d, e);

}

struct Circle {

Point o;

double r;// Squared

bool Inside(Point a) {

return Sqr(a.x - o.x) + Sqr(a.y - o.y) <= r;

}

void Calc(Point a, Point b) {

o.x = (a.x + b.x) / 2;

o.y = (a.y + b.y) / 2;

r = Sqr(a.x - o.x) + Sqr(a.y - o.y);

}

void Calc(Point a, Point b, Point c) {// Not certain if a, b and c lie in the same line, which needs prejudging.

double a1 = 2 \* (a.x - b.x);

double b1 = 2 \* (a.y - b.y);

double c1 = Sqr(a.x) - Sqr(b.x) + Sqr(a.y) - Sqr(b.y);

double a2 = 2 \* (a.x - c.x);

double b2 = 2 \* (a.y - c.y);

double c2 = Sqr(a.x) - Sqr(c.x) + Sqr(a.y) - Sqr(c.y);

o.x = (c1 \* b2 - c2 \* b1) / (a1 \* b2 - a2 \* b1);

o.y = (c1 \* a2 - c2 \* a1) / (a2 \* b1 - a1 \* b2);

r = Sqr(a.x - o.x) + Sqr(a.y - o.y);

}

bool Intersect\_With\_Line(Point fr, Point to, Point &A, Point &B) const {

if (Sign(Det(o - fr, to - fr)) > 0) swap(fr, to);

double R = Sqrt(r);

double h = Dist(fr, to, o);

if (Sign(h - R) > 0) return 0;

Point mm = (to - fr).Unit().Rotate(-pi / 2) \* h + o;

double l = Sqrt(Sqr(R) - Sqr(h));

Point vv = (to - fr).Unit() \* l;

A = mm - vv;

B = mm + vv;

return 1;

}

bool Contain(const Circle &a) const {// Not tested

return Sign(Sqrt(a.r) + (o - a.o).Length() - Sqrt(r)) < 0;

}

bool Disjunct(const Circle &a) const {// Not tested

return Sign(Sqrt(a.r) + Sqrt(r) - (o - a.o).Length()) < 0;

}

};

bool Intersect(Circle a, Circle b, Point &A, Point &B) {// Not tested, and must take care if a and b are the same one

if (a.Contain(b) || b.Contain(a) || a.Disjunct(b)) return 0;

double s1 = (a.o - b.o).Length();

double s2 = (a.r - b.r) / s1;

double aa = (s1 + s2) / 2;

double bb = (s1 - s2) / 2;

Point mm = (b.o - a.o) \* (aa / (aa + bb)) + a.o;

double h = Sqrt(a.r - Sqr(aa));

Point vv = (b.o - a.o).Unit().Rotate(pi / 2) \* h;

A = mm + vv;

B = mm - vv;

return 1;

}

struct Polygon {

Point list[maxn];

int n;

Polygon() {

}

Polygon(const Polygon &a) {

n = a.n;

int i;

for (i = 0; i < n; i++)

list[i] = a.list[i];

}

Polygon &operator = (const Polygon &a) {

if (this == &a) return \*this;

n = a.n;

int i;

for (i = 0; i < n; i++)

list[i] = a.list[i];

return \*this;

}

Polygon Cut(const Point &a, const Point &b) {

static Polygon res;

res.n = 0;

int i, s1, s2;

Point curr;

for (i = 0; i < n; i++) {

s1 = Sign(Det(list[i] - a, b - a));

s2 = Sign(Det(list[(i + 1) % n] - a, b - a));

if (s1 <= 0) res.list[res.n++] = list[i];

if (s1 \* s2 < 0) {

Line\_Intersect(a, b, list[i], list[(i + 1) % n], curr);

res.list[res.n++] = curr;

}

}

return res;

}

Polygon Strict\_Cut(const Point &fr, const Point &to) const {

static Polygon res;

res.n = 0;

int i, s1, s2;

Point a, b;

for (i = 0; i < n; i++)

if (Side(fr, to, list[i]) < 0) break;

if (i == n) return res;

Point c;

for (i = 0; i < n; i++) {

a = list[i];

b = list[(i + 1) % n];

s1 = Side(fr, to, a);

s2 = Side(fr, to, b);

if (s1 <= 0) res.list[res.n++] = a;

if (s1 \* s2 < 0) {

Line\_Intersect(fr, to, a, b, c);

res.list[res.n++] = c;

}

}

return res;

}

bool Contain(const Point &curr) const {

int i, res = 0;

Point A, B;

for (i = 0; i < n; i++) {

A = list[i];

B = list[(i + 1) % n];

if (In\_The\_Seg(A, B, curr)) return 1;

if (Sign(A.y - B.y) <= 0) swap(A, B);

if (Sign(curr.y - A.y) > 0) continue;

if (Sign(curr.y - B.y) <= 0) continue;

res += Sign(Det(B - curr, A - curr)) > 0;

}

return res & 1;

}

};

## 三维计算几何操作

//BEGIN TEMPLATE HERE  
const double eps = 1e-8;  
int Sign(double x) {  
    return x < -eps ? -1 : x > eps;  
}  
struct point3 {  
    double x, y, z;  
    point3() {}  
    point3(double x, double y, double z): x(x), y(y), z(z) {}  
    point3 operator +(const point3 &a) const { return point3(x+a.x, y+a.y, z+a.z); }  
    point3 operator -(const point3 &a) const { return point3(x-a.x, y-a.y, z-a.z); }  
    point3 operator \*(double k) const { return point3(x\*k, y\*k, z\*k); }  
    point3 operator /(double k) const { return point3(x/k, y/k, z/k); }  
    double len() const { return sqrt(len2()); }  
    double len2() const { return x\*x + y\*y + z\*z; }  
};  
double vlen(const point3 &a) {  
    return a.len();  
}  
point3 det(const point3 &a, const point3 &b) {   
    return point3(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 dot(const point3 &a, const point3 &b) {   
    return a.x\*b.x + a.y\*b.y + a.z\*b.z;   
}  
struct line3 {  
    point3 a, b;  
    line3() {}  
    line3(point3 a, point3 b): a(a), b(b) {}  
};  
struct plane3 {  
    point3 a, b, c;  
    plane3() {}  
    plane3(point3 a, point3 b, point3 c): a(a), b(b), c(c) {}   
};  
**//平面法向量**

point3 pvec(point3 s1,point3 s2,point3 s3){return det((s1-s2),(s2-s3));}

**//check共线**

int dots\_inline(point3 p1,point3 p2,point3 p3){

return vlen(det(p1-p2,p2-p3))<eps;}

**//check共平面**

int dots\_onplane(point3 a,point3 b,point3 c,point3 d){

return zero(dot(pvec(a,b,c),d-a));}

**//check在线段上(end point inclusive)**

int dot\_online\_in(point3 p,line3 l)

int dot\_online\_in(point3 p,point3 l1,point3 l2){return zero(vlen(det(p-l1,p-l2)))&&(l1.x-p.x)\*(l2.x-p.x)<eps&&(l1.y-p.y)\*(l2.y-p.y)<eps&&(l1.z-p.z)\*(l2.z-p.z)<eps; }

**//check在线段上(end point exclusive)**

int dot\_online\_ex(point3 p,line3 l)

int dot\_online\_ex(point3 p,point3 l1,point3 l2){ return dot\_online\_in(p,l1,l2)&&(!zero(p.x-l1.x)||!zero(p.y-l1.y)||!zero(p.z-l1.z))&&(!zero(p.x-l2.x)||!zero(p.y-l2.y)||!zero(p.z-l2.z));

}

**//check一个点是否在三角形里(inclusive)**

int dot\_inplane\_in(point3 p,plane3 s)

int dot\_inplane\_in(point3 p,point3 s1,point3 s2,point3 s3){

return zero(vlen(det(s1-s2,s1-s3))-vlen(det(p-s1,p-s2))-

vlen(det(p-s2,p-s3))-vlen(det(p-s3,p-s1)));

}

**//check一个点是否在三角形里(exclusive)**

int dot\_inplane\_ex(point3 p,plane3 s)

int dot\_inplane\_ex(point3 p,point3 s1,point3 s2,point3 s3){

return dot\_inplane\_in(p,s1,s2,s3)&&vlen(det(p-s1,p-s2))>eps&&

vlen(det(p-s2,p-s3))>eps&&vlen(det(p-s3,p-s1))>eps;

}

**//check if two point and a segment in one plane have the same side**

int same\_side(point3 p1,point3 p2,point3 l1,point3 l2)

int same\_side(point3 p1,point3 p2,line3 l){

return dot(det(l.a-l.b,p1-l.b),det(l.a-l.b,p2-l.b))>eps;

}

**//check if two point and a segment in one plane have the opposite side**

int opposite\_side(point3 p1,point3 p2,point3 l1,point3 l2)

int opposite\_side(point3 p1,point3 p2,line3 l){

return dot(det(l.a-l.b,p1-l.b), det(l.a-l.b,p2-l.b))<-eps;

}

**//check if two point is on the same side of a plane**

int same\_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3)

int same\_side(point3 p1,point3 p2,plane3 s){

return dot(pvec(s),p1-s.a)\*dot(pvec(s),p2-s.a)>eps;

}

**//check if two point is on the opposite side of a plane**

int opposite\_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3)

int opposite\_side(point3 p1,point3 p2,plane3 s){

return dot(pvec(s),p1-s.a)\*dot(pvec(s),p2-s.a)<-eps;

}

**//check if two straight line is parallel**

int parallel(point3 u1,point3 u2,point3 v1,point3 v2)

int parallel(line3 u,line3 v){ return vlen(det(u.a-u.b,v.a-v.b))<eps; }

**//check if two plane is parallel**

int parallel(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)

int parallel(plane3 u,plane3 v){return vlen(det(pvec(u),pvec(v)))<eps;}

**//check if a plane and a line is parallel**

int parallel(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)

int parallel(line3 l,plane3 s){ return zero(dot(l.a-l.b,pvec(s))); }

**//check if two line is perpendicular**

int perpendicular(point3 u1,point3 u2,point3 v1,point3 v2)

int perpendicular(line3 u,line3 v){return zero(dot(u.a-u.b,v.a-v.b)); }

**//check if two plane is perpendicular**

int perpendicular(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)

int perpendicular(plane3 u,plane3 v){ return zero(dot(pvec(u),pvec(v))); }

**//check if plane and line is perpendicular**

int perpendicular(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)

int perpendicular(line3 l,plane3 s){return vlen(det(l.a-l.b,pvec(s)))<eps;}

**//check 两条线段是否有交点(end point inclusive)**

int intersect\_in(point3 u1,point3 u2,point3 v1,point3 v2)

int intersect\_in(line3 u,line3 v){

if (!dots\_onplane(u.a,u.b,v.a,v.b)) return 0;

if (!dots\_inline(u.a,u.b,v.a)||!dots\_inline(u.a,u.b,v.b))

return !same\_side(u.a,u.b,v)&&!same\_side(v.a,v.b,u);

return dot\_online\_in(u.a,v)||dot\_online\_in(u.b,v)||

dot\_online\_in(v.a,u)||dot\_online\_in(v.b,u);

}

**//check 两条线段是否有交点(end point exclusive)**

int intersect\_ex(point3 u1,point3 u2,point3 v1,point3 v2)

int intersect\_ex(line3 u,line3 v){

return dots\_onplane(u.a,u.b,v.a,v.b)&&opposite\_side(u.a,u.b,v)&&

opposite\_side(v.a,v.b,u);

}

**//check线段和三角形是否有交点(end point and border inclusive)**

int intersect\_in(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)

int intersect\_in(line3 l,plane3 s){

return !same\_side(l.a,l.b,s)&&!same\_side(s.a,s.b,l.a,l.b,s.c)&&

!same\_side(s.b,s.c,l.a,l.b,s.a)&&!same\_side(s.c,s.a,l.a,l.b,s.b);

}

**//check线段和三角形是否有交点(end point and border exclusive)**

int intersect\_ex(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)

int intersect\_ex(line3 l,plane3 s){

return opposite\_side(l.a,l.b,s)&&opposite\_side(s.a,s.b,l.a,l.b,s.c)&& opposite\_side(s.b,s.c,l.a,l.b,s.a)&&opposite\_side(s.c,s.a,l.a,l.b,s.b);}

**//calculate the intersection of two line**

**//Must you should ensure they are co-plane and not parallel**

point3 intersection(point3 u1,point3 u2,point3 v1,point3 v2)

point3 intersection(line3 u,line3 v){

point3 ret=u.a;

double t=((u.a.x-v.a.x)\*(v.a.y-v.b.y)-(u.a.y-v.a.y)\*(v.a.x-v.b.x))

/((u.a.x-u.b.x)\*(v.a.y-v.b.y)-(u.a.y-u.b.y)\*(v.a.x-v.b.x));

ret+=(u.b-u.a)\*t; return ret;

}

**//calculate the intersection of plane and line**

point3 intersection(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)

point3 intersection(line3 l,plane3 s){

point3 ret=pvec(s);

double t=(ret.x\*(s.a.x-l.a.x)+ret.y\*(s.a.y-l.a.y)+ret.z\*(s.a.z-l.a.z))/

(ret.x\*(l.b.x-l.a.x)+ret.y\*(l.b.y-l.a.y)+ret.z\*(l.b.z-l.a.z));

ret=l.a + (l.b-l.a)\*t; return ret;

}

**//calculate the intersection of two plane**

bool intersection(plane3 pl1 , plane3 pl2 , line3 &li) {

if (parallel(pl1,pl2)) return false;

li.a=parallel(pl2.a,pl2.b, pl1) ? intersection(pl2.b,pl2.c, pl1.a,pl1.b,pl1.c) : intersection(pl2.a,pl2.b, pl1.a,pl1.b,pl1.c);

point3 fa; fa=det(pvec(pl1),pvec(pl2)); li.b=li.a+fa; return true;

}

**//distance from point to line**

double ptoline(point3 p,point3 l1,point3 l2)

double ptoline(point3 p,line3 l){

return vlen(det(p-l.a,l.b-l.a))/distance(l.a,l.b);}

**//distance from point to plane**

double ptoplane(point3 p,plane3 s){

return fabs(dot(pvec(s),p-s.a))/vlen(pvec(s));}

double ptoplane(point3 p,point3 s1,point3 s2,point3 s3)

**//distance between two line 当u,v平行时有问题**

double linetoline(line3 u,line3 v){

point3 n=det(u.a-u.b,v.a-v.b); return fabs(dot(u.a-v.a,n))/vlen(n);

}

double linetoline(point3 u1,point3 u2,point3 v1,point3 v2)

**//cosine value of the angle formed by two lines**

double angle\_cos(line3 u,line3 v){

return dot(u.a-u.b,v.a-v.b)/vlen(u.a-u.b)/vlen(v.a-v.b);

}

double angle\_cos(point3 u1,point3 u2,point3 v1,point3 v2)

**//cosine value of the angle formed by two planes**

double angle\_cos(plane3 u,plane3 v){

return dot(pvec(u),pvec(v))/vlen(pvec(u))/vlen(pvec(v));}

double angle\_cos(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)

**//cosine value of the angle formed by plane and line**

double angle\_cos(line3 l,plane3 s){

return dot(l.a-l.b,pvec(s))/vlen(l.a-l.b)/vlen(pvec(s));}

double angle\_cos(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)

## 三维几何操作合并

const double pi = acos(-1.0); double a[4][4];

int dcmp(const double &a, const double &b = 0, const double & zero = 1e-6){

if (a - b < -zero) return -1; return a - b > zero;}

void multi(const double a[4][4],const double b[4][4],double c[4][4]){

for(int i=0;i<4;i++)

for(int j=0;j<4;j++){

c[i][j]=a[i][0]\*b[0][j];

for(int k=1;k<4;k++)

c[i][j]+=a[i][k]\*b[k][j];

}}

void multi(double a[4][4],const double b[4][4]){

static double c[4][4];

multi(a,b,c);

memcpy(a,c,sizeof(a[0][0])\*16);

}

void Macro(){

double b[4][4]={1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1};

memcpy(a,b,sizeof(a[0][0])\*16);

}

void Translation(const Point\_3 &s){

double p[4][4]={1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, s.x, s.y, s.z, 1};

multi(a,p);

}

void Scaling(const Point\_3 &s){

double p[4][4]={s.x, 0, 0, 0, 0, s.y, 0, 0, 0, 0, s.z, 0, 0, 0, 0, 1};

multi(a,p);

}

void Rotate(const Point\_3 &s, double r) {

double l=s.Length(); double x=s.x/l,y=s.y/l,z=s.z/l;

double SinA=sin(r),CosA=cos(r);

double p[4][4]={CosA + (1 - CosA) \* x \* x, (1 - CosA) \* x \* y - SinA \* z, (1 - CosA) \* x \* z + SinA \* y, 0,(1 - CosA) \* y \* x + SinA \* z,

CosA + (1 - CosA) \* y \* y, (1 - CosA) \* y \* z - SinA \* x, 0,

(1 - CosA) \* z \* x - SinA \* y, (1 - CosA) \* z \* y + SinA \* x, CosA + (1 - CosA) \* z \* z, 0, 0, 0, 0, 1};

multi(a,p);

}

Point\_3 opt(const Point\_3&s){

double x,y,z;

return Point\_3( s.x \* a[0][0] + s.y \* a[1][0] + s.z \* a[2][0] + a[3][0],

s.x \* a[0][1] + s.y \* a[1][1] + s.z \* a[2][1] + a[3][1],

s.x \* a[0][2] + s.y \* a[1][2] + s.z \* a[2][2] + a[3][2]);

}

int main(){

Macro();

int n;for (scanf("%d", &n); n; n--) {

char c; Point\_3 p;

scanf("\n%c%lf%lf%lf", &c, &p.x, &p.y, &p.z);

if (c == 'T') Translation(p); if (c == 'S') Scaling(p);

if (c == 'R') { double r;scanf("%lf\n", &r);

Rotate(p, r); //===========绕OP逆时针旋转r角度

}}

for (scanf("%d", &n); n; n--) {

Point\_3 p, p2; scanf("%lf%lf%lf", &p.x, &p.y, &p.z);

p2 = opt(p); printf(“%f %f %f\n”,p2.x,p2.y,p2.z);

}}

## 三维旋转操作

//a点绕Ob向量，逆时针旋转弧度angle, sin(angle),cos(angle)先求出来，减少精度问题。

point e1,e2,e3; point Rotate( point a, point b, double angle ){

b.std();//单位化，注意b不能为（0，0，0）

e3=b; double lens=a\*e3;//dot(a,e3)

e1=a - e3\*lens; if (e1.len()>(1e-8)) e1.std(); else return a;

e2=e1/e3; //det(e1,e3)

double x1,y1,x,y; y1=a\*e1; x1=a\*e2;

x=x1\*cos(angle) - y1\*sin(angle); y=x1\*sin(angle) + y1\*cos(angle);

return e3\*lens + e1\*y + e2\*x; }

## 三维凸包（n^2）

#define SIZE(X) (int(X.size()))

#define PI 3.14159265358979323846264338327950288

const double eps = 1e-8;

inline int Sign(double x) {

return x < -eps ? -1 : (x > eps ? 1 : 0);

}

inline double Sqrt(double x) {

return x < 0 ? 0 : sqrt(x);

}

struct Point {

double x, y, z;

Point() {

x = y = z = 0;

}

Point(double x, double y, double z): x(x), y(y), z(z) {

}

bool operator <(const Point &p) const {

return x < p.x || x == p.x && y < p.y || x == p.x && y == p.y && z < p.z;

}

bool operator ==(const Point &p) const {

return Sign(x - p.x) == 0 && Sign(y - p.y) == 0 && Sign(z - p.z) == 0;

}

Point operator +(const Point &p) const {

return Point(x + p.x, y + p.y, z + p.z);

}

Point operator -(const Point &p) const {

return Point(x - p.x, y - p.y, z - p.z);

}

Point operator \*(const double k) const {

return Point(x \* k, y \* k, z \* k);

}

Point operator /(const double k) const {

return Point(x / k, y / k, z / k);

}

Point cross(const Point &p) const {

return Point(y \* p.z - z \* p.y, z \* p.x - x \* p.z, x \* p.y - y \* p.x);

}

double dot(const Point &p) const {

return x \* p.x + y \* p.y + z \* p.z;

}

double norm() {

return dot(\*this);

}

double length() {

return Sqrt(norm());

}

void Input() {

scanf("%lf%lf%lf", &x, &y, &z);

}

void Output() {

printf("%.10f %.10f %.10f\n", x, y, z);

}

};

int mark[1005][1005];

Point info[1005];

int n, cnt;

double mix(const Point &a, const Point &b, const Point &c) {

return a.dot(b.cross(c));

}

double area(int a, int b, int c) {

return ((info[b] - info[a]).cross(info[c] - info[a])).length();

}

double volume(int a, int b, int c, int d) {

return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]);

}

struct Face {

int a, b, c;

Face() {}

Face(int a, int b, int c): a(a), b(b), c(c) {}

int &operator [](int k) {

if (k == 0) return a;

if (k == 1) return b;

return c;

}

};

vector <Face> face;

inline void insert(int a, int b, int c) {

face.push\_back(Face(a, b, c));

}

void add(int v) {

vector <Face> tmp;

int a, b, c;

cnt++;

for (int i = 0; i < SIZE(face); i++) {

a = face[i][0];

b = face[i][1];

c = face[i][2];

if (Sign(volume(v, a, b, c)) < 0)

mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] = mark[a][c] = cnt;

else

tmp.push\_back(face[i]);

}

face = tmp;

for (int i = 0; i < SIZE(tmp); i++) {

a = face[i][0];

b = face[i][1];

c = face[i][2];

if (mark[a][b] == cnt) insert(b, a, v);

if (mark[b][c] == cnt) insert(c, b, v);

if (mark[c][a] == cnt) insert(a, c, v);

}

}

int Find() {

for (int i = 2; i < n; i++) {

Point ndir = (info[0] - info[i]).cross(info[1] - info[i]);

if (ndir == Point()) continue;

swap(info[i], info[2]);

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

if (Sign(volume(0, 1, 2, j)) != 0) {

swap(info[j], info[3]);

insert(0, 1, 2);

insert(0, 2, 1);

return 1;

}

}

return 0;

}

int main() {

for (; scanf("%d", &n) == 1; ) {

for (int i = 0; i < n; i++)

info[i].Input();

sort(info, info + n);

n = unique(info, info + n) - info;

face.clear();

random\_shuffle(info, info + n);

if (Find()) {

memset(mark, 0, sizeof(mark));

cnt = 0;

for (int i = 3; i < n; i++) add(i);

vector<Point> Ndir;

for (int i = 0; i < SIZE(face); ++i) {

Point p = (info[face[i][0]] - info[face[i][1]]).cross(info[face[i][2]] - info[face[i][1]]);

p = p / p.length();

Ndir.push\_back(p);

}

sort(Ndir.begin(), Ndir.end());

int ans = unique(Ndir.begin(), Ndir.end()) - Ndir.begin();

printf("%d\n", ans);

} else {

printf("1\n");

}

}

}

## 三维凸包求重心

double calcDist(const Point &p, int a, int b, int c) {  
    return fabs(mix(info[a] - p, info[b] - p, info[c] - p) / area(a, b, c));  
}  
//compute the minimal distance of center of any faces  
double findDist() {  
    //compute center of mass  
    double totalWeight = 0;  
    Point center(.0, .0, .0);  
    Point first = info[face[0][0]];  
    for (int i = 0; i < SIZE(face); ++i) {  
        Point p = (info[face[i][0]] + info[face[i][1]] + info[face[i][2]] + first) \* .25;  
        double weight = mix(info[face[i][0]] - first, info[face[i][1]] - first, info[face[i][2]] - first);   
        totalWeight += weight;  
        center = center + p \* weight;  
    }  
    center = center / totalWeight;  
    //compute distance   
    double res = 1e100;  
    for (int i = 0; i < SIZE(face); ++i) {  
        res = min(res, calcDist(center, face[i][0], face[i][1], face[i][2]));  
    }  
    return res;  
}

## 随机增量最小覆盖圆

using namespace std;

const double zero=1e-8;

struct point{

double x, y;

point( double xx=0, double yy=0 ){

x=xx; y=yy;

}

point operator +( point &b ){

return point( x+b.x, y+b.y );

}

point operator -( point &b ){

return point( x-b.x, y-b.y );

}

double operator \*( point &b ){

return x\*b.x+y\*b.y;

}

point operator \*( double t ){

return point( x\*t, y\*t );

}

double operator /( point &b ){

return x\*b.y-y\*b.x;

}

point operator /( double t ){

return point( x/t, y/t );

}

};

double sqr( double x ){

return x\*x;

}

double dist( point a, point b ){

return ( sqrt( sqr(a.x-b.x)+sqr(a.y-b.y) ) );

}

struct circle{

point cp;

double r;

circle( point a, point b ){

cp=(a+b)/2;

r=dist( a, b )/2;

}

circle( point a, point b, point c ){

double A,B,C,D,E,F;

A = 2 \* a.x - 2 \* b.x;

B = 2 \* a.y - 2 \* b.y;

C = a.x\*a.x + a.y\*a.y - b.x\*b.x - b.y\*b.y;

D = 2 \* a.x - 2 \* c.x;

E = 2 \* a.y - 2 \* c.y;

F = a.x\*a.x + a.y\*a.y - c.x\*c.x - c.y\*c.y;

cp.x = (C \* E - B \* F) / (A \* E - B \* D);

cp.y = (A \* F - C \* D) / (A \* E - B \* D);

r = dist( a, cp );

}

circle( point a, double b ){

cp=a; r=b;

}

};

bool isin( circle a, point b ){

if ( dist( b, a.cp )-a.r>zero ) return false;

return true;

}

circle ans(point(0,0),1);

int n;

point pp[100010];

void random\_data(){

for ( int i=0; i<n; i++ ){

int j=rand()%n;

point t=pp[i]; pp[i]=pp[j]; pp[j]=t;

}

}

int main(){

int test=0;

scanf("%d", &test);

while ( test-- ){

ans=circle(point(0,0),1);

scanf("%d", &n);

for ( int i=0; i<n; i++ )

scanf("%lf %lf", &pp[i].x, &pp[i].y);

random\_data();

for ( int i=2; i<n; i++ )

if ( ! isin( ans, pp[i] ) ){

ans=circle( pp[0], pp[i] );

for ( int j=1; j<i; j++ )

if ( ! isin( ans, pp[j] ) ){

ans=circle( pp[i], pp[j] );

for ( int k=0; k<j; k++ )

if ( ! isin( ans, pp[k] ) )

ans=circle( pp[i], pp[j], pp[k] );

}

}

printf("%.2f\n", ans.r);

printf("%.2f %.2f\n", ans.cp.x, ans.cp.y);

}

}

## 两圆面积交

struct TC {

double x, y, r;

}a, b, c, d;

double a1;

double sqr(double a)

{

return a \* a;

}

double cirins(TC a, TC b)

{

double ans = 0;

double d = sqrt(sqr(a.x - b.x) + sqr(a.y - b.y));

if (a.r < b.r)

swap(a, b);

if (d + eps > a.r + b.r) return 0;

if (d < a.r - b.r + eps) return pi \* sqr(b.r);

double a1 = acos((sqr(a.r) + d \* d - sqr(b.r)) / 2. / a.r / d);

double a2 = acos((sqr(b.r) + d \* d - sqr(a.r)) / 2. / b.r / d);

ans -= d \* a.r \* sin(a1);

ans += a1 \* sqr(a.r) + a2 \* sqr(b.r);

return ans;

}

int main()

{

scanf("%lf", &a1);

a.x = 0, a.y = 0, b.x = 0, b.y = 0, c.x = 0, c.y = a1, d.x = 0, d.y = a1;

scanf("%lf%lf%lf%lf", &a.r, &b.r, &c.r, &d.r);

printf("%.6f\n", d.r \* d.r \* pi + b.r \* b.r \* pi - cirins(b, d) - (a.r \* a.r \* pi - cirins(a, d)) - (c.r \* c.r \* pi - cirins(b, c)) - cirins(a, c));

}

## 圆的面积模板（n^2logn）

const double eps = 1e-9;

const double PI = acos(-1.0);

int Sign(double x) {

if (x < -eps) return -1;

return x > eps;

}

struct point {

double x, y;

point() {

x = 0;

y = 0;

}

point(double x, double y): x(x), y(y) {

}

point operator +(const point &a) const {

return point(x + a.x, y + a.y);

}

point operator -(const point &a) const {

return point(x - a.x, y - a.y);

}

point operator \*(double k) const {

return point(x \* k, y \* k);

}

point operator /(double k) const {

return point(x / k, y / k);

}

double len() const {

return sqrt(len2());

}

double len2() const {

return x \* x + y \* y;

}

};

double cross(const point &a, const point &b) {

return a.x \* b.y - a.y \* b.x;

}

struct Tcir {

point o;

double r;

Tcir() {

}

Tcir(const point &o, double r): o(o), r(r) {

}

};

const int maxn = 111;

struct Tevent {

point p;

double ang;

int add;

Tevent() {

}

Tevent(const point &\_p, double \_ang, int \_add): p(\_p), ang(\_ang), add(\_add) {

}

bool operator <(const Tevent &a) const {

return ang < a.ang;

}

} eve[maxn \* 2];

int E, cnt;

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

void circleCrossCircle(const Tcir &a, const Tcir &b) {

double l = (a.o - b.o).len2();

double s = ((a.r - b.r) \* (a.r + b.r) / l + 1) \* .5;

double t = sqrt(-(l - sqr(a.r - b.r)) \* (l - sqr(a.r + b.r)) / (l \* l \* 4.));

point dir = b.o - a.o;

point Ndir = point(-dir.y, dir.x);

point aa = a.o + dir \* s + Ndir \* t;

point bb = a.o + dir \* s - Ndir \* t;

double A = atan2(aa.y - a.o.y, aa.x - a.o.x);

double B = atan2(bb.y - a.o.y, bb.x - a.o.x);

eve[E++] = Tevent(bb, B, 1);

eve[E++] = Tevent(aa, A, -1);

if (B > A) {

cnt++;

}

}

bool contain(int x1, int y1, int r1, int x2, int y2, int r2) {

return r1 >= r2 && (x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2) <= (r1 - r2) \* (r1 - r2);

}

bool disjoint(int x1, int y1, int r1, int x2, int y2, int r2) {

return (x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2) >= (r1 + r2) \* (r1 + r2);

}

bool Same(int x1, int y1, int r1, int x2, int y2, int r2) {

return r1 == r2 && x1 == x2 && y1 == y2;

}

bool g[maxn][maxn], Overlap[maxn][maxn];

double Area[maxn];

int cX[maxn], cY[maxn], cR[maxn];

Tcir c[maxn];

int C;

int main() {

scanf("%d", &C);

for (int i = 0; i < C; ++i) {//去掉重复的圆

scanf("%d%d%d", cX+i, cY+i, cR+i);

bool found = false;

for (int j = 0; j < i; ++j) {

if (Same(cX[i], cY[i], cR[i], cX[j], cY[j], cR[j])) {

found = true;

break;

}

}

if (found) {

i--;

C--;

continue;

}

c[i] = Tcir(point(cX[i], cY[i]), cR[i]);

}

for (int i = 0; i <= C; ++i) Area[i] = 0;

for (int i = 0; i < C; ++i) {

for (int j = 0; j < C; ++j) {

Overlap[i][j] = contain(cX[i], cY[i], cR[i], cX[j], cY[j], cR[j]);

}

}

for (int i = 0; i < C; ++i) {

for (int j = 0; j < C; ++j) {

g[i][j] = !(Overlap[i][j] || Overlap[j][i] || disjoint(cX[i], cY[i], cR[i], cX[j], cY[j], cR[j]));

}

}

for (int i = 0; i < C; ++i) {

E = 0;

cnt = 1;

for (int j = 0; j < C; ++j) if (j != i && Overlap[j][i]) cnt++;

for (int j = 0; j < C; ++j) {

if (i != j && g[i][j]) {

circleCrossCircle(c[i], c[j]);

}

}

//cnt表示覆盖次数超过cnt

if (E == 0) {

Area[cnt] += PI \* c[i].r \* c[i].r;

} else {

double counts = 0;

sort(eve, eve + E);

eve[E] = eve[0];

for (int j = 0; j < E; ++j) {

cnt += eve[j].add;

Area[cnt] += cross(eve[j].p, eve[j + 1].p) \* .5;

double theta = eve[j + 1].ang - eve[j].ang;

if (theta < 0) theta += PI \* 2.;

Area[cnt] += theta \* c[i].r \* c[i].r \* .5 - sin(theta) \* c[i].r \* c[i].r \* .5;

}

}

}

printf("%.5f\n", Area[1]);

return 0;

}

## 最小覆盖球

int npoint, nouter;

Tpoint pt[200000], outer[4],res;

double radius,tmp;

inline double dist(Tpoint p1, Tpoint p2) {

double dx=p1.x-p2.x, dy=p1.y-p2.y, dz=p1.z-p2.z;

return ( dx\*dx + dy\*dy + dz\*dz );

}

inline double dot(Tpoint p1, Tpoint p2) {

return p1.x\*p2.x + p1.y\*p2.y + p1.z\*p2.z;

}

void ball() {

Tpoint q[3]; double m[3][3], sol[3], L[3], det;

int i,j;

res.x = res.y = res.z = radius = 0;

switch ( nouter ) {

case 1: res=outer[0]; break;

case 2:

res.x=(outer[0].x+outer[1].x)/2;

res.y=(outer[0].y+outer[1].y)/2;

res.z=(outer[0].z+outer[1].z)/2;

radius=dist(res, outer[0]);

break;

case 3:

for (i=0; i<2; ++i ) {

q[i].x=outer[i+1].x-outer[0].x;

q[i].y=outer[i+1].y-outer[0].y;

q[i].z=outer[i+1].z-outer[0].z;

}

for (i=0; i<2; ++i) for(j=0; j<2; ++j)

m[i][j]=dot(q[i], q[j])\*2;

for (i=0; i<2; ++i ) sol[i]=dot(q[i], q[i]);

if (fabs(det=m[0][0]\*m[1][1]-m[0][1]\*m[1][0])<eps)

return;

L[0]=(sol[0]\*m[1][1]-sol[1]\*m[0][1])/det;

L[1]=(sol[1]\*m[0][0]-sol[0]\*m[1][0])/det;

res.x=outer[0].x+q[0].x\*L[0]+q[1].x\*L[1];

res.y=outer[0].y+q[0].y\*L[0]+q[1].y\*L[1];

res.z=outer[0].z+q[0].z\*L[0]+q[1].z\*L[1];

radius=dist(res, outer[0]);

break;

case 4:

for (i=0; i<3; ++i) {

q[i].x=outer[i+1].x-outer[0].x;

q[i].y=outer[i+1].y-outer[0].y;

q[i].z=outer[i+1].z-outer[0].z;

sol[i]=dot(q[i], q[i]);

}

for (i=0;i<3;++i)

for(j=0;j<3;++j) m[i][j]=dot(q[i],q[j])\*2;

det= m[0][0]\*m[1][1]\*m[2][2]

+ m[0][1]\*m[1][2]\*m[2][0]

+ m[0][2]\*m[2][1]\*m[1][0]

- m[0][2]\*m[1][1]\*m[2][0]

- m[0][1]\*m[1][0]\*m[2][2]

- m[0][0]\*m[1][2]\*m[2][1];

if ( fabs(det)<eps ) return;

for (j=0; j<3; ++j) {

for (i=0; i<3; ++i) m[i][j]=sol[i];

L[j]=( m[0][0]\*m[1][1]\*m[2][2]

+ m[0][1]\*m[1][2]\*m[2][0]

+ m[0][2]\*m[2][1]\*m[1][0]

- m[0][2]\*m[1][1]\*m[2][0]

- m[0][1]\*m[1][0]\*m[2][2]

- m[0][0]\*m[1][2]\*m[2][1]

) / det;

for (i=0; i<3; ++i)

m[i][j]=dot(q[i], q[j])\*2;

}

res=outer[0];

for (i=0; i<3; ++i ) {

res.x += q[i].x \* L[i];

res.y += q[i].y \* L[i];

res.z += q[i].z \* L[i];

}

radius=dist(res, outer[0]);

}

}

void minball(int n) {

ball();

//printf("(%.3f,%.3f,%.3f) %.3f\n", res.x,res.y,res.z,radius);

if ( nouter<4 )

for (int i=0; i<n; ++i)

if (dist(res, pt[i])-radius>eps) {

outer[nouter]=pt[i];

++nouter;

minball(i);

--nouter;

if (i>0) {

Tpoint Tt = pt[i];

memmove(&pt[1], &pt[0], sizeof(Tpoint)\*i);

pt[0]=Tt;

}

}

}

int main(){

scanf("%d",&npoint);

for (int i=0;i<npoint;i++)

scanf("%lf%lf%lf",&pt[i].x,&pt[i].y,&pt[i].z);

random\_shuffle(pt,pt+npoint);

radius=-1;

for (int i=0;i<npoint;i++){

if (dist(res,pt[i])-radius>eps){

nouter=1;

outer[0]=pt[i];

minball(i);

}

}

printf("%.3f\n",sqrt(radius));

}

## 最大空凸包

/\*

算法描述：穷举所要求解的空凸包的最低最左点（先保证最低，再保证最左）。

对于每一个穷举到的点v，进行动态规划，用opt[i][j]表示符合如下限制的凸包中的最大面积：

在凸包上v顺时针过来第一个点是i，并且i顺时针过来第一个点k不在i->j的左手域（k也可能就是j）。

具体如何推的，可以参考程序。

\*/

/\*

Program : The Picnic

Author : Chen Mingcheng

\*/

#include <cstdio>

#include <cmath>

#include <algorithm>

using namespace std;

const int maxn = 100;

const double zero = 1e-8;

struct Vector {

double x, y;

};

inline Vector operator - (Vector a, Vector b) {

Vector c;

c.x = a.x - b.x;

c.y = a.y - b.y;

return c;

}

inline double Sqr(double a) {

return a \* a;

}

inline int Sign(double a) {

if (fabs(a) <= zero) return 0;

return a < 0 ? -1 : 1;

}

inline bool operator < (Vector a, Vector b) {

return Sign(b.y - a.y) > 0 || Sign(b.y - a.y) == 0 && Sign(b.x - a.x) > 0;

}

inline double Max(double a, double b) {

return a > b ? a : b;

}

inline double Length(Vector a) {

return sqrt(Sqr(a.x) + Sqr(a.y));

}

inline double Cross(Vector a, Vector b) {

return a.x \* b.y - a.y \* b.x;

}

Vector dot[maxn], list[maxn];

double opt[maxn][maxn];

int seq[maxn];

int n, len;

double ans;

bool Compare(Vector a, Vector b) {

int temp = Sign(Cross(a, b));

if (temp != 0) return temp > 0;

temp = Sign(Length(b) - Length(a));

return temp > 0;

}

void Solve(int vv) {

int t, i, j, \_len;

for (i = len = 0; i < n; i++)

if (dot[vv] < dot[i]) list[len++] = dot[i] - dot[vv];

for (i = 0; i < len; i++)

for (j = 0; j < len; j++)

opt[i][j] = 0;

sort(list, list + len, Compare);

double v;

for (t = 1; t < len; t++) {

\_len = 0;

for (i = t - 1; i >= 0 && Sign(Cross(list[t], list[i])) == 0; i--);

while (i >= 0) {

v = Cross(list[i], list[t]) / 2;

seq[\_len++] = i;

for (j = i - 1; j >= 0 && Sign(Cross(list[i] - list[t], list[j] - list[t])) > 0; j--);

if (j >= 0) v += opt[i][j];

ans = Max(ans, v);

opt[t][i] = v;

i = j;

}

for (i = \_len - 2; i >= 0; i--)

opt[t][seq[i]] = Max(opt[t][seq[i]], opt[t][seq[i + 1]]);

}

}

int main() {

int t, i;

scanf("%d", &t);

while (t--) {

scanf("%d", &n);

for (i = 0; i < n; i++)

scanf("%lf %lf", &dot[i].x, &dot[i].y);

ans = 0;

for (i = 0; i < n; i++)

Solve(i);

printf("%.1f\n", ans);

}

return 0;

}

## Voronoi

#define Oi(e) ((e)->oi)

#define Dt(e) ((e)->dt)

#define On(e) ((e)->on)

#define Op(e) ((e)->op)

#define Dn(e) ((e)->dn)

#define Dp(e) ((e)->dp)

#define Other(e, p) ((e)->oi == p ? (e)->dt : (e)->oi)

#define Next(e, p) ((e)->oi == p ? (e)->on : (e)->dn)

#define Prev(e, p) ((e)->oi == p ? (e)->op : (e)->dp)

#define V(p1, p2, u, v) (u = p2->x - p1->x, v = p2->y - p1->y)

#define C2(u1, v1, u2, v2) (u1 \* v2 - v1 \* u2)

#define C3(p1, p2, p3) ((p2->x - p1->x) \* (p3->y - p1->y) - (p2->y - p1->y) \* (p3->x - p1->x))

#define Dot(u1, v1, u2, v2) (u1 \* u2 + v1 \* v2)

#define dis(a,b) (sqrt( (a->x - b->x) \* (a->x - b->x) + (a->y - b->y) \* (a->y - b->y) ))

const int maxn = 110024;

const double eps=1e-7;

const int aix=4;

int n, M , k;

struct gEdge

{

int u, v;

double w;

bool operator < (const gEdge &e1) const {return w < e1.w-eps;}

}E[aix \* maxn], MST[maxn];

int b[maxn];

int Find(int x)

{

while (x!=b[x]) {

b[x]=b[b[x]];

x=b[x];

}

return x;

}

void Kruskal()

{

int m1,m2;

memset(b,0,sizeof(b));

for(int i = 0 ;i < n ; i++ ) b[i]=i;

sort(E, E + M);

for(int i = 0, kk = 0; i < M && kk < n - 1; i ++)

{

m1=Find(E[i].u);

m2=Find(E[i].v);

if (m1!=m2) {

b[m1]=m2; MST[kk++] = E[i];

}

}/\*

for(int i = 0; i < n - 1; i++)

printf("%d %d %.3f\n", MST[i].u, MST[i].v, MST[i].w);

\*/

}

struct point

{

double x, y;

int index;

struct edge \*in;

bool operator < (const point &p1) const

{

return x < p1.x-eps || ( abs(x-p1.x)<=eps && y < p1.y-eps);

}

};

struct edge

{

point \*oi, \*dt;

edge \*on, \*op, \*dn, \*dp;

};

point p[maxn], \*Q[maxn];

edge mem[aix \* maxn], \*elist[aix \* maxn];

int nfree;

//memory

void Alloc\_memory()

{

nfree = aix \* n;

edge \*e = mem;

for(int i = 0; i < nfree; i ++) elist[i] = e++;

}

//Add an edge to a ring of edges

void Splice(edge \*a, edge \*b, point \*v)

{

edge \*next;

if(Oi(a) == v) next = On(a), On(a) = b;

else next = Dn(a), Dn(a) = b;

if(Oi(next) == v) Op(next) = b;

else Dp(next) = b;

if(Oi(b) == v) On(b) = next, Op(b) = a;

else Dn(b) = next, Dp(b) = a;

}

//Initialise a new edge

edge \*Make\_edge(point \*u, point \*v)

{

edge \*e = elist[--nfree];

e->on = e->op = e->dn = e->dp = e; e->oi = u; e->dt = v;

if(!u->in) u->in = e; if(!v->in) v->in = e;

return e;

}

//Creates a new edge and adds it to two rings of edges.

edge \*Join(edge \*a, point \*u, edge \*b, point \*v, int side)

{

edge \*e = Make\_edge(u, v);

if(side == 1)

{

if(Oi(a) == u) Splice(Op(a), e, u);

else Splice(Dp(a), e, u);

Splice(b, e, v);

}

else

{

Splice(a, e, u);

if(Oi(b) == v) Splice(Op(b), e, v);

else Splice(Dp(b), e, v);

}

return e;

}

//Remove an edge

void Remove(edge \*e)

{

point \*u = Oi(e), \*v = Dt(e);

if(u->in == e) u->in = e->on; if(v->in == e) v->in = e->dn;

if(Oi(e->on) == u) e->on->op = e->op;

else e->on->dp = e->op;

if(Oi(e->op) == u) e->op->on = e->on;

else e->op->dn = e->on;

if(Oi(e->dn) == v) e->dn->op = e->dp;

else e->dn->dp = e->dp;

if(Oi(e->dp) == v) e->dp->on = e->dn;

else e->dp->dn = e->dn;

elist[nfree++] = e;

}

//Determines the lower tangent of two triangulations

void Low\_tangent(edge \*e\_l, point \*o\_l, edge \*e\_r, point \*o\_r, edge \*\*l\_low, point \*\*OL, edge \*\*r\_low, point \*\*OR)

{

point \*d\_l = Other(e\_l, o\_l), \*d\_r = Other(e\_r, o\_r);

while(1)

{

if(C3(o\_l, o\_r, d\_l) < -eps)

{

e\_l = Prev(e\_l, d\_l);

o\_l = d\_l; d\_l = Other(e\_l, o\_l);

}

else if(C3(o\_l, o\_r, d\_r) < -eps)

{

e\_r = Next(e\_r, d\_r);

o\_r = d\_r; d\_r = Other(e\_r, o\_r);

}

else break;

}

\*OL = o\_l, \*OR = o\_r;

\*l\_low = e\_l, \*r\_low = e\_r;

}

void Merge(edge \*lr, point \*s, edge \*rl, point \*u, edge \*\*tangent)

{

double l1, l2, l3, l4, r1, r2, r3, r4, cot\_L, cot\_R, u1, v1, u2, v2, n1, cot\_n, P1, cot\_P;

point \*O, \*D, \*OR, \*OL;

edge \*B, \*L, \*R;

Low\_tangent(lr, s, rl, u, &L, &OL, &R, &OR);

\*tangent = B = Join(L, OL, R, OR, 0);

O = OL, D = OR;

do

{

edge \*El = Next(B, O), \*Er = Prev(B, D), \*next, \*prev;

point \*l = Other(El, O), \*r = Other(Er, D);

V(l, O, l1, l2); V(l, D, l3, l4); V(r, O, r1, r2); V(r, D, r3, r4);

double cl = C2(l1, l2, l3, l4), cr = C2(r1, r2, r3, r4);

bool BL = cl > eps, BR = cr > eps;

if(!BL && !BR) break;

if(BL)

{

double dl = Dot(l1, l2, l3, l4);

cot\_L = dl / cl;

do

{

next = Next(El, O);

V(Other(next, O), O, u1, v1); V(Other(next, O), D, u2, v2);

n1 = C2(u1, v1, u2, v2);

if(!(n1 > eps)) break;

cot\_n = Dot(u1, v1, u2, v2) / n1;

if(cot\_n > cot\_L) break;

Remove(El);

El = next;

cot\_L = cot\_n;

}

while(1);

}

if(BR)

{

double dr = Dot(r1, r2, r3, r4);

cot\_R = dr / cr;

do

{

prev = Prev(Er, D);

V(Other(prev, D), O, u1, v1); V(Other(prev, D), D, u2, v2);

P1 = C2(u1, v1, u2, v2);

if(!(P1 > eps)) break;

cot\_P = Dot(u1, v1, u2, v2) / P1;

if(cot\_P > cot\_R) break;

Remove(Er);

Er = prev;

cot\_R = cot\_P;

}

while(1);

}

l = Other(El, O); r = Other(Er, D);

if(!BL || (BL && BR && cot\_R < cot\_L)) { B = Join(B, O, Er, r, 0); D = r; }

else { B = Join(El, l, B, D, 0); O = l; }

}

while(1);

}

void Divide(int s, int t, edge \*\*L, edge \*\*R)

{

edge \*a, \*b, \*c, \*ll, \*lr, \*rl, \*rr, \*tangent;

int n = t - s + 1;

if(n == 2) \*L = \*R = Make\_edge(Q[s], Q[t]);

else if(n == 3)

{

a = Make\_edge(Q[s], Q[s + 1]), b = Make\_edge(Q[s + 1], Q[t]);

Splice(a, b, Q[s + 1]);

double v = C3(Q[s], Q[s + 1], Q[t]);

if(v > eps)

{

c = Join(a, Q[s], b, Q[t], 0);

\*L = a; \*R = b;

}

else if(v < -eps)

{

c = Join(a, Q[s], b, Q[t], 1);

\*L = c; \*R = c;

}

else { \*L = a; \*R = b; }

}

else if(n > 3)

{

int split = (s + t) / 2;

Divide(s, split, &ll, &lr); Divide(split + 1, t, &rl, &rr);

Merge(lr, Q[split], rl, Q[split + 1], &tangent);

if(Oi(tangent) == Q[s]) ll = tangent;

if(Dt(tangent) == Q[t]) rr = tangent;

\*L = ll; \*R = rr;

}

}

void Make\_Graph()

{

edge \*start, \*e;

point \*u, \*v;

int i;

for(i = 0; i < n; i++)

{

u = &p[i];

start = e = u->in;

do

{

v = Other(e, u);

if(u < v)

{

E[M].u = u - p, E[M].v = v - p;

E[M++].w = dis(u,v);

if (M>=aix\*maxn) OLE();

}

e = Next(e, u);

}

while(e != start);

}

}

void solve()

{

int i , test;

scanf("%d",&test);

while (test)

{

test--;

n=0;

double ans = -1;

scanf("%d", &n);

for(i=0; i<n;i++) {

scanf("%lf%lf",&p[i].x,&p[i].y);

p[i].index=i;

p[i].in=NULL;

}

Alloc\_memory();

if(n == 1 || n==0 ){ continue;} // else RE

sort(p, p + n);

//=========点不能有重点，有的话不满足voronoi图的性质了

for(i = 0; i < n; i++) Q[i] = p + i;

edge \*L, \*R;

Divide(0, n - 1, &L, &R);

M = 0;

Make\_Graph();

Kruskal();

// puts("---------------------");

}

}

int main()

{

freopen("input.txt","r",stdin);

freopen("output.txt","w",stdout);

solve();

return 0;

}

## 三角形的心

//三角形

#include <cstdio>

#include <cstdlib>

#include <cmath>

#define SQR(x) ((x)\*(x))

//传入的参数point a,b,c; 三角形顶点

double area(point a,point b,point c) //面积

{

return(fabs(det(b-a,c-a))/2);

}

point barycenter(point a,point b,point c) //重心

{

return(point((a.x+b.x+c.x)/3.0,(a.y+b.y+c.y)/3.0));

}

point orthocenter(point a,point b,point c) //垂心

{

double d,dx,dy;

d=(c.x-b.x)\*(c.y-a.y)-(c.x-a.x)\*(c.y-b.y);

dx=(a.y\*(c.y-b.y)+a.x\*(c.x-b.x))\*(c.y-a.y)-(b.y\*(c.y-a.y)+b.x\*(c.x-a.x))\*(c.y-b.y);

dy=(c.x-b.x)\*(b.y\*(c.y-a.y)+b.x\*(c.x-a.x))-(c.x-a.x)\*(a.y\*(c.y-b.y)+a.x\*(c.x-b.x));

return(point(dx/d,dy/d));

}

point circumcenter(point a,point b,point c) //外心

{

double A,B,C;

A=dist(b,c),B=dist(a,c),C=dist(a,b);

double P,Q;

P=(SQR(A)+SQR(B)+SQR(C))/2.0;

Q=1.0/(1/(P-SQR(A))+1/(P-SQR(B))+1/(P-SQR(C)));

double R=sqrt(P-Q)/2; //R为外接圆半径，需要时可用，否则可删去

double d1,d2,d3;

d1=Q/(P-SQR(A)),d2=Q/(P-SQR(B)),d3=Q/(P-SQR(C));

return((1-d1)/2.0\*a+(1-d2)/2.0\*b+(1-d3)/2.0\*c);

}

point incenter(point a,point b,point c)

{

double A,B,C;

A=dist(b,c),B=dist(a,c),C=dist(a,b);

double r=2\*area(a,b,c)/(A+B+C); //r为内切圆半径，需要时可用 ，否则可删去

return(point((A\*a.x+B\*b.x+C\*c.x)/(A+B+C),(A\*a.y+B\*b.y+C\*c.y)/(A+B+C)));

}

## 四边形费马点

//BEGIN

//POINT CLASS

typedef complex <double> Tpoint;

const double eps = 1e-8;

const double sqrt3 = sqrt(3.0);

istream& operator >>(istream& cin, Tpoint &p) {

double x, y;

cin >> x >> y;

p = Tpoint(x, y);

return cin;

}

ostream& operator <<(ostream& cout, const Tpoint &p) {

cout << "(" << p.real() << ", " << p.imag() << ")";

return cout;

}

int Sign(double x) {

return fabs(x) < eps ? 0 : x > 0 ? 1 : -1;

}

bool operator ==(const Tpoint &a, const Tpoint &b) {

return !Sign(a.real() - b.real()) && !Sign(b.imag() - a.imag());

}

bool cmp(const Tpoint &a, const Tpoint &b) {

return a.real() < b.real() - eps || (a.real() < b.real() + eps && a.imag() < b.imag());

}

double cross(const Tpoint &a, const Tpoint &b) {

return (conj(a) \* b).imag();

}

double dot(const Tpoint &a, const Tpoint &b) {

return (conj(a) \* b).real();

}

double cross(const Tpoint &a, const Tpoint &b, const Tpoint &c) {

return cross(b - a, c - a);

}

double dot(const Tpoint &a, const Tpoint &b, const Tpoint &c) {

return dot(b - a, c - a);

}

Tpoint unit(const Tpoint &a) {

return a / abs(a);

}

Tpoint intersect(const Tpoint &a, const Tpoint &b, const Tpoint &c, const Tpoint &d) {

double k1 = cross(a, b, c), k2 = cross(a, b, d);

if (Sign(k1 - k2)) {

return (c \* k2 - d \* k1) / (k2 - k1);

} else {

return Tpoint(0.0, 0.0);

}

}

Tpoint rotate(const Tpoint &a, const Tpoint &b, const Tpoint &c) {

Tpoint d = b - a;

d = Tpoint(-d.imag(), d.real());

if (Sign(cross(a, b, c)) == Sign(cross(a, b, a + d))) {

d \*= -1.0;

}

return unit(d);

}

//END

Tpoint p[10], a[10], b[10];

int N, T;

double totlen(const Tpoint &p, const Tpoint &a, const Tpoint &b, const Tpoint &c) {

return abs(p - a) + abs(p - b) + abs(p - c);

}

double fermat(const Tpoint &x, const Tpoint &y, const Tpoint &z, Tpoint &cp) {

a[0] = a[3] = x;

a[1] = a[4] = y;

a[2] = a[5] = z;

double len = 1e100, len2;

for (int i = 0; i < 3; i ++) {

len2 = totlen(a[i], x, y, z);

if (len2 < len) {

len = len2;

cp = a[i];

}

}

for (int i = 0; i < 3; i ++) {

b[i] = rotate(a[i + 1], a[i], a[i + 2]);

b[i] = (a[i + 1] + a[i]) / 2.0 + b[i] \* (abs(a[i + 1] - a[i]) \* sqrt3 / 2.0);

}

b[3] = b[0];

Tpoint cp2 = intersect(b[0], a[2], b[1], a[3]);

len2 = totlen(cp2, x, y, z);

if (len2 < len) {

len = len2;

cp = cp2;

}

return len;

}

double getans(const Tpoint &a) {

double len = 0;

for (int i = 0; i < N; i ++) len += abs(a - p[i]);

return len;

}

double mindist(const Tpoint &p, const Tpoint &a, const Tpoint &b, const Tpoint &c, const Tpoint &d) {

return min(min(abs(p - a), abs(p - b)), min(abs(p - c), abs(p - d)));

}

int main() {

N = 4;

for (cin >> T; T; T --) {

for (int i = 0; i < N; i ++) {

cin >> p[i];

}

Tpoint cp;

double ret = 1e100;

for (int i = 0; i < N; i ++) ret = min(ret, getans(p[i]));

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

for (int j = 1; j < N; j ++) {

if (j != i) {

for (int k = 1; k < N; k ++) {

if (k != i && k != j) {

ret = min(ret, abs(p[0] - p[i]) + abs(p[j] - p[k]) +

min(min(abs(p[0] - p[j]), abs(p[0] - p[k])), min(abs(p[i] - p[j]), abs(p[i] - p[k]))));

ret = min(ret, getans(intersect(p[0], p[i], p[j], p[k])));

}

}

}

}

}

for (int i = 0; i < N; i ++) {

for (int j = i + 1; j < N; j ++) {

for (int k = j + 1; k < N; k ++) {

double len = fermat(p[i], p[j], p[k], cp);

ret = min(ret, len + mindist(p[6 - i - j - k], p[i], p[j], p[k], cp));

}

}

}

sort(p, p + N, cmp);

Tpoint cp1, cp2;

double len\_cur, len\_before;

double len1, len2, len;

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

cp1 = (p[0] + p[i]) / 2.0;

int j, k;

for (j = 1; j < N && j == i; j ++);

k = 6 - i - j;

len\_before = 1e100;

for (;;) {

len1 = fermat(cp1, p[j], p[k], cp2);

len1 = fermat(cp2, p[0], p[i], cp1);

len = len1 + abs(cp2 - p[j]) + abs(cp2 - p[k]);

if (len < len\_before - (1e-6)) {

len\_before = len;

} else {

break;

}

}

ret = min(ret, len\_before);

}

printf("%.4f\n", ret);

}

return 0;

}

## 最近点对

#include <iostream>

#include <cstdio>

#include <cstring>

#include <algorithm>

#include <cmath>

using namespace std;

const int maxn = 101000;

const double zero = 1e-7;

struct dot{

double x, y;

dot(const double & v1 = 0, const double & v2 = 0): x(v1), y(v2){}

dot operator + (const dot & b){

return dot(x+b.x, y+b.y);

}

dot operator - (const dot & b){

return dot(x-b.x, y-b.y);

}

double dis(){

return sqrt(x\*x+y\*y);

}

};

dot a[maxn];

int n, ys[maxn], tmp[maxn];

double ans;

void init(){

int i;

for (i=0; i<n; ++i) scanf("%lf%lf", &a[i].x, &a[i].y);

}

inline int dcmp(const double & v){

if (v<-zero) return -1;

return v>zero;

}

bool xcmp(const dot & a, const dot & b){

return dcmp(a.x-b.x)<0;

}

bool ystmp(int v1, int v2){

return a[v1].y<a[v2].y;

}

double minimal\_dis(dot \* c, int n, int \* ys){

int i, j, mid = n/2, cnt = 0;

double ret = 1e+20, xmid = c[mid].x;

if (n<20){

for (i=0; i<n; ++i)

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

if (dcmp((c[i]-c[j]).dis()-ret)<0) ret=(c[i]-c[j]).dis();

if (a[ys[i]].y>a[ys[j]].y) swap(ys[i], ys[j]);

}

return ret;

}

ret = min(minimal\_dis(c, mid, ys), minimal\_dis(c+mid, n-mid, ys+mid));

merge(ys, ys+mid, ys+mid, ys+n, tmp, ystmp);

copy(tmp, tmp+n, ys);

for (i=0; i<n; ++i) {

while (i<n && dcmp(fabs(a[ys[i]].x-xmid)-ret)>0) ++i;

j=i+1; cnt=0;

while (j<n && dcmp(a[ys[j]].y-a[ys[i]].y-ret)<=0) {

if (dcmp(fabs(a[ys[j]].x-xmid)-ret)<=0){

ret=min(ret, (a[ys[i]]-a[ys[j]]).dis());

if (++cnt>=10) break;

}

++j;

}

}

return ret;

}

void work(){

int i;

sort(a, a+n, xcmp);

for (i=0; i<n; ++i) ys[i]=i;

ans=minimal\_dis(a, n, ys);

}

void print(){

printf("%.4f\n", fabs(ans));

}

int main(){

while (scanf("%d", &n)==1 && n){

init();

work();

print();

}

return 0;

}

## 最远点对

#include <cstdio>

#include <algorithm>

#include <cmath>

using namespace std;

struct point{

int x, y;

point( int xx=0, int yy=0 ){

x=xx; y=yy;

}

point operator +( const point &b )const{

return point( x+b.x, y+b.y );

}

point operator -( const point &b )const{

return point( x-b.x, y-b.y );

}

double operator \*( const point &b )const{

return x\*b.x+y\*b.y;

}

double operator /( const point &b )const{

return x\*b.y-y\*b.x;

}

};

int sqr( int x ){

return x\*x;

}

int dist( point a, point b ){

return sqr(a.x-b.x)+sqr(a.y-b.y);

}

bool cmp( point a, point b ){

return (a.y<b.y || a.y==b.y && a.x<b.x);

}

point conv[100000];

int totco;

int n;

//凸包

void convex( point p[], int n ){

sort( p, p+n, cmp );

conv[0]=p[0]; conv[1]=p[1]; totco=2;

for ( int i=2; i<n; i++ ){

while ( totco>1 && (conv[totco-1]-conv[totco-2])/(p[i]-conv[totco-2])<=0 ) totco--;

conv[totco++]=p[i];

}

int limit=totco;

for ( int i=n-1; i>=0; i-- ){

while ( totco>limit && (conv[totco-1]-conv[totco-2])/(p[i]-conv[totco-2])<=0 ) totco--;

conv[totco++]=p[i];

}

}

point pp[100000];

int main(){

scanf("%d", &n);

for ( int i=0; i<n; i++ )

scanf("%d %d", &pp[i].x, &pp[i].y);

convex( pp, n );

n=totco;

for ( int i=0; i<n; i++ ) pp[i]=conv[i];

/\*for ( int i=0; i<n; i++ )

printf("%d %d\n", pp[i].x, pp[i].y);\*/

n--;

int ans=0;

for ( int i=0; i<n; i++ )

pp[n+i]=pp[i];

int now=1;

for ( int i=0; i<n; i++ ){

point tt=point( pp[i+1]-pp[i] );

while ( now<2\*n-2 && tt/(pp[now+1]-pp[now])>0 ) now++;

if ( dist( pp[i], pp[now] )>ans ) ans=dist( pp[i], pp[now] );

if ( dist( pp[i+1], pp[now] )>ans ) ans=dist( pp[i+1], pp[now] );

//printf("%d %d\n", i, now);

}

printf("%d\n", ans);

}

## 经纬度求球面最短距离

//lati 为纬度 longi为经度 R为半径

double Dist(double lati1,double longi1,double lati2,double longi2,double R)

{

double pi=acos(-1.0);

lati1\*=pi/180,longi1\*=pi/180,lati2\*=pi/180,longi2\*=pi/180;

double x1=cos(lati1)\*sin(longi1),y1=cos(lati1)\*cos(longi1),z1=sin(lati1);

double x2=cos(lati2)\*sin(longi2),y2=cos(lati2)\*cos(longi2),z2=sin(lati2);

double theta=acos(x1\*x2+y1\*y2+z1\*z2);

return(R\*theta);

}

## 长方体表面两点最短距离

int r;

void turn(int i, int j, int x, int y, int z,int x0, int y0, int L, int W, int H) {

if (z==0) {

int R = x\*x+y\*y;

if (R<r) r=R;

}

else{

if(i>=0 && i< 2)

turn(i+1, j, x0+L+z, y, x0+L-x, x0+L, y0, H, W, L);

if(j>=0 && j< 2)

turn(i, j+1, x, y0+W+z, y0+W-y, x0, y0+W, L, H, W);

if(i<=0 && i>-2)

turn(i-1, j, x0-z, y, x-x0, x0-H, y0, H, W, L);

if(j<=0 && j>-2)

turn(i, j-1, x, y0-z, y-y0, x0, y0-H, L, H, W);

}

}

int main(){

int L, H, W, x1, y1, z1, x2, y2, z2;

cin >> L >> W >> H >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;

if (z1!=0 && z1!=H)

if (y1==0 || y1==W)

swap(y1,z1), std::swap(y2,z2), std::swap(W,H);

else

swap(x1,z1), std::swap(x2,z2), std::swap(L,H);

if (z1==H) z1=0, z2=H-z2;

r=0x3fffffff; turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);

cout<<r<<endl;

return 0;

}

## Farmland

const int mx = 210;

const double eps = 1e-8;

struct TPoint { double x, y;} p[mx];

struct TNode { int n, e[mx];} a[mx];

bool visit[mx][mx], valid[mx];

int l[mx][2], n, m, tp, ans, now, test;

double area;

int dcmp(double x) { return x < eps ? -1 : x > eps; }

int cmp(int a, int b){

return dcmp(atan2(p[a].y - p[now].y, p[a].x - p[now].x) - atan2(p[b].y - p[now].y, p[b].x - p[now].x)) < 0;

}

double cross(const TPoint&a, const TPoint&b){ return a.x \* b.y - b.x \* a.y;}

void init();

void work();

bool check(int, int);

int main()

{

scanf("%d", &test);

while(test--) {

init();

work();

}

return 0;

}

void init()

{

memset(visit, 0, sizeof(visit));

memset(p, 0, sizeof(p));

memset(a, 0, sizeof(a));

scanf("%d", &n);

for(int i = 0; i < n; i++) {

scanf("%d", &a[i].n);

scanf("%lf%lf", &p[i].x, &p[i].y);

scanf("%d", &a[i].n);

for(int j = 0; j < a[i].n; j++) {

scanf("%d", &a[i].e[j]);

a[i].e[j]--;

}

}

scanf("%d", &m);

for(now = 0; now < n; now++) sort(a[now].e, a[now].e + a[now].n, cmp);

}

void work()

{

ans = 0;

for(int i = 0; i < n; i++)

for(int j = 0; j < a[i].n; j++) if(!visit[i][a[i].e[j]])

if(check(i, a[i].e[j])) ans++;

printf("%d\n", ans);

}

bool check(int b1, int b2)

{

area = 0;

l[0][0] = b1;

l[0][1] = b2;

for(tp = 1; ; tp++) {

visit[l[tp - 1][0]][l[tp - 1][1]] = 1;

area += cross(p[l[tp - 1][0]], p[l[tp - 1][1]]);

int k, r(l[tp][0] = l[tp - 1][1]);

for(k = 0; k < a[r].n; k++) if(a[r].e[k] == l[tp - 1][0]) break;

l[tp][1] = a[r].e[(k + a[r].n - 1) % a[r].n];

if(l[tp][0] == b1 && l[tp][1] == b2) break;

}

if(dcmp(area) < 0 || tp < 3 || tp != m) return 0;

fill\_n(valid, n, 0);

for(int i = 0; i < tp; i++) {

if(valid[l[i][0]]) return 0;

valid[l[i][0]] = 1;

}

return 1;

}

# 图论

## 最大团

Int g[][]为图的邻接矩阵。

MC(V)表示点集V的最大团

令Si={vi, vi+1, ..., vn}, mc[i]表示MC(Si)

倒着算mc[i]，那么显然MC(V)=mc[1]

此外有mc[i]=mc[i+1] or mc[i]=mc[i+1]+1

void init(){

int i, j;

for (i=1; i<=n; ++i) for (j=1; j<=n; ++j) scanf("%d", &g[i][j]);

}

void dfs(int size){

int i, j, k;

if (len[size]==0) {

if (size>ans) {

ans=size; found=true;

}

return;

}

for (k=0; k<len[size] && !found; ++k) {

if (size+len[size]-k<=ans) break;

i=list[size][k];

if (size+mc[i]<=ans) break;

for (j=k+1, len[size+1]=0; j<len[size]; ++j)

if (g[i][list[size][j]]) list[size+1][len[size+1]++]=list[size][j];

dfs(size+1);

}

}

void work(){

int i, j;

mc[n]=ans=1;

for (i=n-1; i; --i) {

found=false;

len[1]=0;

for (j=i+1; j<=n; ++j) if (g[i][j]) list[1][len[1]++]=j;

dfs(1);

mc[i]=ans;

}

}

void print(){

printf("%d\n", ans);

}

## 极大团计数

Bool g[][] 为图的邻接矩阵，图点的标号由1至n。

【代码】

void dfs(int size){

int i, j, k, t, cnt, best = 0;

bool bb;

if (ne[size]==ce[size]){

if (ce[size]==0) ++ans;

return;

}

for (t=0, i=1; i<=ne[size]; ++i) {

for (cnt=0, j=ne[size]+1; j<=ce[size]; ++j)

if (!g[list[size][i]][list[size][j]]) ++cnt;

if (t==0 || cnt<best) t=i, best=cnt;

}

if (t && best<=0) return;

for (k=ne[size]+1; k<=ce[size]; ++k) {

if (t>0){

for (i=k; i<=ce[size]; ++i) if (!g[list[size][t]][list[size][i]]) break;

swap(list[size][k], list[size][i]);

}

i=list[size][k];

ne[size+1]=ce[size+1]=0;

for (j=1; j<k; ++j)if (g[i][list[size][j]]) list[size+1][++ne[size+1]]=list[size][j];

for (ce[size+1]=ne[size+1], j=k+1; j<=ce[size]; ++j)

if (g[i][list[size][j]]) list[size+1][++ce[size+1]]=list[size][j];

dfs(size+1);

++ne[size];

--best;

for (j=k+1, cnt=0; j<=ce[size]; ++j) if (!g[i][list[size][j]]) ++cnt;

if (t==0 || cnt<best) t=k, best=cnt;

if (t && best<=0) break;

}

}

void work(){

int i;

ne[0]=0; ce[0]=0;

for (i=1; i<=n; ++i) list[0][++ce[0]]=i;

ans=0;

dfs(0);

}

## 2-SAT

const int maxn = 3000;

inline int Par(int a) {

return a ^ 1;

}

vector<int> ori[maxn], rev[maxn];

int code[maxn], seq[maxn];

int n, m, cnt;

void Add\_Link(int a, int b) {

ori[a].push\_back(b);

rev[b].push\_back(a);

}

void DFS\_1(int v) {

code[v] = 1;

int i;

for (i = ori[v].size() - 1; i >= 0; i--)

if (!code[ori[v][i]]) DFS\_1(ori[v][i]);

seq[cnt++] = v;

}

void DFS\_2(int v) {

code[v] = cnt;

int i;

for (i = rev[v].size() - 1; i >= 0; i--)

if (code[rev[v][i]] == -1) DFS\_2(rev[v][i]);

}

void Work() {

int i;

for (i = 0; i < n \* 2; i++)

code[i] = 0;

cnt = 0;

for (i = 0; i < n \* 2; i++)

if (!code[i]) DFS\_1(i);

reverse(seq, seq + cnt);

cnt = 0;

for (i = 0; i < n \* 2; i++)

code[i] = -1;

for (i = 0; i < n \* 2; i++)

if (code[seq[i]] == -1) {

DFS\_2(seq[i]);

cnt++;

}

for (i = 0; i < n \* 2; i++)

if (code[i] == code[Par(i)]) {

printf("No\n");

return;

}

printf("Yes\n");

for (i = 0; i < n; i++)

if (code[i \* 2] > code[i \* 2 + 1]) printf("%d ", i + 1);

printf("0\n");

}

## KM

int n,b[MAXN],dx[MAXN],dy[MAXN],slack[MAXN],a[MAXN][MAXN];

bool f[MAXN],g[MAXN];

bool hungary(int x)

{

if (!x)

return(true);

f[x]=true;

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

{

if (g[i])

continue;

int t=dx[x]+dy[i]-a[x][i];

if (!t)

{

g[i]=true;

if (hungary(b[i]))

{

b[i]=x;

return(true);

}

}

else if (t<slack[i])

slack[i]=t;

}

return(false);

}

int main()

{

memset(dx,0,sizeof(dx));

memset(dy,0,sizeof(dy));

scanf("%d",&n);

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

for (int j=1;j<=n;j++)

{

scanf("%d",&a[i][j]);

if (a[i][j]>dx[i])

dx[i]=a[i][j];

}

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

{

memset(slack,63,sizeof(slack));

memset(f,0,sizeof(f));

memset(g,0,sizeof(g));

while (!hungary(i))

{

int d=inf;

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

if (!g[i] && slack[i]<d)

d=slack[i];

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

{

if (f[i])

dx[i]-=d;

if (g[i])

dy[i]+=d;

}

memset(f,0,sizeof(f));

memset(g,0,sizeof(g));

}

}

}

## 无向图最小割

#include <cstdio>

#include <algorithm>

using namespace std;

const int maxn = 600;

const int inf = 0x7fffffff;

int cost[maxn][maxn];

int seq[maxn], len[maxn];

bool used[maxn];

int n, m, pop, ans;

void Init() {

int i, j, a, b, c;

for (i = 0; i < n; i++)

for (j = 0; j < n; j++)

cost[i][j] = 0;

for (i = 0; i < m; i++) {

scanf("%d %d %d", &a, &b, &c);

cost[a][b] += c;

cost[b][a] += c;

}

pop = n;

for (i = 0; i < n; i++)

seq[i] = i;

}

void Work() {

ans = inf;

int i, j, k, l, mm, sum, pk;

while (pop > 1) {

for (i = 1; i < pop; i++)

used[seq[i]] = 0;

used[seq[0]] = 1;

for (i = 1; i < pop; i++)

len[seq[i]] = cost[seq[0]][seq[i]];

pk = 0;

mm = -inf;

k = -1;

for (i = 1; i < pop; i++)

if (len[seq[i]] > mm) {

mm = len[seq[i]];

k = i;

}

for (i = 1; i < pop; i++) {

used[seq[l = k]] = 1;

if (i == pop - 2) pk = k;

if (i == pop - 1) break;

mm = -inf;

for (j = 1; j < pop; j++)

if (!used[seq[j]]) {

if ((len[seq[j]] += cost[seq[l]][seq[j]]) > mm) {

mm = len[seq[j]];

k = j;

};

}

}

sum = 0;

for (i = 0; i < pop; i++)

if (i != k) sum += cost[seq[k]][seq[i]];

ans = min(ans, sum);

for (i = 0; i < pop; i++)

cost[seq[k]][seq[i]] = cost[seq[i]][seq[k]] += cost[seq[pk]][seq[i]];

seq[pk] = seq[--pop];

}

printf("%d\n", ans);

}

int main() {

while (scanf("%d %d", &n, &m) == 2) {

Init();

Work();

}

return 0;

}

========================我是分割线==================================

#include <iostream>

#include <algorithm>

using namespace std;

#define initSet(n,Arr) for(int i=0;i<n;++i)Arr[i]=i;

#define MAX 1<<30;

int graph[600][600];

// Stoer-Wagner Algorithm

int globalMinCut(int n){

// A is A set for Stoer-Wagner Algorithm

bool\* A=new bool[n];

// V is vertex index

int\* V=new int[n];

int\* W=new int[n];

initSet(n,V);

int best=MAX;

while(n>1){

//the most tightly connected vertex.

int maxj=1;

// initialize set A and other vertex's weight

A[V[0]] = true;

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

A[V[i]]=false;

W[i]=graph[V[0]][V[i]];

if(W[i]>W[maxj])

maxj=i;

}

// find a min-cut

int prev=0,buf=n;

while(--buf){

// add it to A

A[V[maxj]]=true;

if(buf==1){

// update min cut

best=min(best,W[maxj]);

// merge prev and last vertex

for(int k=0; k<n; ++k)

graph[V[k]][V[prev]]=(graph[V[prev]][V[k]]

+=graph[V[maxj]][V[k]]);

V[maxj]=V[--n];

}

prev=maxj;

maxj=-1;

// update the weights

for(int j=1; j<n; ++j)

if(!A[V[j]]){

W[j]+=graph[V[prev]][V[j]];

if(maxj<0 || W[j]>W[maxj])

maxj=j;

}

}

}

delete[] A;

delete[] V;

delete[] W;

return best;

}

int main(){

// n - vertex number

// m - edge number

int n,m;

while(scanf("%d %d",&n,&m)==2){

memset(graph,0,sizeof(graph)/sizeof(bool));

// v-w is an edge with c weight

int v,w,c;

while(m--){

scanf("%d %d %d",&v,&w,&c);

graph[v][w]+=c;

graph[w][v]+=c;

}

// output min cut

printf("%d\n",globalMinCut(n));

}

}

## 弦图相关

1.团数 ≤ 色数

2.最大独立集数 ≤ 最小团覆盖数

3.任何一个弦图都至少有一个单纯点，不是完全图的弦图至少有两个不相邻的单纯点。

4.设第i个点在弦图的完美消除序列第p(i)个。令N(v) = {w | w与v相邻且p(w) > p(v)}弦图的极大团一定是v∪N(v)的形式。

5.弦图最多有n个极大团。

6.设next(v) 表示N(v)中最前的点。令w\*表示所有满足A∈B的w中最后的一个点。判断v∪N(v)是否为极大团,只需判断是否存在一个w，满足Next(w) = v且|N(v)| + 1 ≤ |N(w)|即可。

7.最小染色：完美消除序列从后往前依次给每个点染色，给每个点染上可以染的最小的颜色。//团数=色数

8.最大独立集：完美消除序列从前往后能选就选。

9.最小团覆盖：设最大独立集为{p1 , p2 , …, pt}，则{p1∪N(p1), …, pt∪N(pt)}为最小团覆盖。 //最大独立集数 = 最小团覆盖数!!!

## 弦图完美消除序列

/\*

弦图的完美消除序列

O(mlogn) 可以做到 O(n+m)

\*/

#include <iostream>

using namespace std;

#define maxn 1005

#define maxm 2000005

int head[maxn],heap[maxn],l[maxn],hz,Link[maxn];

int vtx[maxm],next[maxm],tot,n,m,A[maxn];

bool map[maxn][maxn];

inline void Add(int a,int b)

{

vtx[tot]=b;

next[tot]=head[a];

head[a]=tot++;

}

inline void sink(int x)

{

int mid=x\*2;

while (mid<=hz)

{

if (mid+1<=hz && l[heap[mid+1]]>l[heap[mid]]) ++mid;

if (l[heap[x]]<l[heap[mid]])

{

swap(Link[heap[x]],Link[heap[mid]]);

swap(heap[x],heap[mid]);

}else break;

x=mid;

mid=x\*2;

}

}

inline void up(int x)

{

for (int mid=x/2;mid>0;mid=x/2)

{

if (l[heap[mid]]<l[heap[x]])

{

swap(Link[heap[x]],Link[heap[mid]]);

swap(heap[x],heap[mid]);

}else break;

x=mid;

}

}

int main()

{

for (;scanf("%d%d",&n,&m) && (m+n);)

{

tot=2;

memset(map,false,sizeof(map));

memset(head,0,sizeof(head));

for (int i=0;i<m;++i)

{

int a,b;

scanf("%d%d",&a,&b);

--a;--b;

map[a][b]=map[b][a]=true;

Add(a,b);

Add(b,a);

}

memset(l,0,sizeof(l));

hz=0;

for (int i=0;i<n;++i)

{

Link[i]=++hz;

heap[hz]=i;

}

for (int i=n;i>0;--i)

{

int v=-1;

int u=heap[1];

//序列的第i项就是u

Link[u]=-1;

Link[heap[hz]]=1;

heap[1]=heap[hz--];

sink(1);

for (int p=head[u];p;p=next[p])

if (Link[vtx[p]]!=-1)

{

++l[vtx[p]];

up(Link[vtx[p]]);

}else

{

if (v==-1) v=vtx[p];

else

{

if (!map[v][vtx[p]])

{

printf("Imperfect\n");

//判定不是弦图

goto answer;

}

}

}

}

printf("Perfect\n");

answer:;

printf("\n");

}

return 0;

}

## 带花树

#include <cstdio>

#include <vector>

using namespace std;

#define maxn 301

vector<int> link[maxn];

int n;

int match[maxn];

int Queue[maxn], head, tail;

int pred[maxn], base[maxn];

bool InQueue[maxn], InBlossom[maxn];

int start, finish;

int newbase;

void push(int u) {

Queue[tail++] = u; InQueue[u] = true;

}

int pop() {

return Queue[head++];

}

int FindCommonAncestor(int u, int v) {

bool InPath[maxn];

for (int i = 0; i < n; i++)

InPath[i] = 0;

while(true) {

u = base[u];

InPath[u] = true;

if(u == start) break;

u = pred[match[u]];

}

while(true) {

v = base[v];

if(InPath[v]) break;

v = pred[match[v]];

}

return v;

}

void ResetTrace(int u) {

int v;

while(base[u] != newbase) {

v = match[u];

InBlossom[base[u]] = InBlossom[base[v]] = true;

u = pred[v];

if(base[u] != newbase) pred[u] = v;

}

}

void BlossomContract(int u, int v) {

newbase = FindCommonAncestor(u, v);

for (int i = 0; i < n; i++)

InBlossom[i] = 0;

ResetTrace(u); ResetTrace(v);

if(base[u] != newbase) pred[u] = v;

if(base[v] != newbase) pred[v] = u;

for(int i = 0; i < n; ++i)

if(InBlossom[base[i]]) {

base[i] = newbase;

if(!InQueue[i]) push(i);

}

}

bool FindAugmentingPath(int u) {

bool found = false;

for(int i = 0; i < n; ++i) pred[i] = -1, base[i] = i;

for (int i = 0; i < n; i++)

InQueue[i] = 0;

start = u; finish = -1;

head = tail = 0;

push(start);

while(head < tail) {

int u = pop();

for(int i = link[u].size() - 1; i >= 0; i--) {

int v = link[u][i];

if(base[u] != base[v] && match[u] != v)

if(v == start || (match[v] >= 0 && pred[match[v]] >= 0))

BlossomContract(u, v);

else if(pred[v] == -1) {

pred[v] = u;

if(match[v] >= 0)

push(match[v]);

else {

finish = v;

return true;

}

}

}

}

return found;

}

void AugmentPath() {

int u, v, w;

u = finish;

while(u >= 0) {

v = pred[u];

w = match[v];

match[v] = u;

match[u] = v;

u = w;

}

}

void FindMaxMatching() {

for(int i = 0; i < n; ++i) match[i] = -1;

for(int i = 0; i < n; ++i)

if(match[i] == -1)

if(FindAugmentingPath(i))

AugmentPath();

}

## 最小树形图

#include<iostream>

#include<cstring>

#include<cstdio>

#include<cmath>

using namespace std;

#define INF 99999999

#define min( a, b ) ( (a)< (b)?(a): (b) )

struct point

{

double x;

double y;

}p[200];

int pre[200];//记录该节点的前驱

double graph[200][200], ans;//图数组和结果

bool visit[110], circle[110];//visit记录该点有没有被访问过，circle记录改点是不是在一个圈里

int n, m, root;//顶点数+边数+根节点标号

void dfs( int t )//一个深度优先搜索，搜索出一个最大的联通空间

{

int i;

visit[t]= true;

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

{

if( !visit[i] && graph[t][i]!= INF )

dfs( i );

}

}

bool check()//这个函数用来检查最小树形图是否存在，即如果存在，那么一遍dfs后，应该可以遍历到所有的节点

{

memset( visit, false, sizeof(visit) );

dfs( root );

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

{

if( !visit[i] )

return false;

}

return true;

}

double dist( int i, int j )

{

return sqrt( (p[i].x-p[j].x)\*(p[i].x-p[j].x)+(p[i].y-p[j].y)\*(p[i].y-p[j].y) );

}

int exist\_circle()//判断图中是不是存在有向圈

{

int i;

int j;

root= 1; pre[root]= root;

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

{

if( !circle[i] && i!= root )

{

pre[i]= i; graph[i][i]= INF;

for(j= 1; j<= n; ++j )

{

if( !circle[j] && graph[j][i]< graph[pre[i]][i] )

pre[i]= j;

}

}

}//这个for循环负责找出所有非根节点的前驱节点

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

{

if( circle[i] )

continue;

memset( visit, false, sizeof(visit) );

int j= i;

while( !visit[j] )

{

visit[j]= true;

j= pre[j];

}

if( j== root )

continue;

return j;

}//找圈过程，最后返回值是圈中的一个点

return -1;//如果没有圈，返回-1

}

void update( int t )//缩圈之后更新数据

{

int i;

int j;

ans+= graph[pre[t]][t];

for(i=pre[t]; i!= t; i= pre[i] )

{

ans+= graph[pre[i]][i];

circle[i]= true;

}//首先把圈里的边权全部加起来，并且留出t节点，作为外部接口

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

if( !circle[i] && graph[i][t]!= INF )

graph[i][t]-= graph[pre[t]][t];

//上面这个for循环的作用是对t节点做更新操作，为什么要单独做？你可以看看线面这个循环的跳出条件。

for(j= pre[t]; j!= t; j= pre[j] )

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

{

if( circle[i] )

continue;

if( graph[i][j]!= INF )

graph[i][t]= min( graph[i][t], graph[i][j]- graph[pre[j]][j] );

/\*\*///////////////////////////////////////////////////////////////////////////

graph[t][i]= min( graph[j][i], graph[t][i] );

}

//这个循环对圈中的其他顶点进行更新

}

void solve()

{

int j;

memset( circle, false, sizeof(circle) );

while( ( j= exist\_circle() )!= -1 )

update( j );

for( j= 1; j<= n; ++j )

if( j!= root && !circle[j] )

ans+= graph[pre[j]][j];

printf("%.2f\n", ans );

}

int main()

{

int i;

while( scanf("%d%d",&n,&m)!= EOF )

{

for(i= 0; i<= n; ++i )

for( int j= 0; j<= n; ++j )

graph[i][j]= INF;

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

scanf("%lf%lf",&p[i].x, &p[i].y );

for(i= 0; i< m; ++i )

{

int a, b;

scanf("%d%d",&a,&b);

graph[a][b]= dist( a, b );

}

root= 1;

ans= 0;

if( !check() )

printf("poor snoopy\n");

else

solve();

}

return 0;

}

## 动态最小生成树

/\*

动态最小生成树

Q(logQ)^2

(qx[i], qy[i])表示将编号为qx[i]的边的权值改为qy[i]

删除一条边相当于将其权值改为\infinity

加入一条边相当于将其权值从\infinity变成某个值

\*/

#include<cstdio>

#include<algorithm>

using namespace std;

const int maxn = 100000 + 5;

const int maxm = 1000000 + 5;

const int maxq = 1000000 + 5;

const int qsize = maxm + 3\*maxq;

int x[qsize],y[qsize],z[qsize];

int qx[maxq],qy[maxq];

int n,m,Q;

void init()

{

scanf("%d%d",&n,&m);

for(int i=0;i<m;i++)

scanf("%d%d%d",x+i,y+i,z+i);

scanf("%d",&Q);

for(int i=0;i<Q;i++)

{

scanf("%d%d",qx+i,qy+i);

qx[i]--;

}

}

int a[maxn];

int \*tz;

int find( int x )

{

int root = x;

while( a[root] ) root = a[root];

int next;

while( next = a[x] )

{

a[x] = root;

x = next;

}

return root;

}

inline bool cmp( const int &a,const int &b )

{

return tz[a] < tz[b];

}

int kx[maxn],ky[maxn],kt;

int vd[maxn],id[maxm];

int app[maxm];

bool extra[maxm];

long long printState( int \*qx,int \*qy,int Q,int n,int \*x,int \*y,int \*z,int m,long long ans )

{

printf("%d %d\n",n,m);

for(int i=0;i<m;i++) printf("%d %d %d\n",x[i],y[i],z[i]);

printf("Q = %d\n",Q);

for(int i=0;i<Q;i++) printf("%d %d\n",qx[i],qy[i]);

return ans;

}

void solve( int \*qx,int \*qy,int Q,int n,int \*x,int \*y,int \*z,int m,long long ans )

{

if(Q==1)

{

for(int i=1;i<=n;i++) a[i] = 0;

z[ qx[0] ] = qy[0];

for(int i=0;i<m;i++) id[i] = i;tz = z;

sort(id,id+m,cmp);

int ri,rj;

for(int i=0;i<m;i++)

{

ri = find( x[id[i]] );

rj = find( y[id[i]] );

if(ri!=rj)

{

ans+=z[id[i]];

a[ri] = rj;

}

}

printf("%I64d\n",ans);

return;

}

int ri,rj;

//contract

kt = 0;

for(int i=1;i<=n;i++) a[i] = 0;

for(int i=0;i<Q;i++)

{

ri = find( x[qx[i]] );

rj = find( y[qx[i]] );

if(ri!=rj) a[ri] = rj;

}

int tm = 0;

for(int i=0;i<m;i++) extra[i] = true;

for(int i=0;i<Q;i++) extra[ qx[i] ] = false;

for(int i=0;i<m;i++) if(extra[i])

id[tm++] = i;

tz = z;

sort( id,id+tm,cmp );

for(int i=0;i<tm;i++)

{

ri = find( x[id[i]] );

rj = find( y[id[i]] );

if(ri!=rj)

{

a[ri] = rj;

ans += z[id[i]];

kx[kt] = x[id[i]];

ky[kt] = y[id[i]];

kt++;

}

}

for(int i=1;i<=n;i++) a[i] = 0;

for(int i=0;i<kt;i++)

a[ find( kx[i] ) ] = find( ky[i] );

int n2 = 0;

for(int i=1;i<=n;i++) if(a[i]==0)

vd[i] = ++n2;

for(int i=1;i<=n;i++) if(a[i])

vd[i] = vd[find(i)];

int \*Nx = x + m;

int \*Ny = y + m;

int \*Nz = z + m;

int m2 = 0;

for(int i=0;i<m;i++) app[i] = -1;

for(int i=0;i<Q;i++) if( app[qx[i]]==-1 )

{

Nx[m2] = vd[ x[ qx[i] ] ];

Ny[m2] = vd[ y[ qx[i] ] ];

Nz[m2] = z[ qx[i] ];

app[qx[i]] = m2;

m2++;

}

for(int i=0;i<Q;i++)

{

z[ qx[i] ] = qy[i];

qx[i] = app[qx[i]];

}

for(int i=1;i<=n2;i++) a[i] = 0;

for(int i=0;i<tm;i++)

{

ri = find( vd[ x[id[i]] ] );

rj = find( vd[ y[id[i]] ] );

if(ri!=rj)

{

a[ri] = rj;

Nx[m2] = vd[ x[id[i]] ];

Ny[m2] = vd[ y[id[i]] ];

Nz[m2] = z[id[i]];

m2++;

}

}

int mid = Q/2;

solve( qx,qy,mid,n2,Nx,Ny,Nz,m2,ans );

solve( qx+mid,qy+mid,Q-mid,n2,Nx,Ny,Nz,m2,ans );

}

void work()

{

if(Q) solve( qx,qy,Q,n,x,y,z,m,0 );

}

int main()

{

freopen("input.txt","r",stdin);

init();

work();

return 0;

}

## Hopcroft

#include <cstdio>

#include <cstring>

using namespace std;

int from[1010], wh[1010];

int g[1010];

int num[100010], nxt[100010], tot;

int n, m;

int ans;

int h, t, q[1010], dx[1010], dy[1010];

bool bfs(){

bool ret=false;

h=0; t=0;

for ( int i=0; i<n; i++ )

if ( wh[i]==-1 ){

t++; q[t]=i;

}

memset( dx, 0, sizeof( dx ) );

memset( dy, 0, sizeof( dy ) );

while ( h<t ){

h++;

for ( int i=g[q[h]]; i!=0; i=nxt[i] )

if ( dy[num[i]]==0 ){

dy[num[i]]=dx[q[h]]+1;

if ( from[num[i]]==-1 )

ret=true;

else {

dx[from[num[i]]]=dx[q[h]]+2;

t++; q[t]=from[num[i]];

}

}

}

return ret;

}

bool dfs( int x ){

for ( int i=g[x]; i!=0; i=nxt[i] ){

if ( dy[num[i]]==dx[x]+1 ){

dy[num[i]]=0;

if ( from[num[i]]==-1 || dfs( from[num[i]] ) ){

wh[x]=num[i]; from[num[i]]=x; return true;

}

}

}

return false;

}

void hopcroft(){

memset( from, -1, sizeof( from ) );

memset( wh, -1, sizeof( wh ) );

while ( bfs() ){

for ( int i=0; i<n; i++ )

if ( wh[i]==-1 && dfs(i) ) ans++;

}

}

void insert( int x, int y ){

tot++; num[tot]=y; nxt[tot]=g[x]; g[x]=tot;

}

int main(){

while ( scanf("%d %d", &n, &m)==2 ){

tot=0;

memset( g, 0, sizeof( g ) );

for ( int i=0; i<n; i++ ){

int x;

scanf("%d", &x);

for ( int j=0; j<x; j++ ){

int y;

scanf("%d", &y);

y--;

insert( i, y );

}

}

ans=0;

hopcroft();

printf("%d\n", ans);

}

}

## 割点缩块

//PKU 2942 Knights of the Round Table

bool hostile[maxn][maxn];

vector<int> edge[maxn];

int order[maxn], low[maxn], in\_seq[maxn];

int stack[maxn], list[maxn];

int color[maxn];

bool ok[maxn];

int n, m, ans, cnt, top, pop, len;

for (i = 0; i < n; i++) {

edge[i].clear();

for (j = 0; j < n; j++)

if (i != j && !hostile[i][j])

edge[i].push\_back(j);

bool Draw(int v, int cc) {

color[v] = cc;

int i, succ;

for (i = edge[v].size() - 1; i >= 0; i--) {

succ = edge[v][i];

if (in\_seq[succ] != cnt) continue;

if (color[succ] == cc) return 1;

if (color[succ] == -1 && Draw(succ, 1 - cc)) return 1;

}

return 0;

}

void Check() {

int i;

for (i = 0; i < len; i++)

color[list[i]] = -1;

if (Draw(list[0], 0))

for (i = 0; i < len; i++)

ok[list[i]] = 1;

}

void DFS(int v) {// main

stack[++top] = v;

order[v] = low[v] = pop++;

int i, succ;

for (i = edge[v].size() - 1; i >= 0; i--) {

succ = edge[v][i];

if (order[succ] == -1) {

DFS(succ);

if (low[succ] >= order[v]) {

cnt++;

len = 0;

do {

in\_seq[stack[top]] = cnt;

list[len++] = stack[top];

top--;

} while (stack[top + 1] != succ);

in\_seq[v] = cnt;

list[len++] = v;

Check();

}

low[v] = min(low[v], low[succ]);

} else low[v] = min(low[v], order[succ]);

}

}

void Work() {

int i;

cnt = pop = ans = 0;

for (i = 0; i < n; i++) {

order[i] = in\_seq[i] = -1;

ok[i] = 0;

}

for (i = 0; i < n; i++)

if (order[i] == -1) {

top = -1;

DFS(i);

}

for (i = 0; i < n; i++)

ans += !ok[i];

printf("%d\n", ans);

}

## 割边缩块

void DFS(int par, int x)

    {

      Low[x] = Dfn[x] = ++ idx;

      stack[++top] = x;

      for (int tmp = a[x]; tmp; tmp = next[tmp])

        if (tp[tmp] != par)// 改成按边判断

          if (!Dfn[tp[tmp]])

            {

              DFS(x, tp[tmp]);

              Low[x] = min(Low[x], Low[tp[tmp]]);

            }

          else Low[x] = min(Low[x], Dfn[tp[tmp]]);

      if (Dfn[x] == Low[x])

        {

          ++Tot;

          while (1)

            {

              int cur = stack[top --];

              Mark[cur] = Tot;

              if (cur == x) break;

            }

        }

}

## K短路（可重复）

// Author: Amber

#define for\_each(it, v) for (vector<Edge\*>::iterator it = (v).begin(); it != (v).end(); ++it)

const int MAX\_N = 10000;

const int MAX\_M = 50000;

const int MAX\_K = 10000;

const int INF = 1000000000;

struct Edge

{

int from, to;

int weight;

};

struct HeapNode

{

Edge\* edge;

int depth;

HeapNode\* child[4];

//child[0..1] for heap G

//child[2..3] for heap out edge

};

int n, m, k, s, t;

Edge\* edge[MAX\_M];

int dist[MAX\_N];

Edge\* prev[MAX\_N];

vector<Edge\*> graph[MAX\_N];

vector<Edge\*> graphR[MAX\_N];

HeapNode\* nullNode;

HeapNode\* heapTop[MAX\_N];

HeapNode\* createHeap(HeapNode\* curNode, HeapNode\* newNode)

{

if (curNode == nullNode)

return newNode;

HeapNode\* rootNode = new HeapNode;

memcpy(rootNode, curNode, sizeof(HeapNode));

if (newNode->edge->weight < curNode->edge->weight)

{

rootNode->edge = newNode->edge;

rootNode->child[2] = newNode->child[2];

rootNode->child[3] = newNode->child[3];

newNode->edge = curNode->edge;

newNode->child[2] = curNode->child[2];

newNode->child[3] = curNode->child[3];

}

if (rootNode->child[0]->depth < rootNode->child[1]->depth)

rootNode->child[0] = createHeap(rootNode->child[0], newNode);

else

rootNode->child[1] = createHeap(rootNode->child[1], newNode);

rootNode->depth = max(rootNode->child[0]->depth, rootNode->child[1]->depth) + 1;

return rootNode;

}

bool heapNodeMoreThan(HeapNode\* node1, HeapNode\* node2)

{

return node1->edge->weight > node2->edge->weight;

}

int main()

{

scanf("%d%d%d", &n, &m, &k);

scanf("%d%d", &s, &t);

s--, t--;

while (m--)

{

Edge\* newEdge = new Edge;

int i, j, w;

scanf("%d%d%d", &i, &j, &w);

i--, j--;

newEdge->from = i;

newEdge->to = j;

newEdge->weight = w;

graph[i].push\_back(newEdge);

graphR[j].push\_back(newEdge);

}

//Dijkstra

queue<int> dfsOrder;

memset(dist, -1, sizeof(dist));

typedef pair<int, pair<int, Edge\*> > DijkstraQueueItem;

priority\_queue<DijkstraQueueItem, vector<DijkstraQueueItem>, greater<DijkstraQueueItem> > dq;

dq.push(make\_pair(0, make\_pair(t, (Edge\*) NULL)));

while (!dq.empty())

{

int d = dq.top().first;

int i = dq.top().second.first;

Edge\* edge = dq.top().second.second;

dq.pop();

if (dist[i] != -1)

continue;

dist[i] = d;

prev[i] = edge;

dfsOrder.push(i);

for\_each(it, graphR[i])

dq.push(make\_pair(d + (\*it)->weight, make\_pair((\*it)->from, \*it)));

}

//Create edge heap

nullNode = new HeapNode;

nullNode->depth = 0;

nullNode->edge = new Edge;

nullNode->edge->weight = INF;

fill(nullNode->child, nullNode->child + 4, nullNode);

while (!dfsOrder.empty())

{

int i = dfsOrder.front();

dfsOrder.pop();

if (prev[i] == NULL)

heapTop[i] = nullNode;

else

heapTop[i] = heapTop[prev[i]->to];

vector<HeapNode\*> heapNodeList;

for\_each(it, graph[i])

{

int j = (\*it)->to;

if (dist[j] == -1)

continue;

(\*it)->weight += dist[j] - dist[i];

if (prev[i] != \*it)

{

HeapNode\* curNode = new HeapNode;

fill(curNode->child, curNode->child + 4, nullNode);

curNode->depth = 1;

curNode->edge = \*it;

heapNodeList.push\_back(curNode);

}

}

if (!heapNodeList.empty()) //Create heap out

{

make\_heap(heapNodeList.begin(), heapNodeList.end(), heapNodeMoreThan);

int size = heapNodeList.size();

for (int p = 0; p < size; p++)

{

heapNodeList[p]->child[2] = 2 \* p + 1 < size ? heapNodeList[2 \* p + 1] : nullNode;

heapNodeList[p]->child[3] = 2 \* p + 2 < size ? heapNodeList[2 \* p + 2] : nullNode;

}

heapTop[i] = createHeap(heapTop[i], heapNodeList.front());

}

}

//Walk on DAG

typedef pair<long long, HeapNode\*> DAGQueueItem;

priority\_queue<DAGQueueItem, vector<DAGQueueItem>, greater<DAGQueueItem> > aq;

if (dist[s] == -1)

printf("NO\n");

else

{

printf("%d\n", dist[s]);

if (heapTop[s] != nullNode)

aq.push(make\_pair(dist[s] + heapTop[s]->edge->weight, heapTop[s]));

}

k--;

while (k--)

{

if (aq.empty())

{

printf("NO\n");

continue;

}

long long d = aq.top().first;

HeapNode\* curNode = aq.top().second;

aq.pop();

printf("%I64d\n", d);

if (heapTop[curNode->edge->to] != nullNode)

aq.push(make\_pair(d + heapTop[curNode->edge->to]->edge->weight, heapTop[curNode->edge->to]));

for (int i = 0; i < 4; i++)

if (curNode->child[i] != nullNode)

aq.push(make\_pair(d - curNode->edge->weight + curNode->child[i]->edge->weight, curNode->child[i]));

}

return 0;

}

## K短路（不可重复）

#include <cstdio>

#include <cstring>

#include <vector>

#include <queue>

using namespace std;

int Num[10005][205];

int Path[10005][205];

int dev[10005];

int from[10005];

int value[10005];

int dist[205];

int Next[205];

int Graph[205][205];

bool forbid[205];

bool hasNext[10005][205];

int N, M, K, s, t;

int tot, cnt;

struct cmp {

bool operator() (const int &a, const int &b) {

int \*i, \*j;

if(value[a] != value[b])

return value[a] > value[b];

for(i = Path[a], j = Path[b]; (\*i) == (\*j); i ++, j ++);

return (\*i) > (\*j);

}

};

void Check(int idx, int st, int \*path, int &res) {

int i, j;

for(i = 0; i < N; i ++) {

dist[i] = 1000000000;

Next[i] = t;

}

dist[t] = 0;

forbid[t] = true;

j = t;

while(1) {

for(i = 0; i < N; i ++)

if(!forbid[i] && (i != st || !hasNext[idx][j]) && (dist[j] + Graph[i][j] < dist[i] || dist[j] + Graph[i][j] == dist[i] && j < Next[i])) {

Next[i] = j;

dist[i] = dist[j] + Graph[i][j];

}

j = -1;

for(i = 0; i < N; i ++)

if(!forbid[i] && (j == -1 || dist[i] < dist[j]))

j = i;

if(j == -1)

break;

forbid[j] = 1;

if(j == st)

break;

}

res += dist[st];

for(i = st; i != t; i = Next[i], path ++)

(\*path) = i;

(\*path) = i;

}

int main() {

int i, j, k, l;

while(scanf("%d%d%d%d%d", &N, &M, &K, &s, &t) && N) {

priority\_queue <int, vector <int>, cmp> Q;

for(i = 0; i < N; i ++)

for(j = 0; j < N; j ++)

Graph[i][j] = 1000000000;

for(i = 0; i < M; i ++) {

scanf("%d%d%d", &j, &k, &l);

Graph[j - 1][k - 1] = l;

}

s --;

t --;

memset(forbid, false, sizeof(forbid));

memset(hasNext[0], false, sizeof(hasNext[0]));

Check(0, s, Path[0], value[0]);

dev[0] = 0;

from[0] = 0;

Num[0][0] = 0;

Q.push(0);

cnt = 1;

tot = 1;

for(i = 0; i < K; i ++) {

if(Q.empty())

break;

l = Q.top();

Q.pop();

for(j = 0; j <= dev[l]; j ++)

Num[l][j] = Num[from[l]][j];

for(; Path[l][j] != t; j ++) {

memset(hasNext[tot], false, sizeof(hasNext[tot]));

Num[l][j] = tot ++;

}

for(j = 0; Path[l][j] != t; j ++)

hasNext[ Num[l][j] ][ Path[l][j + 1] ] = true;

for(j = dev[l]; Path[l][j] != t; j ++) {

memset(forbid, false, sizeof(forbid));

value[cnt] = 0;

for(k = 0; k < j; k ++) {

forbid[Path[l][k]] = true;

Path[cnt][k] = Path[l][k];

value[cnt] += Graph[ Path[l][k] ][ Path[l][k + 1] ];

}

Check(Num[l][j], Path[l][j], &Path[cnt][j], value[cnt]);

if(value[cnt] > 2000000)

continue;

dev[cnt] = j;

from[cnt] = l;

Q.push(cnt);

cnt ++;

}

}

if(i < K || value[l] > 2000000)

printf("None\n");

else {

for(i = 0; Path[l][i] != t; i ++)

printf("%d-", Path[l][i] + 1);

printf("%d\n", t + 1);

}

}

return 0;

}

# 数学

## Miller-Rabin

int strong\_pseudo\_primetest(long long n,int base) {

long long n2=n-1,res;

int s=0;

while(n2%2==0) n2>>=1,s++;

res=powmod(base,n2,n);

if((res==1)||(res==n-1)) return 1;

s--;

while(s>=0) {

res=mulmod(res,res,n);

if(res==n-1) return 1;

s--;

}

return 0; // n is not a strong pseudo prime

}

int isprime(long long n) {

if(n<2) return 0;

if(n<4) return 1;

if(strong\_pseudo\_primetest(n,2)==0) return 0;

if(strong\_pseudo\_primetest(n,3)==0) return 0;

if(n<1373653LL) return 1;

if(strong\_pseudo\_primetest(n,5)==0) return 0;

if(n<25326001LL) return 1;

if(strong\_pseudo\_primetest(n,7)==0) return 0;

if(n==3215031751LL) return 0;

if(n<25000000000LL) return 1;

if(strong\_pseudo\_primetest(n,11)==0) return 0;

if(n<2152302898747LL) return 1;

if(strong\_pseudo\_primetest(n,13)==0) return 0;

if(n<3474749660383LL) return 1;

if(strong\_pseudo\_primetest(n,17)==0) return 0;

if(n<341550071728321LL) return 1;

if(strong\_pseudo\_primetest(n,19)==0) return 0;

if(strong\_pseudo\_primetest(n,23)==0) return 0;

if(strong\_pseudo\_primetest(n,29)==0) return 0;

if(strong\_pseudo\_primetest(n,31)==0) return 0;

if(strong\_pseudo\_primetest(n,37)==0) return 0;

return 1;

}

## 启发式分解

#include <cstdio>

#include <cstdlib>

#include <cmath>

#include <ctime>

#include <algorithm>

#include <iostream>

using namespace std;

typedef long long I64;

int ansn;

I64 ans[1000];

I64 mod\_mul(I64 x,I64 y,I64 n){

long long d = (long long)((long double)x\*y/n);

d=x\*y-n\*d;

while (d < 0) d += n;

while (d >= n) d -= n;

return d;

}

I64 myrand(){

I64 a=rand();

a\*=rand();

return abs(a);

}

I64 mod\_exp(I64 a,I64 x,I64 n){

I64 ret=1;

while(x){

if(x&1) ret=mod\_mul(ret,a,n);

a=mod\_mul(a,a,n);

x>>=1;

}

return ret;

}

I64 gcd(I64 x,I64 y){

I64 q;

while(1){

if(!y) return abs(x);

q=x,x=y,y=q%y;

}

}

bool Rabin\_Miller(I64 n){

I64 k=0,i,j,m,a;

if(n<2)

return 0;

if(n==2)

return 1;

if(!(n&1))

return 0;

m=n-1;

while(!(m&1))

m>>=1,k++;

for(i=0;i<20;i++){

a=myrand()%(n-2)+2;

a=mod\_exp(a,m,n);

if(a==1)

continue;

for(j=0;j<k;j++){

if(a==n-1)

break;

a=mod\_mul(a,a,n);

}

if(j<k)

continue;

return 0;

}

return 1;

}

I64 func(I64 x,I64 n){

return (mod\_mul(x,x,n)+1)%n;

}

I64 Pollard(I64 n){

I64 i,x,y,p;

if(Rabin\_Miller(n))

return n;

if(!(n&1))

return 2;

for(i=1;i<20;i++){

x=i;

y=func(x,n);

p=gcd(y-x,n);

while(p==1){

x=func(x,n);

y=func(func(y,n),n);

p=gcd((y-x+n)%n,n)%n;

}

if(p==0||p==n)

continue;

return p;

}

}

void factor(I64 n){

I64 x;

x=Pollard(n);

if(x==n){

ans[ansn++]=x;

return;

}

factor(x);

factor(n/x);

}

void output(){

int i,j;

I64 tmp;

for(i=0;i<ansn;i++)

for(j=i+1;j<ansn;j++)

if(ans[i]>ans[j]){

tmp=ans[i];

ans[i]=ans[j];

ans[j]=tmp;

}

for (i = 0; i < ansn; i += j) {

for (j = 0; i + j < ansn && ans[i] == ans[i + j]; j++);

if (i) cout << " \*";

cout << " " << ans[i];

if (j > 1) cout << "^" << j;

}

cout << endl;

}

int main(){

I64 n;

srand((unsigned)time(NULL));

int tt;

scanf("%d", &tt);

while(tt--){

cin >> n;

if(n==1){

cout<<"1 = 1"<<endl;

continue;

}

if(n<0)

break;

ansn=0;

factor(n);

cout << n << " =";

output();

}

return 0;

}

## N次剩余

//BEGIN TEMPLATE HERE

#define SIZE(X) ((int)(X.size()))

namespace Solution {

typedef long long ll;

ll powMod(ll a, ll n, ll m) {

ll res = 1, ONE = a;

for (; n; n /= 2) {

if (n&1) res = res \* ONE % m;

ONE = ONE \* ONE % m;

}

return res;

}

int findRoot(int p) {

if (p == 2) return 3;

vector<int> D;

int Phi = p - 1;

int t = Phi;

for (int i = 2; (ll)i \* i <= t; ++i) {

if (t % i == 0) {

D.push\_back(i);

for (; t % i == 0; t /= i);

}

}

if (t > 1) D.push\_back(t);

for (int g = 1; ; ++g) {

bool good = true;

for (int i = 0; i < SIZE(D); ++i) {

if (powMod(g, Phi / D[i], p) == 1) {

good = false;

break;

}

}

if (good) {

return g;

}

}

}

// return y such that x^y mod m = n

ll logMod(int x, int n, int m) {

map<ll, int> rec;

int s = (int)(sqrt((double)m));

for (; (ll)s \* s <= m; ) s++;

ll cur = 1;

for (int i = 0; i < s; ++i) {

rec[cur] = i;

cur = cur \* x % m;

}

ll mul = cur;

cur = 1;

for (int i = 0; i < s; ++i) {

ll more = (ll)n \* powMod(cur, m - 2, m) % m;//more = n / cur: mul inverse of cur

if (rec.count(more)) {

return i \* s + rec[more];

}

cur = cur \* mul % m;

}

return -1;

}

ll extGcd(ll a, ll b, ll &x, ll &y) {

if (b == 0) {

x = 1;

y = 0;

return a;

}

ll ret = extGcd(b, a % b, x, y);

ll t = x;

x = y;

y = t - (a / b) \* y;

return ret;

}

// solve x^n mod p = N

vector<int> solve(int p, int N, int a) {

int g = findRoot(p);

ll m = logMod(g, a, p);

vector<int> ret;

if (a == 0) {

ret.push\_back(0);

return ret;

}

if (m == -1) {

return ret;

}

ll A = N, B = p - 1, C = m, x, y;

ll d = extGcd(A, B, x, y);

if (C % d != 0) return ret;

x = x \* (C / d) % B;//g^B mod p = g^(p-1) mod p = 1

ll delta = B / d;

for (int i = 0; i < d; ++i) {

x = ((x + delta) % B + B) % B;

ret.push\_back((int)powMod(g, x, p));

}

sort(ret.begin(), ret.end());

ret.erase(unique(ret.begin(), ret.end()), ret.end());

return ret;

}

};

//END TEMPLATE HERE

## 2次剩余

#include <cstdio>

#include <cstdlib>

#include <algorithm>

using namespace std;

int power(int a, int b, const int MODE) {

if (b == 0) return 1;

int t = power(a, b / 2, MODE);

t = (t \* t) % MODE;

if (b & 1) t = (t \* a) % MODE;

return t;

}

void calcH(int &t, int &h, const int p) {

int tmp = p - 1;

for (t = 0; (tmp & 1) == 0; tmp /= 2) t++;

h = tmp;

}

// solve equation x^2 mod p = a

bool solve(int a, int p, int &x, int &y) {

srand(19920225);

if (p == 2) {

x = y = 1;

return true;

}

int p2 = p / 2;

int tmp = power(a, p2, p);

if (tmp == p - 1) return false;

if ((p + 1) % 4 == 0) {

x = power(a, (p + 1) / 4, p);

y = p - x;

return true;

} else {

int t, h, b, pb;

calcH(t, h, p);

if (t >= 2) {

do {

b = rand() % (p - 2) + 2;

}

while (power(b, p / 2, p) != p - 1);

pb = power(b, h, p);

}

int s = power(a, h / 2, p);

for (int step = 2; step <= t; step++) {

int ss = (((s \* s) % p) \* a) % p;

for (int i = 0; i < t - step; i++) ss = (ss \* ss) % p;

if (ss + 1 == p) s = (s \* pb) % p;

pb = (pb \* pb) % p;

}

x = (s \* a) % p;

y = p - x;

}

return true;

}

## 线性筛法

// There are some details to be changed.

// a \* b <= n <==> a <= n / b

for (i=2;i<=n;i++)

{

if (a[i]==0)

{

num++;p[num]=i;

}

for (j=1;((j<=num) && (i\*p[j]<=n)); j++)

{

a[i\*p[j]] = 1;

if (i%p[j] == 0) break;

}

}

## Pell方程

#define sqr(x) ((x)\*(x))

#define maxn 50

#define UL unsigned long long

UL A,B,p[maxn],q[maxn],a[maxn],g[maxn],h[maxn];

int main()

{

int n;

for (int test=1;scanf("%d",&n) && n;++test)

{

printf("Case %d: ",test);

if (fabs(sqrt(n)-floor(sqrt(n)+1e-7))<=1e-7)

{

int a=(int)(floor(sqrt(n)+1e-7));

printf("%d %d\n",a,1);

}else

{

//求x^2-ny^2=1的最小正整数根,n不是完全平方数

p[1]=1;p[0]=0;

q[1]=0;q[0]=1;

a[2]=(int)(floor(sqrt(n)+1e-7));

g[1]=0;h[1]=1;

for (int i=2;i;++i)

{

g[i]=-g[i-1]+a[i]\*h[i-1];

h[i]=(n-sqr(g[i]))/h[i-1];

a[i+1]=(g[i]+a[2])/h[i];

p[i]=a[i]\*p[i-1]+p[i-2];

q[i]=a[i]\*q[i-1]+q[i-2];

if (sqr((UL)(p[i]))-n\*sqr((UL)(q[i]))==1)

{

A=p[i];B=q[i];

break;

}

}

cout << A << ' ' << B <<endl;

}

}

return 0;

}

## 皮克公式

一个多边形的顶点如果全是格点，这多边形就叫做格点多边形。有趣的是，这种格点多边形的面积计算起来很方便，只要数一下图形边线上的点的数目及图内的点的数目，就可用公式算出。

　　这个公式是皮克(Pick)在1899年给出的，被称为“皮克定理”，这是一个实用而有趣的定理。

　　给定顶点坐标均是整点（或正方形格点）的简单多边形，皮克定理说明了其面积S和内部格点数目a、边上格点数目b的关系：

　　S=a+ b/2 - 1。

　　(其中a表示多边形内部的点数,b表示多边形边界上的点数,S表示多边形的面积)

## 蔡勒公式

int zeller(int y,int m,int d)

{

if (m<=2)

y--,m+=12;

int c=y/100;

y%=100;

int w=((c>>2)-(c<<1)+y+(y>>2)+(13\*(m+1)/5)+d-1)%7;

if (w<0)

w+=7;

return(w);

}

## 莫比乌斯函数以及gcd=1的对数

#define maxn 10000000

int div[maxn+5],sum[maxn+5],p[1000000],len;

long long ans;

inline void prepare()

{

memset(div,0,sizeof(div));

for (int i=2;i<=maxn;++i)

if (!div[i])

{

div[i]=i;

p[len++]=i;

if (i>maxn/i) continue;

for (int j=i\*i;j<=maxn;j+=i)

if (!div[j]) div[j]=i;

}

for (int i=1;i<=maxn;++i)

{

int cnt=0,last=0;

for (int j=i;j>1;last=div[j],j/=div[j])

{

if (div[j]==last)

{

sum[i]=0;

goto Break;

}

cnt^=1;

}

if (cnt) sum[i]=-1;

else sum[i]=1;

Break:;

sum[i]+=sum[i-1];

}

}

//计算莫比乌斯函数，及其前缀和

//复杂度O(nlogn)

inline void calc(int a,int b)

{

for (int i=1,j,p,q;i<=a;i=j+1)

{

p=a/i;

q=b/i;

j=b/q;

if (a<p\*j) j=a/p;

ans+=(long long)(sum[j]-sum[i-1])\*p\*q;

}

}

//求1..a和1..b中有多少对的gcd=1

//复杂度O(sqrt(a+b))

## 牛顿迭代

　x1=x0-func(x0)/func1(x0);//进行牛顿迭代计算

我们要求f(x)=0的解。func(x)为原方程，func1为原方程的导数方程

## FFT

FFt\_speed

typedef long long int64;

#define two(X) (1<<(X))

const double pi=acos(-1.0);

template<class T> inline T lowbit(T n){return (n^(n-1))&n;}

class complex

{

public:

double a,b;

complex(){};

complex(double \_a,double \_b) {a=\_a;b=\_b;}

};

const int maxn=two(19)+5;

int L1,L2;

int s1[maxn],s2[maxn];

int n,id;

int A[maxn];

complex tmp[maxn],P[maxn],PB[maxn];

int lowbit(int n)

{

return (n^(n-1))&n;

}

int getnumber(int s[],int L,int id)

{

if (id>L)

return 0;

return s[L-id]-48;

}

void Fill(int s[],int L,int m,int d)

{

if (m==n)

P[d]=complex(s[id++],0);

else

{

Fill(s,L,m\*2,d);

Fill(s,L,m\*2,d+m);

}

}

void Fill2(int m,int d)

{

if (m==n)

P[d]=tmp[id++];

else

{

Fill2(m\*2,d);

Fill2(m\*2,d+m);

}

}

void FFT(int oper)

{

for (int d=0;(1<<d)<n;d++)

{

int i,m=(1<<d);

double p0=2\*pi/double(m\*2)\*double(oper);

double sinp0=sin(p0);

double cosp0=cos(p0);

for (i=0;i<n;i+=(m\*2))

{

double sinp=0;

double cosp=1;

for (int j=0;j<m;j++)

{

double ta=cosp\*P[i+j+m].a-sinp\*P[i+j+m].b;

double tb=cosp\*P[i+j+m].b+sinp\*P[i+j+m].a;

P[i+j+m].a=P[i+j].a-ta;

P[i+j+m].b=P[i+j].b-tb;

P[i+j].a+=ta;

P[i+j].b+=tb;

double tsinp=sinp;

sinp=sinp\*cosp0+ cosp\*sinp0;

cosp=cosp\*cosp0-tsinp\*sinp0;

}

}

}

}

class CircularShifts

{

public:

int Z[maxn];

int maxScore(int L, int Z0, int A, int B, int M)

{

Z[0]=Z0%M;

for (int i=1;i<L+L;i++)

Z[i]=(int)(((int64)Z[i-1]\*(int64)A+(int64)B)%M);

memset(s1,0,sizeof(s1));

memset(s2,0,sizeof(s2));

for (int i=0;i<L;i++)

{

s1[i+L]=s1[i]=Z[i]%100;

s2[L-1-i]=Z[i+L]%100;

}

n=L+L;

for (;n!=lowbit(n);n+=lowbit(n)); //不同长度按L1,L2补全

id=0;

Fill(s1,L,1,0);

FFT(1);

for (int i=0;i<n;i++)

PB[i]=P[i];

id=0;

Fill(s2,L,1,0);

FFT(1);

for (int i=0;i<n;i++)

{

tmp[i].a=P[i].a\*PB[i].a-P[i].b\*PB[i].b;

tmp[i].b=P[i].a\*PB[i].b+P[i].b\*PB[i].a;

}

id=0;

Fill2(1,0);

FFT(-1);

double result=-1e100;

for (int i=L-1;i<L+L-1;i++)

{

double t=P[i].a/(double)(n);

if (t>result)

result=t;

}

return (int)(result+0.5);

}

};

int main()

{

//这个程序中没有出现小写的L。

//这个程序是求s1[]\*s2[]平移后的矩阵的。倍长了各自的长度后，只需要截取中间的一段即可。

}

## FFT(integer)

using namespace std;

#define Inv(n) PowMod(n, P - 2, P)

/\*

\* P = C \* 2^k + 1 ， P是素数

\* G 为原根

\* 对于 N = 2^w 的 FFT, 在 Zp 中 用 g = G^((P - 1) / N) (mod P) 来代替复根 e^[ -j(2PI / N)]

\*/

const int maxn = 1 << 19;

char A[maxn], B[maxn];

int a[maxn], b[maxn], n;

int P;

int \_g[25];

int BIT\_CNT;

int ans[maxn];

inline int PowMod(long long a, int b, int c)

{

long long Res = 1;

for (; b; b >>= 1)

{

if(b & 1)

Res = Res\*a % c;

a = a\*a % c;

}

return Res;

}

bool IsPrime(int n)

{

int i;

for(i = 2; i\*i <= n; ++i)

if (n % i == 0)

return 0;

return 1;

}

int GetP(int Limit) // P = C \* 2^21 + 1, P >= Lim

{

int c = 3;

for(int t; ; ++c)

{

t = c << 21 | 1;

if (IsPrime(t) && t >= Limit)

return t;

}

return -1;

}

bool Isg(int a, int P)

{

int i, p0 = P-1;

for(i = 1; i \* i <= p0; ++i)

if( p0 % i == 0)

if (PowMod(a, i, P) == 1 && i < p0 || PowMod(a, p0/i, P) == 1 && p0/i < p0)

return 0;

return 1;

}

int getG(int P)

{

int g;

for(g = 2; ! Isg(g, P); ++g);

return g;

}

void Getg(int G, int P, int bLimit,int \_g[])

{

for(int i = 0; i < bLimit; ++i)

\_g[i] = PowMod(G, (P-1)/(1 << i), P);

}

int Reverse(int j)

{

int k = 0;

for(int i = 0; i < BIT\_CNT; ++i)

if((j >> i) & 1)

k |= 1 << (BIT\_CNT-i-1);

return k;

}

void FFT(int x[], int n)

{

int t0, t1, i0, j0, tt;

for(int i, j, m = 1; m <= BIT\_CNT; ++m)

{

j0 = (i0 = 1 << m) >> 1;

for(i = 0; i < n; i += i0)

for(j = 0, tt = 1; j < j0; ++j, tt = (long long)tt\*\_g[m] % P)

{

t0 = tt;

t1 = (long long)x[i+j+j0]\*t0 % P;

t0 = (x[i+j]+t1) % P;

t1 = (x[i+j]-t1) % P;

if(t1 < 0)

t1 += P;

x[i+j] = t0;

x[i+j+j0] = t1;

}

}

}

void Conv(int a[], int b[], int n) {

int i;

FFT(a, n);

FFT(b, n);

for(i = 0; i < n; ++ i)

b[i] = (long long)a[i]\*b[i] % P;

for(i = 0; i < n; ++ i)

a[Reverse(i)] = b[i == 0 ? 0 : n-i];

FFT(a, n);

for(i = 0; i < n; ++i)

a[i] = (long long)a[i] \* Inv( n ) % P ;

}

void Init()

{

P = GetP(1000000000);

Getg(getG(P), P, 21, \_g) ;

}

void Get()

{

int i,j;

scanf("%d", &n);

scanf("%s%s", A, B);

int v, c = 0, k = 0;

int av, bv, t = 1;

av = bv = 0;

int on = n/1+(n % 1 != 0);

for(BIT\_CNT = 1; on+on > (1 << BIT\_CNT); ++BIT\_CNT);

for(i = n-1; i >= 0; --i)

{

av = av+t\*(A[i]-'0');

bv = bv+t\*(B[i]-'0');

++c;

if(c == 1 || i == 0)

{

j = Reverse(k);

a[j] = av;

b[j] = bv;

++k;

c = av = bv = 0;

t = 1;

}

else

t \*= 10;

}

n = 1 << BIT\_CNT;

}

void Work(){

int i, j = 0, k;

Conv(a, b, n);

for(i = 0; i < n; ++i)

{

k = a[i]+j;

ans[i] = k % 10;

j = k/10;

}

for(i = n-1; i >= 0 && ans[i] == 0; --i);

for(printf("%d", ans[i--]); i >= 0; --i)

printf("%d", ans[i]);

puts("");

}

int main()

{

Init();

Get();

Work();

return 0;

}

## Romberg&Simpson

#include<vector>

#include<cmath>

template<class T>

double romberg(const T&f,double a,double b,double eps=1e-8){

std::vector<double>t;

double h=b-a,last,curr;

int k=1,i=1;

t.push\_back(h\*(f(a)+f(b))/2); // 梯形

do{

last=t.back();

curr=0;

double x=a+h/2;

for(int j=0;j<k;++j){

curr+=f(x);

x+=h;

}

curr=(t[0]+h\*curr)/2;

double k1=4.0/3.0,k2=1.0/3.0;

for(int j=0;j<i;j++){

double temp=k1\*curr-k2\*t[j];

t[j]=curr;

curr=temp;

k2/=4\*k1-k2; // 防止溢出

k1=k2+1;

}

t.push\_back(curr);

k\*=2;

h/=2;

i++;

}while(std::fabs(last-curr)>eps);

return t.back();

}

template<class T>

double simpson(const T&f,double a,double b,int n){

const double h=(b-a)/n;

double ans=f(a)+f(b);

for(int i=1;i<n;i+=2)ans+=4\*f(a+i\*h);

for(int i=2;i<n;i+=2)ans+=2\*f(a+i\*h);

return ans\*h/3;

}

#include<cstdio>

double test(double x){

if(x==0)return 1;

else return sin(x)/x;

}

int main(){

printf("%f\n",romberg(test,0,1));

printf("%f\n",simpson(test,0,1,(int)1e6));

}

## 多项式求根（求导二分）

const double error=1e-12;

const double infi=1e+12;

double a[10],x[10];

int n;

int sign(double x) {

return (x<-error)?(-1):(x>error);

}

double f(double a[],int n,double x) {

double tmp=1,sum=0;

for (int i=0;i<=n;i++) {

sum=sum+a[i]\*tmp;

tmp=tmp\*x;

}

return sum;

}

double binary(double l,double r,double a[],int n) {

int sl=sign(f(a,n,l)),sr=sign(f(a,n,r));

if (sl==0) return l;

if (sr==0) return r;

if (sl\*sr>0) return infi;

while (r-l>error) {

double mid=(l+r)/2;

int ss=sign(f(a,n,mid));

if (ss==0) return mid;

if (ss\*sl>0) l=mid; else r=mid;

}

return l;

}

void solve(int n,double a[],double x[],int &nx) {

if (n==1) {

x[1]=-a[0]/a[1];

nx=1;

return;

}

double da[10],dx[10];

int ndx;

for (int i=n;i>=1;i--) da[i-1]=a[i]\*i;

solve(n-1,da,dx,ndx);

nx=0;

if (ndx==0) {

double tmp=binary(-infi,infi,a,n);

if (tmp<infi) x[++nx]=tmp;

return;

}

double tmp;

tmp=binary(-infi,dx[1],a,n);

if (tmp<infi) x[++nx]=tmp;

for (int i=1;i<=ndx-1;i++) {

tmp=binary(dx[i],dx[i+1],a,n);

if (tmp<infi) x[++nx]=tmp;

}

tmp=binary(dx[ndx],infi,a,n);

if (tmp<infi) x[++nx]=tmp;

}

int main() {

scanf("%d",&n);

for (int i=n;i>=0;i--) scanf("%lf",&a[i]);

int nx;

solve(n,a,x,nx);

for (int i=1;i<=nx;i++) printf("%0.6f\n",x[i]);

return 0;

}

## 线性规划

/\*

说明：

本来变量都应放在class里面的，但是由于在里面开大内存会RE，所以暂时先放外面。

N[0]代表N中的元素个数，B[0]代表B中的元素个数。

读入格式（在文件名为inputName的文件中读入)：

首先两个数n, m，表示未知数的数量和约束的数量。

接下来一行n个数，为目标函数的系数。

然后m行，每行m+1个数，表示一个约束。前m个数是系数，最后一个是常数项。

输出格式（在文件名为outputName的文件中输出）：

如果无解，只有一行"Infeasible"。

如果解可以无穷大，只有一行"Unbounded"。

否则，第一行为最大的目标函数值，接下来是每个未知数的值。

\*/

const double eps = 1e-10;

const int MAXSIZE = 2000;

const int oo = 19890709;

double A[MAXSIZE+1][MAXSIZE+1], tA[MAXSIZE+1][MAXSIZE+1];

double b[MAXSIZE+1], tb[MAXSIZE+1], c[MAXSIZE+1], tc[MAXSIZE+1];

int N[MAXSIZE+1+1], B[MAXSIZE+1+1];

int n, m;

double v;

class LinearProgramming

{

void read()

{

scanf("%d%d", &n, &m);

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

scanf("%lf", &c[i]);

for(int i=1; i<=m; i++)

{

for(int j=1; j<=n; j++)

scanf("%lf", &A[n+i][j]);

scanf("%lf", &b[n+i]);

}

}

void pivot(int l, int e)

{

tb[e] = b[l]/A[l][e];

tA[e][l] = 1/A[l][e];

for(int i=1; i<=N[0]; i++)

if (N[i] != e)

tA[e][N[i]] = A[l][N[i]]/A[l][e];

for(int i=1; i<=B[0]; i++)

{

tb[B[i]] = b[B[i]]-A[B[i]][e]\*tb[e];

tA[B[i]][l] = -A[B[i]][e]\*tA[e][l];

for(int j=1; j<=N[0]; j++)

if (N[j] != e)

tA[B[i]][N[j]] = A[B[i]][N[j]]-tA[e][N[j]]\*A[B[i]][e];

}

v += tb[e]\*c[e];

tc[l] = -tA[e][l]\*c[e];

for(int i=1; i<=N[0]; i++)

if (N[i] != e)

tc[N[i]] = c[N[i]]-tA[e][N[i]]\*c[e];

for(int i=1; i<=N[0]; i++)

if (N[i] == e) N[i] = l;

for(int i=1; i<=B[0]; i++)

if (B[i] == l) B[i] = e;

for(int i=1; i<=B[0]; i++)

{

for(int j=1; j<=N[0]; j++)

A[B[i]][N[j]] = tA[B[i]][N[j]];

b[B[i]] = tb[B[i]];

}

for(int i=1; i<=N[0]; i++)

c[N[i]] = tc[N[i]];

}

bool opt()//false stands for unbounded

{

while (true)

{

int l, e;

double maxUp = -1;//不能是0！

for(int ie=1; ie<=N[0]; ie++)

{

int te = N[ie];

if (c[te] <= eps) continue;//eps or 0????????????

double delta = oo;

int tl = MAXSIZE+1;

for(int i=1; i<=B[0]; i++)

if (A[B[i]][te] > eps)//eps or 0???????????

{

double temp = b[B[i]]/A[B[i]][te];

if (delta == oo || temp < delta || temp == delta && B[i] < tl)

{

delta = temp;

tl = B[i];

}

}

if (tl == MAXSIZE+1) return false;

if (delta\*c[te] > maxUp)

{

maxUp = delta\*c[te];

l = tl;

e = te;

}

}

if (maxUp == -1) break;

pivot(l, e);

}

return true;

}

void delete0()

{

int p;

for(p=1; p<=B[0]; p++)

if (B[p] == 0) break;

if (p <= B[0]) pivot(0, N[1]);

for(p=1; p<=N[0]; p++)

if (N[p] == 0) break;

for(int i=p; i<N[0]; i++)

N[i] = N[i+1];

N[0]--;

}

bool initialize()

{

N[0] = B[0] = 0;

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

N[++N[0]] = i;

for(int i=1; i<=m; i++)

B[++B[0]] = n+i;

v = 0;

int l = B[1];

for(int i=2; i<=B[0]; i++)

if (b[B[i]] < b[l])

l = B[i];

if (b[l] >= 0) return true;

double origC[MAXSIZE+1];

memcpy(origC, c, sizeof(double)\*(n+m+1));

N[++N[0]] = 0;

for(int i=1; i<=B[0]; i++)

A[B[i]][0] = -1;

memset(c, 0, sizeof(double)\*(n+m+1));

c[0] = -1;

pivot(l, 0);

opt();//unbounded????

if (v < -eps) return false;//eps?????????????

delete0();

memcpy(c, origC, sizeof(double)\*(n+m+1));

bool inB[MAXSIZE+1];

memset(inB, false, sizeof(bool)\*(n+m+1));

for(int i=1; i<=B[0]; i++)

inB[B[i]] = true;

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

if (inB[i] && c[i] != 0)

{

v += c[i]\*b[i];

for(int j=1; j<=N[0]; j++)

c[N[j]] -= A[i][N[j]]\*c[i];

c[i] = 0;

}

return true;

}

public: void simplex(string inputName, string outputName)

{

freopen(inputName.c\_str(), "r", stdin);

freopen(outputName.c\_str(), "w", stdout);

read();

if (!initialize())

{

printf("Infeasible\n");

return;

}

if (!opt())

{

printf("Unbounded\n");

return;

}

else printf("Max value is %lf\n", v);

bool inN[MAXSIZE+1];

memset(inN, false, sizeof(bool)\*(n+m+1));

for(int i=1; i<=N[0]; i++)

inN[N[i]] = true;

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

if (inN[i]) printf("x%d = %lf\n", i, 0.0);

else printf("x%d = %lf\n", i, b[i]);

}

};

int main()

{

LinearProgramming test;

test.simplex("a.in", "a.out");

}

# 数据结构

## 回文串

for(int i = 1, j = 0; i != (n << 1)- 1; ++ i)

{

int p = i >> 1, q = i - p, r = ((j + 1) >> 1) + l[j] - 1;

l[i] = r < q? 0: min(r - q + 1, l[(j << 1) - i]);

while(p - l[i] != -1 && q + l[i] != n && s[p - l[i]] == s[q + l[i]])

l[i] ++;

if(q + l[i] - 1 > r)

j = i;

a += l[i];

}

## 后缀数组（DC3）

/\* len should be greater than or equal to 2 - precondition for DC3 to execute correctly \*/

#include <cstdio>

#include <algorithm>

#define ALPHABET\_SIZE 1000001

using namespace std;

const int MAX\_N = 70000;

inline bool leq(int a1, int a2, int b1, int b2) {

return a1 < b1 || a1 == b1 && a2 <= b2;

}

inline bool leq(int a1, int a2, int a3, int b1, int b2, int b3) {

return a1 < b1 || a1 == b1 && leq(a2, a3, b2, b3);

}

int radixCnt[ALPHABET\_SIZE + 1];

inline void radixPass(int \*a, int \*b, int \*r, int n, int K) {

fill(radixCnt, radixCnt + K + 1, 0);

for (int i = 0; i < n; i++)

radixCnt[r[a[i]]]++;

for (int i = 0, sum = 0; i <= K; i++) {

int t = radixCnt[i]; radixCnt[i] = sum; sum += t;

}

for (int i = 0; i < n; i++)

b[radixCnt[r[a[i]]]++] = a[i];

}

#define GetI() (SA12[t] < n0 ? SA12[t] \* 3 + 1 : (SA12[t] - n0) \* 3 + 2)

int stackR[MAX\_N \* 4], stackSA12[MAX\_N \* 4], stackR0[MAX\_N \* 2], stackSA0[MAX\_N \* 2];

int allocR, allocSA12, allocR0, allocSA0;

void suffixArray(int\* T, int\* SA, int n, int K) {

int n0 = (n + 2) / 3, n1 = (n + 1) / 3, n2 = n / 3, n02 = n0 + n2;

int \*R = stackR + allocR, \*SA12 = stackSA12 + allocSA12;

allocR += n02 + 3;

allocSA12 += n02 + 3;

if (allocR >= MAX\_N \* 4)

for (int i = 0; i > -1; ++i)

printf("%d\n", i);

fill(R + n02, R + n02 + 3, 0);

fill(SA12 + n02, SA12 + n02 + 3, 0);

for (int i = 0, j = 0; i < n + (n0 - n1); i++)

if (i % 3 != 0)

R[j++] = i;

radixPass(R, SA12, T + 2, n02, K);

radixPass(SA12, R, T + 1, n02, K);

radixPass(R, SA12, T, n02, K);

int name = 0, c0 = -1, c1 = -1, c2 = -1;

for (int i = 0; i < n02; i++) {

if (T[SA12[i]] != c0 || T[SA12[i] + 1] != c1 || T[SA12[i] + 2] != c2) {

name++; c0 = T[SA12[i]]; c1 = T[SA12[i] + 1]; c2 = T[SA12[i] + 2];

}

if (SA12[i] % 3 == 1)

R[SA12[i] / 3] = name;

else

R[SA12[i] / 3 + n0] = name;

}

if (name < n02) {

suffixArray(R, SA12, n02, name);

for (int i = 0; i < n02; ++i)

R[SA12[i]] = i + 1;

} else

for (int i = 0; i < n02; ++i)

SA12[R[i] - 1] = i;

int \*R0 = stackR0 + allocR0, \*SA0 = stackSA0 + allocSA0;

allocR0 += n0;

allocSA0 += n0;

for (int i = 0, j = 0; i < n02; i++)

if (SA12[i] < n0)

R0[j++] = 3 \* SA12[i];

radixPass(R0, SA0, T, n0, K);

for (int p = 0, t = n0 - n1, k = 0; k < n; k++) {

int i = GetI();

int j = SA0[p];

if (SA12[t] < n0 ?

leq(T[i], R[SA12[t] + n0], T[j], R[j / 3]) :

leq(T[i], T[i + 1], R[SA12[t] - n0 + 1], T[j], T[j + 1], R[j / 3 + n0])) {

SA[k] = i;

if (++t == n02)

for (k++; p < n0; p++, k++)

SA[k] = SA0[p];

} else {

SA[k] = j;

if (++p == n0)

for (k++; t < n02; t++, k++)

SA[k] = GetI();

}

}

allocR -= n02 + 3;

allocSA12 -= n02 + 3;

allocSA0 -= n0;

allocR0 -= n0;

}

/\* len should be greater than or equal to 2 - precondition for DC3 to execute correctly \*/

static void suffixArray(int len, int \*x, int \*sa, int \*rank, int \*height, int alphaSize) {

allocR = allocSA12 = allocR0 = allocSA0 = 0;

suffixArray(x, sa, len, alphaSize);

for (int i = 0; i < len; ++i)

rank[sa[i]] = i;

height[0] = 0;

for (int i = 0, matched = 0, prev; i < len; ++i) {

if (rank[i] == 0) { matched = 0; continue; }

prev = sa[rank[i] - 1];

while (x[i + matched] == x[prev + matched])

++matched;

height[rank[i]] = matched;

if (matched > 0)

--matched;

}

}

## 后缀数组(nlogn)

//Suffix array

//n为串长度 a为原串

int n,a[20010],sa[20010],rank[20010],height[20010];

void build()

{

a[n+1]=-1;

void sort(int \*);

int count(int \*,int \*);

int b[20010],c[20010];

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

{

c[i]=a[i];

b[i]=-1;

sa[i]=i;

}

sort(c);

count(c,b);

int k=1;

while (1)

{

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

{

c[i]=rank[i];

if (i+k<=n)

b[i]=rank[i+k];

else

b[i]=0;

}

sort(b);

sort(c);

if (count(c,b)>=n)

break;

k<<=1;

}

k=0;

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

{

k=k?k-1:0;

if (rank[i]==1)

{

height[rank[i]]=0;

continue;

}

int p=sa[rank[i]-1],q=sa[rank[i]];

while (a[p+k]==a[q+k])

k++;

height[rank[i]]=k;

}

}

void sort(int \*a)

{

int f[20010],x[20010],t=0;

memset(f,0,sizeof(f));

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

{

f[a[i]]++;

if (a[i]>t)

t=a[i];

}

for (int i=1;i<=t;i++)

f[i]+=f[i-1];

for (int i=n;i>=1;i--)

{

x[f[a[sa[i]]]]=sa[i];

f[a[sa[i]]]--;

}

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

sa[i]=x[i];

}

int count(int \*a,int \*b)

{

rank[sa[1]]=1;

int t=1;

for (int i=2;i<=n;i++)

{

if (a[sa[i]]!=a[sa[i-1]] || b[sa[i]]!=b[sa[i-1]])

t++;

rank[sa[i]]=t;

}

return(t);

}

## 后缀自动机

struct State {

int length;

State \*parent;

State \*go[C];

State(int length): length(length), parent(NULL) {

memset(go, 0, sizeof(go));

}

State\* extend(State \*start, int token) {

State \*p = this;

State \*np = new State(this->length + 1);

while (p != NULL && p->go[token] == NULL) {

p->go[token] = np;

p = p->parent;

}

if (p == NULL) {

np->parent = start;

} else {

State \*q = p->go[token];

if (p->length + 1 == q->length) {

np->parent = q;

} else {

State \*nq = new State(p->length + 1);

memcpy(nq->go, q->go, sizeof(q->go));

nq->parent = q->parent;

np->parent = q->parent = nq;

while (p != NULL && p->go[token] == q) {

p->go[token] = nq;

p = p->parent;

}

}

}

return np;

}

};

## 扩展KMP

//BEGIN  
//extended KMP  
void ExtendedKMP(char \*a, char \*b, int M, int N, int \*Next, int \*ret) {// a -> 模式串 b -> 匹配串  
    int i, j, k;  
    for (j = 0; 1 + j < M && a[j] == a[1 + j]; j++);  
    Next[1] = j;  
    k = 1;  
    for (i = 2; i < M; i++) {  
        int Len = k + Next[k], L = Next[i - k];  
        if (L < Len - i) {  
            Next[i] = L;  
        } else {  
            for (j = max(0, Len - i); i + j < M && a[j] == a[i + j]; j++);  
            Next[i] = j;  
            k = i;  
        }  
    }  
    for (j = 0; j < N && j < M && a[j] == b[j]; j++);  
    ret[0] = j;  
    k = 0;  
    for (i = 1; i < N; i++) {  
        int Len = k + ret[k], L = Next[i - k];  
        if (L < Len - i) {  
            ret[i] = L;  
        } else {  
            for (j = max(0, Len - i); j < M && i + j < N && a[j] == b[i + j]; j++);  
            ret[i] = j;  
            k = i;  
        }  
    }  
}  
//END

## 动态树

/\*

Expose(x) 求出x到根的路径

Modify(x, co) 将x改成co

Query(x, y) 询问x到y的路径

Join(x, y) 添加edge(x, y)

Cut(x, y) 删除edge(x, y)

\*/

//BEGIN TEMPLATE HERE

#define SIZE(X) ((int)(X.size()))

#define LENGTH(X) ((int)(X.length()))

//END TEMPLATE HERE

const int maxn = 11000;

char op[100];

int N, Q;

int lc[maxn], rc[maxn], fa[maxn], Sum[maxn], Size[maxn], Rev[maxn], color[maxn];

int List[maxn], total;

inline bool isroot(int x) {

if (fa[x] == 0) return true;

return x != lc[fa[x]] && x != rc[fa[x]];

}

inline void update(int x) {

Sum[x] = Sum[lc[x]] + Sum[rc[x]] + color[x];

Size[x] = Size[lc[x]] + Size[rc[x]] + 1;

}

inline void Reverse(int x) {

if (Rev[x]) {

if (lc[x]) Rev[lc[x]] ^= 1;

if (rc[x]) Rev[rc[x]] ^= 1;

swap(lc[x], rc[x]);

Rev[x] = 0;

}

}

inline void right(int x, int y) {

lc[y] = rc[x];

if (lc[y]) fa[lc[y]] = y;

rc[x] = y;

fa[x] = fa[y];

if (fa[y]) {

if (y == lc[fa[y]]) {

lc[fa[x]] = x;

} else if (y == rc[fa[y]]) {

rc[fa[x]] = x;

}

}

fa[y] = x;

update(y);

update(x);

}

inline void left(int x, int y) {

rc[y] = lc[x];

if (rc[y]) fa[rc[y]] = y;

lc[x] = y;

fa[x] = fa[y];

if (fa[y]) {

if (y == lc[fa[y]]) {

lc[fa[x]] = x;

} else if (y == rc[fa[y]]) {

rc[fa[x]] = x;

}

}

fa[y] = x;

update(y);

update(x);

}

void splay(int t) {

List[total = 1] = t;

for (int x = t; !isroot(x); x = fa[x]) List[++total] = fa[x];

for (; total; --total) {

if (Rev[List[total]]) Reverse(List[total]);

}

for (; !isroot(t); ) {

int f = fa[t];

if (isroot(f)) {

if (t == lc[f]) {

right(t, f);

} else {

left(t, f);

}

} else {

int ff = fa[f];

if (f == lc[ff]) {

if (t == lc[f]) {

right(f, ff);

right(t, f);

} else {

left(t, f);

right(t, ff);

}

} else {

if (t == rc[f]) {

left(f, ff);

left(t, f);

} else {

right(t, f);

left(t, ff);

}

}

}

}

}

int Expose(int u) {

int v = 0;

for (; u; u = fa[u]) {

splay(u); rc[u] = v; update(u); v = u;

}

for (; lc[v]; v = lc[v]);

return v;

}

void Join(int x, int y) {

int fx = Expose(x);

int fy = Expose(y);

if (fx != fy) {

Expose(x);

splay(x);

rc[x] = 0; fa[x] = y; Rev[x] = true; Reverse(x); update(x);

}

}

void Cut(int x, int y) {

int fx = Expose(x);

int fy = Expose(y);

if (fx == fy) {

Expose(x);

splay(x);

bool flag = false;

if (lc[x]) {

int k;

for (k = lc[x]; rc[k]; k = rc[k]);

if (k == y) {

flag = true;

}

}

if (flag) {

fa[lc[x]] = 0;

lc[x] = 0;

update(x);

} else {

Expose(y);

splay(y);

fa[lc[y]] = 0;

lc[y] = 0;

update(y);

}

}

}

void Modify(int x, char co) {

splay(x); color[x] = co == 'B'; update(x);

}

void Query(int x, int y) {

int fx = Expose(x);

int fy = Expose(y);

if (fx != fy) {

puts("-1");

} else {

for (int u = x, v = 0; u; u = fa[u]) {

splay(u);

if (fa[u] == 0) {

int cnt = Size[rc[u]] + Size[v] + 1;

int cntB = Sum[rc[u]] + Sum[v] + color[u];

printf("%d %d\n", cntB, cnt - cntB);

return;

}

rc[u] = v; update(u); v = u;

}

}

}

int main() {

while (scanf("%d%d", &N, &Q) == 2 && (N || Q)) {

memset(lc, 0, sizeof lc);

memset(rc, 0, sizeof rc);

memset(fa, 0, sizeof fa);

memset(Sum, 0, sizeof Sum);

memset(Size, 0, sizeof Size);

memset(Rev, 0, sizeof Rev);

memset(color, 0, sizeof color);

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

char co;

scanf(" %c", &co);

Size[i] = 1;

Modify(i, co);

}

for (int i = 0; i < Q; ++i) {

int x, y;

char co;

scanf("%s", op);

if (op[0] == 'q') {

scanf("%d%d", &x, &y);

Query(x, y);

} else if (op[0] == 'a') {

scanf("%d%d", &x, &y);

Join(x, y);

} else if (op[0] == 'd') {

scanf("%d%d", &x, &y);

Cut(x, y);

} else if (op[0] == 's') {

scanf("%d %c", &x, &co);

Modify(x, co);

}

}

}

return 0;

}

## KD-Tree

#include <iostream>

#include <algorithm>

using namespace std;

#define sqr(x) ((long long)(x) \* (x))

const long long inf = 1000000000000000000LL;

struct TP {

int x, y;

}a[101000], P, ord[100100];

int n;

int max(const int &a, const int &b)

{

return a > b ? a : b;

}

int min(const int &a, const int &b)

{

return a < b ? a : b;

}

inline long long dis2(const TP &a, const TP &b)

{

return sqr(a.x - b.x) + sqr(a.y - b.y);

}

inline bool cmpx(const TP &a, const TP &b)

{

return a.x < b.x || a.x == b.x && a.y < b.y;

}

inline bool cmpy(const TP &a, const TP &b)

{

return a.y < b.y || a.y == b.y && a.x < b.x;

}

struct TR {

int minx, maxx, miny, maxy;

inline void rect(const TP &a)

{

minx = maxx = a.x;

miny = maxy = a.y;

}

inline void merge(const TR &a)

{

minx = min(minx, a.minx);

miny = min(miny, a.miny);

maxx = max(maxx, a.maxx);

maxy = max(maxy, a.maxy);

}

inline long long dis2(const TP &a)

{

if (a.x <= minx && a.y <= miny) return sqr(a.x - minx) + sqr(a.y - miny);

if (a.x <= maxx && a.y <= miny) return sqr(a.y - miny);

if (a.x >= maxx && a.y <= miny) return sqr(a.x - maxx) + sqr(a.y - miny);

if (a.x >= maxx && a.y <= maxy) return sqr(a.x - maxx);

if (a.x >= maxx && a.y >= maxy) return sqr(a.x - maxx) + sqr(a.y - maxy);

if (a.x >= minx && a.y >= maxy) return sqr(a.y - maxy);

if (a.x <= minx && a.y >= maxy) return sqr(a.x - minx) + sqr(a.y - maxy);

if (a.x <= minx && a.y <= maxy) return sqr(a.x - minx);

return 0;

}

};

struct TT {

TP m;

TR rt;

}Tree[310100];

inline void Build(int now, int l, int r, int dep)

{

if (l >= r) return;

int mid = ((l + r) >> 1);

nth\_element(a + l, a + mid, a + r, dep ? cmpx : cmpy);

Tree[now].m = a[mid];

Tree[now].rt.rect(a[mid]);

if (l == r) return;

Build(now << 1, l, mid, !dep);

Build((now << 1) + 1, mid + 1, r, !dep);

if (l < mid) Tree[now].rt.merge(Tree[now << 1].rt);

if (mid + 1 < r) Tree[now].rt.merge(Tree[(now << 1) + 1].rt);

}

long long res;

inline void ask(int now, int l, int r, int dep)

{

int mid = ((l + r) >> 1);

if (Tree[now].rt.dis2(P) >= res) return;

long long d = dis2(P, Tree[now].m);

if (d && d < res) res = d;

if (dep && cmpx(P, Tree[now].m) || !dep && cmpy(P, Tree[now].m)) {

if (l < mid) ask(now << 1, l, mid, !dep);

if (mid + 1 < r) ask((now << 1) + 1, mid + 1, r, !dep);

} else {

if (mid + 1 < r) ask((now << 1) + 1, mid + 1, r, !dep);

if (l < mid) ask(now << 1, l, mid, !dep);

}

}

int main()

{

freopen("k.in", "r", stdin);

freopen("k.out", "w", stdout);

int T;

for (scanf("%d", &T); T; T--) {

scanf("%d", &n);

for (int i = 0; i < n; i++) {

scanf("%d%d", &a[i].x, &a[i].y);

ord[i] = a[i];

}

Build(1, 0, n, 0);

for (int i = 0; i < n; i++) {

P = ord[i];

res = inf;

ask(1, 0, n, 0);

printf("%lld\n", res);

}

}

}

## AC自动机

struct trie

{

char ch;

int son,next,father,suffix;

vector <int> danger;

};

trie a[10000];

int now,m;

void clear(int x)

{

a[x].son=a[x].next=0;

a[x].danger.clear();

}

void insert(char \*s,int l,int t,int x)

{

if (!a[x].son)

{

a[x].son=++m;

clear(m);

a[m].father=x;

a[m].ch=s[t];

if (t+1==l)

a[m].danger.push\_back(now);

else

insert(s,l,t+1,m);

return;

}

int i=a[x].son;

while (1)

{

if (!a[i].next || a[i].ch==s[t])

break;

i=a[i].next;

}

if (a[i].ch==s[t] && t+1==l)

a[i].danger.push\_back(now);

else if (a[i].ch==s[t])

insert(s,l,t+1,i);

else

{

a[i].next=++m;

clear(m);

a[m].father=x;

a[m].ch=s[t];

if (t+1==l)

a[m].danger.push\_back(now);

else

insert(s,l,t+1,m);

}

}

int q[100000];

int child(int x,char ch)

{

for (int i=a[x].son;i;i=a[i].next)

if (a[i].ch==ch)

return(i);

if (x==1)

return(1);

return(child(a[x].suffix,ch));

}

void build\_trie()

{

int l,r;

l=r=1;

q[1]=1;

while (l<=r)

{

int x=q[l++];

for (int i=a[x].son;i;i=a[i].next)

q[++r]=i;

}

a[1].suffix=1;

for (int i=2;i<=r;i++)

{

int x=q[i];

if (a[x].father==1)

{

a[x].suffix=1;

continue;

}

a[x].suffix=child(a[a[x].father].suffix,a[x].ch);

for (int j=0;j<a[a[x].suffix].danger.size();j++)

a[x].danger.push\_back(a[a[x].suffix].danger[j]);

}

}

int main()

{

clear(m=1);

for (int i=0;i<n;i++)

{

scanf("%s",s);

now=i;

insert(s,strlen(s),0,1);

}

build\_trie();

}

## 左偏树

//Leftist tree

//v值 l左儿子 r右儿子 d深度

int v[100001],l[100001],r[100001],d[100001];

int merge(int x,int y)

{

if (!x)

return(y);

if (!y)

return(x);

if (v[x]<v[y])

swap(x,y);

r[x]=merge(r[x],y);

if (d[l[x]]<d[r[x]])

swap(l[x],r[x]);

d[x]=d[r[x]]+1;

return(x);

}

# 杂

## 字符串最小表示

#include <string>

std::string find(std::string s) {

int i, j, k, l;

int N = s.length();

s += s;

for (i = 0, j = 1; j < N; ) {

for (k = 0; k < N && s[i + k] == s[j + k]; k ++);

if (k >= N) break;

if (s[i + k] < s[j + k]) {

j += k + 1;

} else {

l = i + k;

i = j;

j = max(l, j) + 1;

}

}

return s.substr(i, N);

}

## 曼哈顿最小生成树

#include <vector>

#include <list>

#include <set>

#include <map>

#include <stack>

#include <deque>

#include <queue>

#include <bitset>

#include <functional>

#include <numeric>

#include <utility>

#include <complex>

#include <string>

#include <iomanip>

#include <sstream>

#include <fstream>

#include <iostream>

#include <algorithm>

#include <cstdio>

#include <cmath>

#include <ctime>

#include <cctype>

#include <cstdlib>

#include <cstring>

#include <cassert>

using namespace std;

#define SIZE(X) ((int)(X.size()))

const int maxn = 110000;

struct Tpoint {

int x, y, id;

} a[maxn];

int N;

int father[maxn];

void Rotate(Tpoint &a) {

int t = a.x;

a.x = -a.y;

a.y = t;

}

int find(int x) {

int t, tt;

for (t = x; father[t] >= 0; t = father[t]);

for (; father[x] >= 0; ) {

tt = father[x];

father[x] = t;

x = tt;

}

return t;

}

void Union(int x, int y) {

if (-father[x] > -father[y]) swap(x, y);

father[y] += father[x];

father[x] = y;

}

struct Tedge {

int x, y, z;

Tedge() {

}

Tedge(int x, int y, int z): x(x), y(y), z(z) {

}

bool operator <(const Tedge &a) const {

return z < a.z;

}

};

vector<Tedge> Edge;

void Kruskal() {

sort(Edge.begin(), Edge.end());

for (int i = 0; i < N; ++i) father[i] = -1;

long long ans = 0;

for (int i = 0, kn = 0; i < SIZE(Edge) && kn < N - 1; i++) {

int kx = find(Edge[i].x), ky = find(Edge[i].y);

if (kx != ky) {

ans += Edge[i].z;

Union(kx, ky);

kn++;

}

}

printf("%lld\n", ans);

}

const int inf = 1<<30;

Tpoint cp[maxn];

int dp[maxn], rec[maxn], yl[maxn];

bool cmp(const Tpoint &a, const Tpoint &b) {

return a.y - a.x < b.y - b.x || a.y - a.x == b.y - b.x && a.y > b.y;

}

inline int lowbit(int x) {

return (x & (-x));

}

inline int calc(const Tpoint &s, const Tpoint &t) {

return abs(s.x - t.x) + abs(s.y - t.y);

}

void Work() {

for (int i = 0; i < N; ++i) {

cp[i] = a[i];

}

for (int i = 0; i < N; ++i) {

yl[i] = cp[i].y;

}

sort(yl, yl + N);

int tot = unique(yl, yl + N) - yl;

sort(cp, cp + N, cmp);

for (int i = 1; i <= tot; ++i) dp[i] = inf;

for (int i = 0; i < N; ++i) {

int id = -1, res = inf;

int pos = 1 + (int)(lower\_bound(yl, yl + tot, cp[i].y) - yl);

for (int x = pos; x <= tot; x += lowbit(x)) {

if (dp[x] < res) {

res = dp[x];

id = rec[x];

}

}

if (id != -1) {

Edge.push\_back(Tedge(cp[i].id, id, calc(a[cp[i].id], a[id])));

}

res = cp[i].x + cp[i].y;

for (int x = pos; x > 0; x -= lowbit(x)) {

if (res < dp[x]) {

dp[x] = res;

rec[x] = cp[i].id;

}

}

}

}

void main2() {

for (int i = 0; i < N; ++i) {

scanf("%d%d", &a[i].x, &a[i].y);

a[i].id = i;

}

Edge.clear();

//case 1

Work();

//case 2

for (int j = 0; j < N; ++j) swap(a[j].x, a[j].y);

Work();

//case 3

for (int j = 0; j < N; ++j) swap(a[j].x, a[j].y);

for (int j = 0; j < N; ++j) {

Rotate(a[j]);

}

Work();

//case 4

for (int j = 0; j < N; ++j) swap(a[j].x, a[j].y);

Work();

Kruskal();

}

int main() {

for (int caseId = 1; scanf("%d", &N) == 1 && N; caseId++) {

printf("Case %d: Total Weight = ", caseId);

main2();

}

return 0;

}

## 表达式计算

// PKU 1686 Lazy Math Instructor

#include <cstdio>

#include <cstring>

#include <cctype>

#include <ctime>

#include <cstdlib>

const int maxl = 1000;

const int maxt = 100;

const double eps = 1e-8;

int value[26];

char str1[maxl], str2[maxl];

void Get\_Str(char str[]) {

gets(str);

int i, len = 0;

for (i = 0; str[i]; i++)

if (str[i] > ' ') str[len++] = str[i];

str[len] = 0;

}

void Init() {

Get\_Str(str1);

Get\_Str(str2);

}

inline int Level(char ch) {

switch (ch) {

case '+' :

case '-' : return 0;

case '\*' : return 1;

}

return -1;

}

int Calc(const char \*&p, int level) {

int res;

if (level == 2) {

if (\*p == '(') {

p++;

res = Calc(p, 0);

p++;

} else {

res = isdigit(\*p) ? \*p - '0' : value[\*p - 'a'];

p++;

}

return res;

}

res = Calc(p, level + 1);

char ch;

int next;

while (\*p && Level(\*p) == level) {

ch = \*p++;

next = Calc(p, level + 1);

switch (ch) {

case '+' : res += next; break;

case '-' : res -= next; break;

case '\*' : res \*= next; break;

}

}

return res;

}

int Evaluate(const char \*str) {

const char \*p = str;

return Calc(p, 0);

}

void Work() {

int i, j;

value[0] = 1;

for (i = 0; i < maxt; i++) {

for (j = 0; j < 26; j++)

value[j] = rand();

if (Evaluate(str1) != Evaluate(str2)) {

printf("NO\n");

return;

}

}

printf("YES\n");

}

int main() {

int tt = 0;

scanf("%d", &tt);

gets(str1);

while (tt--) {

Init();

Work();

}

return 0;

}

## DancingLinks

Procedure Algorithm\_X(Dep)

如果矩阵中所有的列均被删除, 找到一组合法解, 退出．

任意选择一个未被删除的列c,

枚举一个未被删除的行r, 且Matrix[r][c] = 1, 将(r, c)加入Ans．

枚举所有的列j, Matrix[r][j] = 1, 将第j列删除．

枚举所有的行i, Matrix[i][j] = 1, 将第i行删除．

Algorithm\_X(Dep + 1)

Procedure Algorithm\_X(Dep)

如果h^.right = h(即所有的列均被删除), 找到一组解, 退出．

利用h和right指针找到一个c, 满足size[c]最小．

如果size[c] = 0(当前列无法被覆盖), 无解, 退出．

Cover(c)

for (i = c^.down; i != c; i ← i^.down)

for (j = i^.right; j != i; j ← j^.right) Cover(j^.col)

将i结点加入Ans, Algorithm\_X(Dep + 1)

for (j = i^.left; j != i; j ← j^.left) Recover(j^.col)

Recover(c)

Soduku问题可以转化一个Exact Cover Problem：16 \* 16 \* 16行, (i, j, k)表示(i, j)这个格子填上字母k．16 \* 16 \* 4列分别表示第i行中的字母k, 第i列中的字母k, 第i个子矩阵中的字母k, 以及(i, j)这个格子．对于每个集合(i, j, k), 它包含了4个元素：Line(i, k), Col(j, k), Sub(P[i][j], k), Grid(i, j), 其中P[i][j]表示(i, j)这个格子所属的子矩阵．本题转化为一个4096行, 1024列, 且1的个数为16384个的矩阵．下面介绍解决一般的Exact Cover Problem的Algorithm X．

N皇后问题：关键是构建Exact Cover问题的矩阵：N \* N行对应了N \* N个格子, 6N-2列对应了N行, N列, 2N-1条主对角线, 2N-1条副对角线．第i行共4个1, 分别对应(i, j)这个格子所处的行, 列, 主对角线和副对角线．直接对这个矩阵作Algorithm X是错误的, 虽然每行, 每列都恰好被覆盖一次, 但是对角线是最多覆盖一次, 它可以不被覆盖, 这与Exact Cover问题的定义是不同的．

有两种处理的方法：

1) 新增4N-2行, 每行只有一个1, 分别对应了2N-1条主对角线和2N-1条副对角线, 这样就可以保证某个对角线不被覆盖的时候, 可以使用新增行来覆盖．

2) 每次选择一个size[]值最小的列c进行覆盖, 而这一步, 我们忽略掉所有的对角线列, 只考虑c为行和列的情况．

事实证明, 第2)种方法的效果好很多, 因此这个问题可以使用Algorithm X轻松得到解决．

struct data

{

int l,r,u,d,x,y;

};

data a[5101];

int sum[310];

void del(int x)

{

a[a[x].l].r=a[x].r;

a[a[x].r].l=a[x].l;

for (int i=a[x].d;i!=x;i=a[i].d)

for (int j=a[i].r;j!=i;j=a[j].r)

{

sum[a[j].y]--;

a[a[j].u].d=a[j].d;

a[a[j].d].u=a[j].u;

}

}

void renew(int x)

{

a[a[x].l].r=x;

a[a[x].r].l=x;

for (int i=a[x].u;i!=x;i=a[i].u)

for (int j=a[i].l;j!=i;j=a[j].l)

{

sum[a[j].y]++;

a[a[j].u].d=j;

a[a[j].d].u=j;

}

}

bool search()

{

if (a[0].r==0)

return(true);

int k,min=20000000;

for (int i=a[0].r;i!=0;i=a[i].r)

if (sum[i]<min)

min=sum[k=i];

del(k);

for (int i=a[k].d;i!=k;i=a[i].d)

{

for (int j=a[i].r;j!=i;j=a[j].r)

del(a[j].y);

if (search())

return(true);

for (int j=a[i].l;j!=i;j=a[j].l)

renew(a[j].y);

}

renew(k);

return(false);

}

## 最长公共子序列

const int dx[]={0,-1,0,1};

const int dy[]={1,0,-1,0};

const string ds="ENWS";

char G[52][52];

char A[22222], B[22222], buf[22222];

int n, m;

typedef unsigned long long ll;

const int M = 62;

const int maxn = 20010;

const int maxt = 130;

const int maxl = maxn / M + 10;

const ll Top = ((ll) 1 << (M));

const ll Topless = Top - 1;

const ll underTop = ((ll) 1 << (M - 1));

typedef ll bitarr[maxl];

bitarr comp[maxt], row[2], X;

void get(char \*S){

int L,x,y,sz=0;

scanf("%d%d%d",&L,&x,&y),x--,y--;

//scanf(" %s",buf);

S[sz++]=G[x][y];

for(int i=0;i<L;i++){

char ch;

scanf(" %c", &ch);

int pos=ds.find(ch);

x+=dx[pos],y+=dy[pos];

if (x < 0 || y < 0 || x >= n || y >= m) for(;;);

S[sz++]=G[x][y];

}

S[sz]=0;

}

bool calc[maxt];

void prepare() {

int u, p;

memset(calc, 0, sizeof(calc));

for (int i = 0; i < m; i++) {

u = B[i];

if (calc[u]) continue; //======仅对所有字符集 ，每次一次

calc[u] = 1;

memset(comp[u], 0, sizeof(comp[u]));

for (p = 0; p < n; p++) if (u == A[p]) comp[u][p / M] ^= ((ll) 1 << (p % M));

}

}

void solve() {

prepare();

memset(row, 0, sizeof(row));

int prev, curt;

int i, u, p, c, cc;

int Ln = (n / M) + 1;

prev = 0;

for (i = 0; i < m; i++) {

curt = 1 - prev; u = B[i];

for (p = 0; p < Ln; p++) X[p] = row[prev][p] | comp[u][p];

c = 0;

for (p = 0; p < Ln; p++) {

cc = (row[prev][p] & underTop) > 0;

row[prev][p] = ((row[prev][p] & (underTop - 1)) << 1) + c;

c = cc;

}

for (p = 0; p < Ln; p++) {

if (row[prev][p] != Topless) {

row[prev][p]++;

break;

}

row[prev][p] = 0;

}

c = 0;

for (p = 0; p < Ln; p++) {

if (X[p] >= row[prev][p] + c)

row[prev][p] = X[p] - (row[prev][p] + c), c = 0;

else

row[prev][p] = Top + X[p] - (row[prev][p] + c), c = 1;

}

for (p = 0; p < Ln; p++)

row[curt][p] = X[p] & (row[prev][p] ^ X[p]);

prev = curt;

}

int ret = 0;

for (i = 0; i < n; i++)

if (row[prev][i / M] & ((ll) 1 << (i % M))) ret++;

// printf("%d %d %d\n", n, m, ret);

//=========ret 就是最长公共子序列。

printf("%d %d\n", n - ret, m - ret);

}

int main(){

int tests=0,T;

scanf("%d",&T);

while(T--){

scanf("%d%d",&n,&m);

for(int i=0;i<n;i++)

for (int j = 0; j < m; j++)

scanf(" %c",&G[i][j]);

get(A),get(B);

printf("Case %d: ", ++tests);

// printf("A = %s\n, B = %s\n", A, B);

n = strlen(A), m = strlen(B);

//n = 20000; m = 20000;

//for (int i = 0; i < m; i++) A[i] = B[i] = 'A';

//A[m] = B[m] = 0;

solve();

}

}

## 高精度计算

#include <iostream>

#include <string>

using namespace std;

#define DIGIT 4

#define DEPTH 10000

#define MAX 100

typedef int bignum\_t[MAX+1];

int read(bignum\_t a,istream& is=cin){//Read an unsiged bignum\_t from cin(can be changed to fit scanf)

char buf[MAX\*DIGIT+1],ch;

int i,j;

memset((void\*)a,0,sizeof(bignum\_t));

if (!(is>>buf)) return 0;

for (a[0]=strlen(buf),i=a[0]/2-1;i>=0;i--)

ch=buf[i],buf[i]=buf[a[0]-1-i],buf[a[0]-1-i]=ch;

for (a[0]=(a[0]+DIGIT-1)/DIGIT,j=strlen(buf);j<a[0]\*DIGIT;buf[j++]='0');

for (i=1;i<=a[0];i++)

for (a[i]=0,j=0;j<DIGIT;j++)

a[i]=a[i]\*10+buf[i\*DIGIT-1-j]-'0';

for (;!a[a[0]]&&a[0]>1;a[0]--);

return 1;

}

void write(const bignum\_t a,ostream& os=cout){// Write down on cout

int i,j;

for (os<<a[i=a[0]],i--;i;i--)

for (j=DEPTH/10;j;j/=10)

os<<a[i]/j%10;

}

int comp(const bignum\_t a,const bignum\_t b){

int i;

if (a[0]!=b[0])

return a[0]-b[0];

for (i=a[0];i;i--)

if (a[i]!=b[i])

return a[i]-b[i];

return 0;

}

int comp(const bignum\_t a,const int b){

int c[12]={1};

for (c[1]=b;c[c[0]]>=DEPTH;c[c[0]+1]=c[c[0]]/DEPTH,c[c[0]]%=DEPTH,c[0]++);

return comp(a,c);

}

int comp(const bignum\_t a,const int c,const int d,const bignum\_t b){

int i,t=0,O=-DEPTH\*2;

if (b[0]-a[0]<d&&c)

return 1;

for (i=b[0];i>d;i--){

t=t\*DEPTH+a[i-d]\*c-b[i];

if (t>0) return 1;

if (t<O) return 0;

}

for (i=d;i;i--){

t=t\*DEPTH-b[i];

if (t>0) return 1;

if (t<O) return 0;

}

return t>0;

}

void add(bignum\_t a,const bignum\_t b){

int i;

for (i=1;i<=b[0];i++)

if ((a[i]+=b[i])>=DEPTH)

a[i]-=DEPTH,a[i+1]++;

if (b[0]>=a[0])

a[0]=b[0];

else

for (;a[i]>=DEPTH&&i<a[0];a[i]-=DEPTH,i++,a[i]++);

a[0]+=(a[a[0]+1]>0);

}

void add(bignum\_t a,const int b){

int i=1;

for (a[1]+=b;a[i]>=DEPTH&&i<a[0];a[i+1]+=a[i]/DEPTH,a[i]%=DEPTH,i++);

for (;a[a[0]]>=DEPTH;a[a[0]+1]=a[a[0]]/DEPTH,a[a[0]]%=DEPTH,a[0]++);

}

void sub(bignum\_t a,const bignum\_t b){

int i;

for (i=1;i<=b[0];i++)

if ((a[i]-=b[i])<0)

a[i+1]--,a[i]+=DEPTH;

for (;a[i]<0;a[i]+=DEPTH,i++,a[i]--);

for (;!a[a[0]]&&a[0]>1;a[0]--);

}

void sub(bignum\_t a,const int b){

int i=1;

for (a[1]-=b;a[i]<0;a[i+1]+=(a[i]-DEPTH+1)/DEPTH,a[i]-=(a[i]-DEPTH+1)/DEPTH\*DEPTH,i++);

for (;!a[a[0]]&&a[0]>1;a[0]--);

}

void sub(bignum\_t a,const bignum\_t b,const int c,const int d){

int i,O=b[0]+d;

for (i=1+d;i<=O;i++)

if ((a[i]-=b[i-d]\*c)<0)

a[i+1]+=(a[i]-DEPTH+1)/DEPTH,a[i]-=(a[i]-DEPTH+1)/DEPTH\*DEPTH;

for (;a[i]<0;a[i+1]+=(a[i]-DEPTH+1)/DEPTH,a[i]-=(a[i]-DEPTH+1)/DEPTH\*DEPTH,i++);

for (;!a[a[0]]&&a[0]>1;a[0]--);

}

void mul(bignum\_t c,const bignum\_t a,const bignum\_t b){

int i,j;

memset((void\*)c,0,sizeof(bignum\_t));

for (c[0]=a[0]+b[0]-1,i=1;i<=a[0];i++)

for (j=1;j<=b[0];j++)

if ((c[i+j-1]+=a[i]\*b[j])>=DEPTH)

c[i+j]+=c[i+j-1]/DEPTH,c[i+j-1]%=DEPTH;

for (c[0]+=(c[c[0]+1]>0);!c[c[0]]&&c[0]>1;c[0]--);

}

void mul(bignum\_t a,const int b){

int i;

for (a[1]\*=b,i=2;i<=a[0];i++){

a[i]\*=b;

if (a[i-1]>=DEPTH)

a[i]+=a[i-1]/DEPTH,a[i-1]%=DEPTH;

}

for (;a[a[0]]>=DEPTH;a[a[0]+1]=a[a[0]]/DEPTH,a[a[0]]%=DEPTH,a[0]++);

for (;!a[a[0]]&&a[0]>1;a[0]--);

}

void mul(bignum\_t b,const bignum\_t a,const int c,const int d){

int i;

memset((void\*)b,0,sizeof(bignum\_t));

for (b[0]=a[0]+d,i=d+1;i<=b[0];i++)

if ((b[i]+=a[i-d]\*c)>=DEPTH)

b[i+1]+=b[i]/DEPTH,b[i]%=DEPTH;

for (;b[b[0]+1];b[0]++,b[b[0]+1]=b[b[0]]/DEPTH,b[b[0]]%=DEPTH);

for (;!b[b[0]]&&b[0]>1;b[0]--);

}

void div(bignum\_t c,bignum\_t a,const bignum\_t b){

int h,l,m,i;

memset((void\*)c,0,sizeof(bignum\_t));

c[0]=(b[0]<a[0]+1)?(a[0]-b[0]+2):1;

for (i=c[0];i;sub(a,b,c[i]=m,i-1),i--)

for (h=DEPTH-1,l=0,m=(h+l+1)>>1;h>l;m=(h+l+1)>>1)

if (comp(b,m,i-1,a)) h=m-1;

else l=m;

for (;!c[c[0]]&&c[0]>1;c[0]--);

c[0]=c[0]>1?c[0]:1;

}

void div(bignum\_t a,const int b,int& c){

int i;

for (c=0,i=a[0];i;c=c\*DEPTH+a[i],a[i]=c/b,c%=b,i--);

for (;!a[a[0]]&&a[0]>1;a[0]--);

}

void sqrt(bignum\_t b,bignum\_t a){

int h,l,m,i;

memset((void\*)b,0,sizeof(bignum\_t));

for (i=b[0]=(a[0]+1)>>1;i;sub(a,b,m,i-1),b[i]+=m,i--)

for (h=DEPTH-1,l=0,b[i]=m=(h+l+1)>>1;h>l;b[i]=m=(h+l+1)>>1)

if (comp(b,m,i-1,a)) h=m-1;

else l=m;

for (;!b[b[0]]&&b[0]>1;b[0]--);

for (i=1;i<=b[0];b[i++]>>=1);

}

int length(const bignum\_t a){

int t,ret;

for (ret=(a[0]-1)\*DIGIT,t=a[a[0]];t;t/=10,ret++);

return ret>0?ret:1;

}

int digit(const bignum\_t a,const int b){

int i,ret;

for (ret=a[(b-1)/DIGIT+1],i=(b-1)%DIGIT;i;ret/=10,i--);

return ret%10;

}

int zeronum(const bignum\_t a){

int ret,t;

for (ret=0;!a[ret+1];ret++);

for (t=a[ret+1],ret\*=DIGIT;!(t%10);t/=10,ret++);

return ret;

}

void comp(int\* a,const int l,const int h,const int d){

int i,j,t;

for (i=l;i<=h;i++)

for (t=i,j=2;t>1;j++)

while (!(t%j))

a[j]+=d,t/=j;

}

void convert(int\* a,const int h,bignum\_t b){

int i,j,t=1;

memset(b,0,sizeof(bignum\_t));

for (b[0]=b[1]=1,i=2;i<=h;i++)

if (a[i])

for (j=a[i];j;t\*=i,j--)

if (t\*i>DEPTH)

mul(b,t),t=1;

mul(b,t);

}

void combination(bignum\_t a,int m,int n){

int\* t=new int[m+1];

memset((void\*)t,0,sizeof(int)\*(m+1));

comp(t,n+1,m,1);

comp(t,2,m-n,-1);

convert(t,m,a);

delete []t;

}

void permutation(bignum\_t a,int m,int n){

int i,t=1;

memset(a,0,sizeof(bignum\_t));

a[0]=a[1]=1;

for (i=m-n+1;i<=m;t\*=i++)

if (t\*i>DEPTH)

mul(a,t),t=1;

mul(a,t);

}

## 图同构hash

枚举点a，迭代K次后求得的就是a点所对应的hash值。

其中K、A、B、C、D、P为hash参数，可自选。

## 双人零和矩阵游戏（公式）

N\*N的方阵A，选行的玩家的最优策略是p，选列的是q,则

q = A逆 \* e / (e转置 \* A逆 \*ｅ)

　　 p转置 = e转置 \* A逆 / (e转置 \* A逆 \*ｅ) e是全为1的列向量

当A不可逆时，每个元素加上一个值就可以了。

当矩阵是m行,n列的时候：

P[1]+P[2]+……+P[m]=1; P[i]>=0

V<=sigma(P[i]\*Matrix[i][j]) 最大化V

## 综合

定理1：最小覆盖数 = 最大匹配数

定理2：最大独立集S 与 最小覆盖集T 互补。

算法：

1. 做最大匹配，没有匹配的空闲点∈S

2. 如果u∈S那么u的临点必然属于T

3. 如果一对匹配的点中有一个属于T那么另外一个属于S

4. 还不能确定的，把左子图的放入S，右子图放入T

算法结束

p是素数且2^p－1的是素数,n不超过258的全部梅森素数终于确定，是

n=2,3,5,7,13,17,31,61,89,107,127,257

有上下界网络流，求可行流部分，增广的流量不是实际流量。若要求实际流量应该强算一遍源点出去的流量。

求最小下届网络流：

方法一：加t-s的无穷大流，求可行流，然后把边反向后（减去下届网络流），在残留网络中从汇到源做最大流。

方法二：在求可行流的时候，不加从汇到源的无穷大边，得到最大流X， 加上从汇到源无穷大边后，再求最大流得到Y。

那么Y即是答案最小下届网络流。

原因：感觉上是在第一遍已经把内部都消耗光了，第二遍是必须的流量。

路径剖分，取节点数最多的子树伸出来的路径。

序列差分表由它的第0行确定，也就是原序列，但同时也可以由第0条对角线上的元素确定。

换句话说，由差分表的第0条对角线就可以确定原序列。有这样两个公式：

原序列为h\_i，第0条对角线为c\_o,c\_1,…,c\_p,0,0,0,…

则h\_n = c\_0\*C(n,0)+c\_1\*C(n,1)+…+c\_p\*C(n,p)，

Σh\_k(k=0..n) = c\_0\*C(n+1,1)+c\_2\*(n+1,2)+…+c\_p\*C(n+1,p+1)。

记住这两个公式，差分表（的第0条对角线）就变得非常有用了。

平面图一定存在一个度小于等于5的点,且可以四染色

（ 欧拉公式 ） 设 G 是连通的平面图，n , m, r 分别是其顶点数、边数和面数，n-m+r=2

极大平面图 m≤3n-6

gcd(2^(a)-1,2^(b)-1)=(2^gcd(a,b))-1.

中国剩余定理：(牛书,P230)

m1,m2......mk两两互素.则下面的同余方程:

x=a1(mod m1)

x=a2(mod m2)

x=a3(mod m3)

.....

在0<=x<=M=m1\*m2\*m3..\*mk内有唯一解.

公式=e1\*a1+e2\*a2+e3\*a3+e4\*a4....就是方程组的一个解.

(附注:x mod 3=a1, x mod 5=a2 , x mod 7=a3.的做法是

x=(5\*7\*a1)+(3\*7\*a2)+(3\*5\*a3)

x= x mod 105.

这个是这个公式的特殊情况,因为ei=大Mi=大M/小mi).

Fibonacci数

gcd(Fn，Fm)=Fgcd(n,m) (牛书，P228)

即是说，两个fibonacci数的最大公约数，肯定是个fibonacci数

Fibonacci质数（和前面所有的Fibonacci数互质）（大多已经是质数了，可能有BUG吧，不确定）

定理：如果a是b的倍数，那么Fa是Fb的倍数。

二次剩余

p为奇素数，若(a,p)=1， a为p的二次剩余必要充分条件为a^((p-1)/2) mod p=1.

(否则为p-1)

p为奇素数，x^b = a(mod p),a为p的b次剩余的必要充分条件为 若a^（(p-1)/ (p-1 和 b的最大公约数)） mod p=1.

平方数的和是平方数的问题。

a[0] := 0;

s := 0;

for i := 1 to n - 2 do

begin

a[i] := a[i - 1] + 1;

s := s + sqr(a[i]);

end;

{======s + sqr(a[n-1]) + sqr(a[n]) = k^2=======}

a[n - 1] := a[n - 2];

repeat

a[n - 1] := a[n - 1] + 1;

until odd(s + sqr(a[n - 1])) and (a[n - 1] > 2);

a[n] := (s + sqr(a[n - 1]) - 1) shr 1;

知道s和a[n-1]后，直接求了a[n].神奇了点。

其实。有当n为奇数：n^2 + ((n^2 - 1) div 2)^2 = ((n^2 + 1) div 2)^2

所以有3 4-- 5 12 -- 7 24 -- 9 40 -- 11 60 ....

a=k\*(s^2 - t^2);

b=2\*k\*s\*t

c=k(s^2 + t^2);

则c^2=a^2+b^2 完全的公式

定义：一颗树T的质心m，就是将m及m连出的边都删除之后，剩下的森林中，每颗树的节点数<=|V(T)|/2。任何树都有质心，并且可以在O(N)的时间内求出。

求的方法如下：以任意一个节点作为T的根，作后序遍历。对于节点v，若是叶子节点，令C(v)=1，否则C(v)=子树和 。遍历过程中，第一次出现C(v)>=|V(T)|/2，那么v就是质心。

质心是个好东西，也许以后对不是二叉树的树进行分治之类的算法，考虑强行把令质心作为根，可以得到二分法一样的时间复杂度。

重加权的方法如下：增加人工结点s，直接到所有点连一条弧，权均为0，然

后以s为起点运行bellman-ford，求出dist(v)。如果有负权圈则退出，否则对于原图中的每个条边(u,v)，设新权w'(u,v)=dist(u)+w(u,v)-dist(v)，则它是非负的

k-连通(k-connected) ：对于任意一对结点都至少存在结点各不相同的k条路。

点连通度(vertex connectivity) ：把图变成非连通图所需删除的最少点

数。

这两个定义是互通的，因为我们有：

Whitney定理：一个图是k-连通的当且仅当它的点连通度至少为k。

Fermat分解算法从t = n^1/2开始，依次检查t2-n; (t+1)2-n; (t+2)2-n … ，直到出现一个平方数y，由于t2-y2 = n，因此分解得n = (t-y)(t+y)。显然，当两个因数很5.1 数论基础243

接近时这个方法能很快找到结果，但如果遇到一个素数，则需要检查(n + 1)/2 - n^1/2个整数，比试除法还慢得多。虽然方法并不是很有效，但是为我们提供了一个思路。

Gessel–Viennot lemma

给定一个图与n个起点n个终点。则从对应的起点至终点的不相交路径条数为det(A)，这里A={a[i][j]}，a[i][j]表示从第i个起点至第j个终点的路径条数。

Stirling公式

n! \approx \sqrt{2\pi n}\, \left(\frac{n}{e}\right)^{n}.

欧拉常数

0.577215,66490,15328,60606,51209,00824,02431,04215,93359,39923,59880,57672,34885

n个球放入m个箱子里，有多少种不同的放法（不一定是球和箱子，也可能是其他的元素与其他的放置位置,例如N个人分到M个单位，每班至少一人，里面已经暗中说明球不同，单位不同）

看似很简单的问题其实非常复杂，球是否相同，箱是否相同？是否允许有空盒

不难看出一共8类情况

1) 球同，盒同，无空箱

2) 球同，盒同，允许空箱

3) 球同，盒不同，无空箱

4) 球同，盒不同，允许空箱

5) 球不同，盒相同，无空箱

6) 球不同，盒相同，允许空箱

7) 球不同，盒不同，无空箱

8) 球不同，盒不同，允许空箱

3的公式是把n个球排成一排，（一种方法），它们中间有n-1个空。取m-1个小棍，放到空上，就把它们分成m部分，由于小棍不相邻，所以没有空箱子。它的方法数有

C(N-1,M-1),也就是球减1里面挑M-1个箱子做组合

4的公式在3的基础上升华出来的，为了避免空箱子，先在每一个箱子假装都放一个球，这样就有n+m个球，C（n+m-1,m-1)，多了M个元素而已

关于1,2类情况，直接f[i][j]计数。

先来分析最特殊的8号：N球不同，M箱不同，允许空。每个球都有M种选择，N个球就有M的N次方分法。

S(n,1)=S(n,n)=1,S（n,k)=S(n-1,k-1)+k\*S(n-1,k)

当遇见类型5即：N不同球，M同箱子，无空箱。一共有S(N,M)种分法。

而类型6，N不同球，M同箱，允许空的时候（在类型5的基础上允许空箱）。明显是N个球不变，一个空箱子都没有+有一个空箱子+有两个空箱子+有三个空箱子+…… 都装在一个箱子。说的简单点一共有就是

S(N,1)+S(N,2)+S(N,3)+..........S(N,M)=也就是说第N排开始第1个数字一直加到第M个数字就是总的分法

而类型7同样是在类型5的基础上升华，因为5是箱同的，而7箱不同，所以箱子自身多了P(M,M)=M!倍可能

所以类型7的公式就是M!乘以S(N,M)

## 多边形内点的计数

//rn中的标号必须逆时针给出。一开始要旋转坐标，保证同一个x值上只有一个点。正向减点，//反向加点。num[i][j]=num[j][i]=严格在这根线下方的点。 on[i][j]=on[j][i]=严格//在线段上的点，包括两个端点。若有回边的话注意计算onit的方法，不要多算了线段上的点。

int ans=0,z,onit=0,lows=0;

rep(z,t) {

i=rn[z]; j=rn[z+1]; onit+=on[i][j]-1;

if (a[j].x>a[i].x){ans-=num[i][j];lows+=on[i][j]-1;}

else ans+=num[i][j];

}//ans-lows+1 is inside. 只会多算一次正向上的点（除去最左和最右的点）。Lows只算了除开最左边的点，但会多算最右边的点，所以要再加上1.

printf("%d\n",ans-lows+1 + onit);

## 基本形公式

**椭圆：**

椭圆，其中离心率焦点参数

椭圆上(x,y)点处的曲率半径为 ,其中分别为(x,y)与两焦点的距离。设点A和点M的坐标分别为(a,0)和(x,y)，则AM的弧长为

椭圆的周长为 ，其中

设椭圆上点M(x,y),N(x,-y),x,y>0,A(a,0),原点O(0,0)。

扇形OAM的面积 弓形MAN的面积

方程，5个点确定一个圆锥曲线。

为(x,y)点关于椭圆中心的极角，r为(x,y)到椭圆中心的距离，椭圆极坐标方程:

**抛物线**

标准方程 曲率半径

弧长：设M(x,y)是抛物线上一点，则]

弓形面积：设M，D是抛物线上两点，且分居一、四象限。作一条平行于MD且与抛物线相切的直线L。若M到L的距离为h。则有

**重心**

半径为r、圆心角为的扇形的重心与圆心的距离为

半径为r、圆心角为的圆弧的重心与圆心的距离为

椭圆上半部分的重心与圆心的距离为

抛物线中弓形MOD的重心满足 , P是直线L与抛物线的切点，Q在MD上且PQ平行x轴。C是重心。

**内心**

**三重积公式**

**额外的公式**

***四边形***: D1,D2为对角线,M对角线中点连线,A为对角线夹角

1.a^2+b^2+c^2+d^2=D1^2+D2^2+4M^2 2. S=D1D2sin(A)/2

(以下对圆的内接四边形)

3. ac+bd=D1D2 4.S=sqrt((P-a)(P-b)(P-c)(P-d)),P为半周长

***正n边形:***R为外接圆半径,r为内切圆半径

1. 中心角 A=2PI/n 2. 内角C=(n-2)PI/n

3. 边长 a=2sqrt(R^2-r^2)=2Rsin(A/2)=2rtan(A/2)

4. 面积S=nar/2=nr^2tan(A/2)=nR^2sin(A)/2=na^2/(4tan(A/2))

***圆:*** 1. 弧长 l=rA 2. 弦长 a=2sqrt(2hr-h^2)=2rsin(A/2)

3. 弓形高h=r-sqrt(r^2-a^2/4)=r(1-cos(A/2))=atan(A/4)/2

4.扇形面积S1=rl/2=r^2A/2

5.弓形面积 S2=(rl-a(r-h))/2=r^2(A-sin(A))/2

***棱柱:*** 1. 体积 V=Ah,A为底面积,h为高

2. 侧面积S=lp,l为棱长,p为直截面周长 3. 全面积 T=S+2A

***棱锥:*** 1.体积 V=Ah/3,A为底面积,h为高 (以下对正棱锥)

2. 侧面积S=lp/2,l为斜高,p为底面周长 3. 全面积 T=S+A

***棱台:***1. 体积 V=(A1+A2+sqrt(A1A2))h/3,A1.A2为上下底面积,h为高(以下为正棱台)

2. 侧面积 S=(p1+p2)l/2,p1.p2为上下底面周长,l为斜高

3. 全面积 T=S+A1+A2

## 树的计数

**有根树的计数**

令

于是，n+1个结点的有根树的总数为

附：

**无根树的计数**

当n是奇数时，则有 种不同的无根树。

当n是偶数时，则有这么多种不同的无根树。

**生成树的计数**

完全图的生成树个数

任意图的生成树个数： 生成树计数行列式tab[i][i] = Di，Di为i的度数tab[i][j] = −k, k为i和j之间的边数。任去一行一列之后的行列式。

## 代数

**Burnside引理**

**三次方程求根公式**

其中 j=0,1,2,

当求解时， 令 再求解y，即转化成的形式

**组合公式**

错排：

## 三角公式

## 积分表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | | | | | |  |
|  | |  | | | | | |  |
|  | |  | | | | | |  |
|  | |  | | | | | |  |
|  | |  | | | | | |  |
|  | | | | | |  | | |
|  | | | | | |  | | |
|  | | | | | |  | | |
|  | | | | | |  | | |
|  | | | | | |  | | |
|  | | | | | |  | | |
|  | | | | | | | | |
|  | | | |  | | | | |
|  | | | |  | | | | |
|  | | | |  | | | | |
|  | | | | | | | | |
|  | | | |  | | | | |
|  | | | |  | | | | |
|  | | | |  | | | | |
|  | | | | | | | | |
|  | | | | |  | | | |
|  | | | | | | | | |
|  | | | | | | | | |
|  | | | | | | | |  |
|  | | | | | | | |
|  |  | | | | | | |  |
|  |  | | | | | | |  |
|  | | |  | | | | | |
|  | |  | | | | |  | |
|  | | |  | | | | | |
|  | | |  | | | | | |
|  | | |  | | | | | |

## Java IO&vimrc

import java.io.\*;

import java.util.\*;

import java.math.\*;

public class Main {

void run() throws Exception {

reader.close();

writer.close();

}

public static void main(String[] args) throws Exception {

(new Main()).run();

}

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

PrintWriter writer = new PrintWriter(System.out);

StringTokenizer tokenizer = null;

String next() throws Exception {

for (; tokenizer == null || !tokenizer.hasMoreTokens(); ) {

tokenizer = new StringTokenizer(reader.readLine());

}

return tokenizer.nextToken();

}

int nextInt() throws Exception {

return Integer.parseInt(next());

}

}

syntax on

set cindent

set number

set nobackup

set expandtab

set softtabstop=4

set shiftwidth=4

set tabstop=4

set guifont=Courier\_New

set cinoptions=:0,g0

nmap <F2> :vs %:r.in <CR>

autocmd filetype cpp nmap <F5> :!%:r <%:r.in <CR>

autocmd filetype cpp nmap <F9> :!make %:r <CR>

autocmd filetype java nmap <F5> :!java %:r <%:r.in <CR>

autocmd filetype java nmap <F9> :!javac %:r.java <CR>