ACM Template

135_Morisummer

Last build at July 8, 2014

Contents

1 Geometry

1.1 注意

I. 注意舍入方式 (0.5 的舍入方向); 防止输出 -0.

II. 几何题注意多测试不对称数据.

III. 整数几何注意 xmult 和 dmult 是否会出界;
符点几何注意 eps 的使用.

IV. 避免使用斜率; 注意除数是否会为 0.

V. 公式一定要化简后再代入.

VI. 判断同一个 $2 \times PI$ 域内两角度差应该是
$abs(a1-a2) < beta \parallel abs(a1-a2) > \pi + \pi - beta$;
相等应该是
$abs(a1-a2) < eps \parallel abs(a1-a2) > \pi + \pi - eps$.

VII. 需要的话尽量使用 $atan2$, 注意:$atan2(0,0)=0$,
$atan2(1,0)=\pi/2$,$atan2(-1,0)=-\pi/2$,$atan2(0,1)=0$,$atan2(0,-1)=\pi$.

VIII. cross product $= \mid u \mid \times \mid v \mid \times sin(a)$
dot product $= \mid u \mid \times \mid v \mid \times cos(a)$

IX. $(P1-P0)X(P2-P0)$ 结果的意义:
正: $<P0,P1>$ 在 $<P0,P2>$ 顺时针 $(0,\pi)$ 内
负: $<P0,P1>$ 在 $<P0,P2>$ 逆时针 $(0,\pi)$ 内
$0: <P0,P1>$,$<P0,P2>$ 共线, 夹角为 0 或 π

X. 误差限缺省使用 $1e-8$!

1.2 几何公式

1.2.1 三角形

I. 半周长 $P = \frac{a+b+c}{2}$

II. 面积 $S = \frac{a \times H}{2} = \frac{a \times b \times sin(C)}{2} = \sqrt{P \times (P-a) \times (P-b) \times (P-c)}$

III. 中线 $Ma = \frac{\sqrt{2 \times (b^2+c^2)-a^2}}{2} = \frac{\sqrt{b^2+c^2+2 \times b \times c \times cos(A)}}{2}$

IV. 角平分线 $Ta = \frac{\sqrt{b \times c((b+c)^2-a^2)}}{b+c} = \frac{2 \times b \times c \times cos(\frac{A}{2})}{b+c}$

V. 高线 $Ha = b \times sin(C) = c \times sin(B) = \sqrt{b^2 - (\frac{a^2+b^2-c^2}{2 \times a})^2}$

VI. 内切圆半径 $r = \frac{S}{P} = \frac{a \times sin(\frac{B}{2}) \times sin(\frac{C}{2})}{sin(\frac{B+C}{2})}$
$= 4 \times R \times sin(\frac{A}{2}) \times sin(\frac{B}{2}) \times sin(\frac{C}{2}) = \sqrt{\frac{(P-a) \times (P-b) \times (P-c)}{P}}$
$= P \times tan(\frac{A}{2}) \times tan(\frac{B}{2}) \times tan(\frac{C}{2})$

VII. 外接圆半径 $R = \frac{a \times b \times c}{4 \times S} = \frac{a}{2 \times sin(A)} = \frac{b}{2 \times sin(B)} = \frac{c}{2 \times sin(C)}$

1.2.2 四边形

$D1$,$D2$ 为对角线,M 对角线中点连线,A 为对角线夹角

I. $a^2+b^2+c^2+d^2 = D1^2 + D2^2 + 4 \times M^2$

II. $S = \frac{D1 \times D2 \times sin(A)}{2}$

1.2.3 圆内接四边形

I. $a \times c + b \times d = D1 \times D2$

II. $S = \sqrt{(P-a) \times (P-b) \times (P-c) \times (P-d)}$,$P$ 为半周长

1.2.4 正 N 边形

R 为外接圆半径,r 为内切圆半径

1. 中心角 $A = \frac{2\times\pi}{N}$
2. 内角 $C = \frac{(N-2)\times\pi}{N}$
3. 边长 $a = 2 \times \sqrt{R^2 - r^2} = 2 \times R \times sin(\frac{A}{2}) = 2 \times r \times tan(\frac{A}{2})$
4. 面积 $S = \frac{N\times a\times r}{2} = N \times r^2 \times tan(\frac{A}{2}) = \frac{N\times R^2\times sin(A)}{2} = \frac{N\times a^2}{4\times tan(\frac{A}{2})}$

1.2.5 圆

I. 弧长 $l = rA$

II. 弦长 $a = 2 \times \sqrt{2 \times h \times r - h^2} = 2 \times r \times sin(\frac{A}{2})$

III. 弓形高 $h = r - \sqrt{r^2 - \frac{a^2}{4}} = r \times (1 - cos(\frac{A}{2})) = \frac{a\times tan(\frac{A}{4})}{2}$

IV. 扇形面积 $S1 = \frac{r\times l}{2} = \frac{r^2\times A}{2}$

V. 弓形面积 $S2 = \frac{r\times l - a\times(r-h)}{2} = \frac{r^2\times(A - sin(A))}{2}$

1.2.6 棱柱

I. 体积 $V = A \times h$ A 为底面积,h 为高

II. 侧面积 $S = l \times p$ l 为棱长,p 为直截面周长

III. 全面积 $T = S + 2 \times A$

1.2.7 棱锥

I. 体积 $V = \frac{A\times h}{3}$ A 为底面积,h 为高

1.2.8 正棱锥

I. 侧面积 $S = \frac{l\times p}{2}$ l 为斜高,p 为底面周长

II. 全面积 $T = S + A$

1.2.9 棱台

I. 体积 $V = \frac{(A1 + A2 + \sqrt{A1\times A2})\times h}{3}$ $A1$,$A2$ 为上下底面积,h 为高

1.2.10 正棱台

I. 侧面积 $S = \frac{(p1+p2)\times l}{2}$ $p1$,$p2$ 为上下底面周长,l 为斜高

II. 全面积 $T = S + A1 + A2$

1.2.11 圆柱

I. 侧面积 $S = 2 \times \pi \times r \times h$

II. 全面积 $T = 2 \times \pi \times r \times (h + r)$

III. 体积 $V = \pi \times r^2 \times h$

1.2.12 圆锥

I. 母线 $l = \sqrt{h^2 + r^2}$

II. 侧面积 $S = \pi \times r \times l$

III. 全面积 $T = \pi \times r \times (l + r)$

IV. 体积 $V = \frac{\pi \times r^2 \times h}{3}$

1.2.13 圆台

I. 母线 $l = \sqrt{h^2 + (r1 - r2)^2}$

II. 侧面积 $S = \pi \times (r1 + r2) \times l$

III. 全面积 $T = \pi \times r1 \times (l + r1) + \pi \times r2 \times (l + r2)$

IV. 体积 $V = \frac{\pi \times (r1^2 + r2^2 + r1 \times r2) \times h}{3}$

1.2.14 球

I. 全面积 $T = 4 \times \pi \times r^2$

II. 体积 $V = \frac{4 \times \pi \times r^3}{3}$

1.2.15 球台

I. 侧面积 $S = 2 \times \pi \times r \times h$

II. 全面积 $T = \pi \times (2 \times r \times h + r1^2 + r2^2)$

III. 体积 $V = \frac{\pi \times h \times (3 \times (r1^2 + r2^2) + h^2)}{6}$

1.2.16 球扇形

I. 全面积 $T = \pi \times r \times (2 \times h + r0)$ h 为球冠高,$r0$ 为球冠底面半径

II. 体积 $V = \frac{2 \times \pi \times r^2 \times h}{3}$

1.3 多边形

1.3.1 头文件

```
#define MAXN 1000
#define offset 10000
#define eps 1e-8
#define zero(x) (((x)>0?(x):-(x))<eps)
#define _sign(x) ((x)>eps?1:((x)<-eps?2:0))
struct point{double x,y;};
struct line{point a,b;};
double xmult(point p1,point p2,point p0)
{
  return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
```

1.3.2 判定凸多边形, 允许相邻边共线

```
int is_convex(int n,point* p)
{
  int i,s[3]={1,1,1};
  for (i=0;i<n&&s[1]|s[2];i++)
    s[_sign(xmult(p[(i+1)%n],p[(i+2)%n],p[i]))]=0;
  return s[1]|s[2];
}
```

1.3.3 判定凸多边形, 不允许相邻边共线

```
int is_convex_v2(int n,point* p)
{
  int i,s[3]={1,1,1};
  for (i=0;i<n&&s[0]&&s[1]|s[2];i++)
    s[_sign(xmult(p[(i+1)%n],p[(i+2)%n],p[i]))]=0;
  return s[0]&&s[1]|s[2];
}
```

1.3.4 判点在凸多边形内或多边形边上

```
int inside_convex(point q,int n,point* p)
{
  int i,s[3]={1,1,1};
  for (i=0;i<n&&s[1]|s[2];i++)
    s[_sign(xmult(p[(i+1)%n],q,p[i]))]=0;
  return s[1]|s[2];
}
```

1.3.5 判点在凸多边形内

```
int inside_convex_v2(point q,int n,point* p)
{
  int i,s[3]={1,1,1};
  for (i=0;i<n&&s[0]&&s[1]|s[2];i++)
    s[_sign(xmult(p[(i+1)%n],q,p[i]))]=0;
  return s[0]&&s[1]|s[2];
}
```

1.3.6 判点在任意多边形内

```
//表示点在多边形边上时的返回值on_edge,为多边形坐标上限offset
int inside_polygon(point q,int n,point* p,int on_edge=1)
{
  point q2;
  int i=0,count;
  while (i<n)
    for (count=i=0,q2.x=rand()+offset,q2.y=rand()+offset;i<n;i++)
      if (zero(xmult(q,p[i],p[(i+1)%n]))&&(p[i].x-q.x)*(p[(i+1)%n].x-q.x)<eps&&(p[i].y-q.y)*(p[(i+1)%n].y-q.y)<eps)
        return on_edge;
      else if (zero(xmult(q,q2,p[i])))
        break;
      else if (xmult(q,p[i],q2)*xmult(q,p[(i+1)%n],q2)<-eps&&xmult(p[i],q,p[(i+1)%n])*xmult(p[i],q2,p[(i+1)%n])<-eps)

        count++;
  return count&1;
}
```

1.3.7 判线段在任意多边形内

```
inline int opposite_side(point p1,point p2,point l1,point l2)
{
  return xmult(l1,p1,l2)*xmult(l1,p2,l2)<-eps;
}
inline int dot_online_in(point p,point l1,point l2)
{
  return zero(xmult(p,l1,l2))&&(l1.x-p.x)*(l2.x-p.x)<eps&&(l1.y-p.y)*(l2.y-p.y)<eps;
}
//判线段在任意多边形内顶点按顺时针或逆时针给出与边界相交返回,,1
int inside_polygon_v2(point l1,point l2,int n,point* p)
{
  point t[MAXN],tt;
  int i,j,k=0;
  if (!inside_polygon(l1,n,p)||!inside_polygon(l2,n,p))
```

```
    return 0;
  for (i=0;i<n;i++)
    if (opposite_side(l1,l2,p[i],p[(i+1)%n])&&opposite_side(p[i],p[(i+1)%n],l1,l2))
      return 0;
    else if (dot_online_in(l1,p[i],p[(i+1)%n]))
      t[k++]=l1;
    else if (dot_online_in(l2,p[i],p[(i+1)%n]))
      t[k++]=l2;
    else if (dot_online_in(p[i],l1,l2))
      t[k++]=p[i];
  for (i=0;i<k;i++)
    for (j=i+1;j<k;j++){
      tt.x=(t[i].x+t[j].x)/2;
      tt.y=(t[i].y+t[j].y)/2;
      if (!inside_polygon(tt,n,p))
        return 0;
    }
  return 1;
}
```

1.3.8 多边形重心

```
point intersection(line u,line v)
{
  point 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.x+=(u.b.x-u.a.x)*t;
  ret.y+=(u.b.y-u.a.y)*t;
  return ret;
}

point barycenter(point a,point b,point c)
{
  line u,v;
  u.a.x=(a.x+b.x)/2;
  u.a.y=(a.y+b.y)/2;
  u.b=c;
  v.a.x=(a.x+c.x)/2;
  v.a.y=(a.y+c.y)/2;
  v.b=b;
  return intersection(u,v);
}

//多边形重心
point barycenter(int n,point* p)
{
  point ret,t;
  double t1=0,t2;
  int i;
  ret.x=ret.y=0;
  for (i=1;i<n-1;i++)
    if (fabs(t2=xmult(p[0],p[i],p[i+1]))>eps){
      t=barycenter(p[0],p[i],p[i+1]);
      ret.x+=t.x*t2;
      ret.y+=t.y*t2;
      t1+=t2;
    }
  if (fabs(t1)>eps)
    ret.x/=t1,ret.y/=t1;
  return ret;
}
```

1.4 浮点函数

1.4.1 头文件

```
#include <math.h>
#define eps 1e-8
#define zero(x) (((x)>0?(x):-(x))<eps)
struct point
{
  double x,y;
  point(double a=0.0,b=0.0)
  {
    x=a,y=b;
  }
};
struct line
```

```
{
  point a,b;
  line(point s=point(),point e=point())
  {
    a=s,b=e;
  }
};
double xmult(point p1,point p2,point p0)
{
  return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
double xmult(double x1,double y1,double x2,double y2,double x0,double y0)
{
  return (x1-x0)*(y2-y0)-(x2-x0)*(y1-y0);
}
double dmult(point p1,point p2,point p0)
{
  return (p1.x-p0.x)*(p2.x-p0.x)+(p1.y-p0.y)*(p2.y-p0.y);
}
double dmult(double x1,double y1,double x2,double y2,double x0,double y0)
{
  return (x1-x0)*(x2-x0)+(y1-y0)*(y2-y0);
}
```

1.4.2 两点距离

```
double dis(point p1,point p2)
{
  return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}
double dis(double x1,double y1,double x2,double y2)
{
  return sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
}
```

1.4.3 判三点共线

```
int dots_inline(point p1,point p2,point p3)
{
  return zero(xmult(p1,p2,p3));
}
int dots_inline(double x1,double y1,double x2,double y2,double x3,double y3)
{
  return zero(xmult(x1,y1,x2,y2,x3,y3));
}
```

;

1.4.4 判点在线段上，包括端点

```
int dot_online_in(point p,line l)
{
  return zero(xmult(p,l.a,l.b))&&(l.a.x-p.x)*(l.b.x-p.x)<eps&&(l.a.y-p.y)*(l.b.y-p.y)<eps;
}
int dot_online_in(point p,point l1,point l2)
{
  return zero(xmult(p,l1,l2))&&(l1.x-p.x)*(l2.x-p.x)<eps&&(l1.y-p.y)*(l2.y-p.y)<eps;
}
int dot_online_in(double x,double y,double x1,double y1,double x2,double y2)
{
  return zero(xmult(x,y,x1,y1,x2,y2))&&(x1-x)*(x2-x)<eps&&(y1-y)*(y2-y)<eps;
}
```

1.4.5 判点在线段上, 不包括端点

```
int dot_online_ex(point p,line l)
{
  return dot_online_in(p,l)&&(!zero(p.x-l.a.x)||!zero(p.y-l.a.y))&&(!zero(p.x-l.b.x)||!zero(p.y-l.b.y));
}
int dot_online_ex(point p,point l1,point l2)
{
  return dot_online_in(p,l1,l2)&&(!zero(p.x-l1.x)||!zero(p.y-l1.y))&&(!zero(p.x-l2.x)||!zero(p.y-l2.y));
}
int dot_online_ex(double x,double y,double x1,double y1,double x2,double y2)
{
  return dot_online_in(x,y,x1,y1,x2,y2)&&(!zero(x-x1)||!zero(y-y1))&&(!zero(x-x2)||!zero(y-y2));
}
```

1.4.6 判两点在线段同侧, 点在线段上返回 0

```
int same_side(point p1,point p2,line l)
{
  return xmult(l.a,p1,l.b)*xmult(l.a,p2,l.b)>eps;
}
int same_side(point p1,point p2,point l1,point l2)
{
  return xmult(l1,p1,l2)*xmult(l1,p2,l2)>eps;
}
```

1.4.7 判两点在线段异侧, 点在线段上返回 0

```
int opposite_side(point p1,point p2,line l)
{
  return xmult(l.a,p1,l.b)*xmult(l.a,p2,l.b)<-eps;
}
int opposite_side(point p1,point p2,point l1,point l2)
{
  return xmult(l1,p1,l2)*xmult(l1,p2,l2)<-eps;
}
```

1.4.8 判两直线平行

```
int parallel(line u,line v)
{
  return zero((u.a.x-u.b.x)*(v.a.y-v.b.y)-(v.a.x-v.b.x)*(u.a.y-u.b.y));
}
int parallel(point u1,point u2,point v1,point v2)
{
  return zero((u1.x-u2.x)*(v1.y-v2.y)-(v1.x-v2.x)*(u1.y-u2.y));
}
```

1.4.9 判两直线垂直

```
int perpendicular(line u,line v)
{
  return zero((u.a.x-u.b.x)*(v.a.x-v.b.x)+(u.a.y-u.b.y)*(v.a.y-v.b.y));
}
int perpendicular(point u1,point u2,point v1,point v2)
{
  return zero((u1.x-u2.x)*(v1.x-v2.x)+(u1.y-u2.y)*(v1.y-v2.y));
}
```

1.4.10 判两线段相交, 包括端点和部分重合

```
int intersect_in(line u,line v)
{
  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);
}
int intersect_in(point u1,point u2,point v1,point v2)
{
  if (!dots_inline(u1,u2,v1)||!dots_inline(u1,u2,v2))
    return !same_side(u1,u2,v1,v2)&&!same_side(v1,v2,u1,u2);
  return dot_online_in(u1,v1,v2)||dot_online_in(u2,v1,v2)||dot_online_in(v1,u1,u2)||dot_online_in(v2,u1,u2
      );
}
```

1.4.11 判两线段相交, 不包括端点和部分重合

```
int intersect_ex(line u,line v)
{
  return opposite_side(u.a,u.b,v)&&opposite_side(v.a,v.b,u);
}
int intersect_ex(point u1,point u2,point v1,point v2)
{
  return opposite_side(u1,u2,v1,v2)&&opposite_side(v1,v2,u1,u2);
}
```

1.4.12 计算两直线交点

```
point intersection(line u,line v)
{
  point 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.x+=(u.b.x-u.a.x)*t;
  ret.y+=(u.b.y-u.a.y)*t;
  return ret;
}
point intersection(point u1,point u2,point v1,point v2)
{
  point ret=u1;
  double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
      /((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
  ret.x+=(u2.x-u1.x)*t;
  ret.y+=(u2.y-u1.y)*t;
  return ret;
}
```

1.4.13 点到直线上的最近点

```
point ptoline(point p,line l)
{
  point t=p;
  t.x+=l.a.y-l.b.y,t.y+=l.b.x-l.a.x;
  return intersection(p,t,l.a,l.b);
}
point ptoline(point p,point l1,point l2)
{
  point t=p;
  t.x+=l1.y-l2.y,t.y+=l2.x-l1.x;
  return intersection(p,t,l1,l2);
}
```

1.4.14 点到直线距离

```
double disptoline(point p,line l)
{
  return fabs(xmult(p,l.a,l.b))/distance(l.a,l.b);
}
double disptoline(point p,point l1,point l2)
{
  return fabs(xmult(p,l1,l2))/distance(l1,l2);
}
double disptoline(double x,double y,double x1,double y1,double x2,double y2)
{
  return fabs(xmult(x,y,x1,y1,x2,y2))/distance(x1,y1,x2,y2);
}
```

1.4.15 点到线段上的最近点

```
point ptoseg(point p,line l)
{
  point t=p;
  t.x+=l.a.y-l.b.y,t.y+=l.b.x-l.a.x;
  if (xmult(l.a,t,p)*xmult(l.b,t,p)>eps)
    return distance(p,l.a)<distance(p,l.b)?l.a:l.b;
  return intersection(p,t,l.a,l.b);
}
point ptoseg(point p,point l1,point l2)
{
  point t=p;
  t.x+=l1.y-l2.y,t.y+=l2.x-l1.x;
  if (xmult(l1,t,p)*xmult(l2,t,p)>eps)
    return distance(p,l1)<distance(p,l2)?l1:l2;
  return intersection(p,t,l1,l2);
}
```

1.4.16 点到线段距离

```
double disptoseg(point p,line l)
{
  point t=p;
  t.x+=l.a.y-l.b.y,t.y+=l.b.x-l.a.x;
  if (xmult(l.a,t,p)*xmult(l.b,t,p)>eps)
    return distance(p,l.a)<distance(p,l.b)?distance(p,l.a):distance(p,l.b);
  return fabs(xmult(p,l.a,l.b))/distance(l.a,l.b);
}
double disptoseg(point p,point l1,point l2)
{
  point t=p;
  t.x+=l1.y-l2.y,t.y+=l2.x-l1.x;
  if (xmult(l1,t,p)*xmult(l2,t,p)>eps)
    return distance(p,l1)<distance(p,l2)?distance(p,l1):distance(p,l2);
  return fabs(xmult(p,l1,l2))/distance(l1,l2);
}
```

1.4.17 矢量 V 以 P 为顶点逆时针旋转 angle 并放大 scale 倍

```
point rotate(point v,point p,double angle,double scale)
{
  point ret=p;
  v.x-=p.x,v.y-=p.y;
  p.x=scale*cos(angle);
  p.y=scale*sin(angle);
  ret.x+=v.x*p.x-v.y*p.y;
  ret.y+=v.x*p.y+v.y*p.x;
  return ret;
}
```

1.5 三角形

1.5.1 头文件

```
#include <math.h>
struct point{double x,y;};
struct line{point a,b;};

double distance(point p1,point p2)
{
  return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}

point intersection(line u,line v)
{
  point 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.x+=(u.b.x-u.a.x)*t;
  ret.y+=(u.b.y-u.a.y)*t;
  return ret;
}
```

1.5.2 外心

```
point circumcenter(point a,point b,point c)
{
  line u,v;
  u.a.x=(a.x+b.x)/2;
  u.a.y=(a.y+b.y)/2;
  u.b.x=u.a.x-a.y+b.y;
  u.b.y=u.a.y+a.x-b.x;
  v.a.x=(a.x+c.x)/2;
  v.a.y=(a.y+c.y)/2;
  v.b.x=v.a.x-a.y+c.y;
  v.b.y=v.a.y+a.x-c.x;
  return intersection(u,v);
}
```

1.5.3 内心

```
point incenter(point a,point b,point c)
{
  line u,v;
  double m,n;
  u.a=a;
  m=atan2(b.y-a.y,b.x-a.x);
  n=atan2(c.y-a.y,c.x-a.x);
  u.b.x=u.a.x+cos((m+n)/2);
  u.b.y=u.a.y+sin((m+n)/2);
  v.a=b;
  m=atan2(a.y-b.y,a.x-b.x);
  n=atan2(c.y-b.y,c.x-b.x);
  v.b.x=v.a.x+cos((m+n)/2);
  v.b.y=v.a.y+sin((m+n)/2);
  return intersection(u,v);
}
```

1.5.4 垂心

```
point perpencenter(point a,point b,point c)
{
  line u,v;
  u.a=c;
  u.b.x=u.a.x-a.y+b.y;
  u.b.y=u.a.y+a.x-b.x;
  v.a=b;
  v.b.x=v.a.x-a.y+c.y;
  v.b.y=v.a.y+a.x-c.x;
  return intersection(u,v);
}
```

1.5.5 重心

```
point barycenter(point a,point b,point c)
{
  line u,v;
  u.a.x=(a.x+b.x)/2;
  u.a.y=(a.y+b.y)/2;
  u.b=c;
  v.a.x=(a.x+c.x)/2;
  v.a.y=(a.y+c.y)/2;
  v.b=b;
  return intersection(u,v);
}
```

1.5.6 费马点

```
point fermentpoint(point a,point b,point c)
{
  point u,v;
  double step=fabs(a.x)+fabs(a.y)+fabs(b.x)+fabs(b.y)+fabs(c.x)+fabs(c.y);
  int i,j,k;
  u.x=(a.x+b.x+c.x)/3;
  u.y=(a.y+b.y+c.y)/3;
  while (step>1e-10)
    for (k=0;k<10;step/=2,k++)
      for (i=-1;i<=1;i++)
        for (j=-1;j<=1;j++)
        {
          v.x=u.x+step*i;
          v.y=u.y+step*j;
          if (distance(u,a)+distance(u,b)+distance(u,c)>distance(v,a)+distance(v,b)+distance(v,c))
            u=v;
        }
  return u;
}
```

1.6 三维几何

1.6.1 头文件

```
#include <math.h>
#define eps 1e-8
#define zero(x) (((x)>0?(x):-(x))<eps)
struct point3{double x,y,z;};
struct line3{point3 a,b;};
struct plane3{point3 a,b,c;};

point3 xmult(point3 u,point3 v)
{
  point3 ret;
  ret.x=u.y*v.z-v.y*u.z;
  ret.y=u.z*v.x-u.x*v.z;
  ret.z=u.x*v.y-u.y*v.x;
  return ret;
}

double dmult(point3 u,point3 v)
{
  return u.x*v.x+u.y*v.y+u.z*v.z;
}

point3 subt(point3 u,point3 v)
{
  point3 ret;
  ret.x=u.x-v.x;
  ret.y=u.y-v.y;
  ret.z=u.z-v.z;
  return ret;
}

double dist3(point3 p1,point3 p2)
{
  return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y)+(p1.z-p2.z)*(p1.z-p2.z));
}

double vlen(point3 p)
{
  return sqrt(p.x*p.x+p.y*p.y+p.z*p.z);
}
```

1.6.2 取平面法向量

```
point3 pvec(plane3 s)
{
  return xmult(subt(s.a,s.b),subt(s.b,s.c));
}
point3 pvec(point3 s1,point3 s2,point3 s3)
{
  return xmult(subt(s1,s2),subt(s2,s3));
}
```

1.6.3 判三点共线

```
int dots_inline(point3 p1,point3 p2,point3 p3)
{
  return vlen(xmult(subt(p1,p2),subt(p2,p3)))<eps;
}
```

1.6.4 判四点共面

```
int dots_onplane(point3 a,point3 b,point3 c,point3 d)
{
  return zero(dmult(pvec(a,b,c),subt(d,a)));
}
```

1.6.5 判点是否在线段上, 包括端点和共线

```
int dot_online_in(point3 p,line3 l)
{
  return zero(vlen(xmult(subt(p,l.a),subt(p,l.b))))&&(l.a.x-p.x)*(l.b.x-p.x)<eps&&
    (l.a.y-p.y)*(l.b.y-p.y)<eps&&(l.a.z-p.z)*(l.b.z-p.z)<eps;
}
int dot_online_in(point3 p,point3 l1,point3 l2)
{
  return zero(vlen(xmult(subt(p,l1),subt(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;
}
```

1.6.6 判点是否在线段上, 不包括端点

```
int dot_online_ex(point3 p,line3 l)
{
  return dot_online_in(p,l)&&(!zero(p.x-l.a.x)||!zero(p.y-l.a.y)||!zero(p.z-l.a.z))&&
    (!zero(p.x-l.b.x)||!zero(p.y-l.b.y)||!zero(p.z-l.b.z));
}
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));
}
```

1.6.7 判点是否在空间三角形上, 包括边界, 三点共线无意义

```
int dot_inplane_in(point3 p,plane3 s)
{
  return zero(vlen(xmult(subt(s.a,s.b),subt(s.a,s.c)))-vlen(xmult(subt(p,s.a),subt(p,s.b)))-
    vlen(xmult(subt(p,s.b),subt(p,s.c)))-vlen(xmult(subt(p,s.c),subt(p,s.a))));
}
int dot_inplane_in(point3 p,point3 s1,point3 s2,point3 s3)
{
  return zero(vlen(xmult(subt(s1,s2),subt(s1,s3)))-vlen(xmult(subt(p,s1),subt(p,s2)))-
    vlen(xmult(subt(p,s2),subt(p,s3)))-vlen(xmult(subt(p,s3),subt(p,s1))));
}
```

1.6.8 判点是否在空间三角形上, 不包括边界, 三点共线无意义

```
int dot_inplane_ex(point3 p,plane3 s)
{
  return dot_inplane_in(p,s)&&vlen(xmult(subt(p,s.a),subt(p,s.b)))>eps&&
    vlen(xmult(subt(p,s.b),subt(p,s.c)))>eps&&vlen(xmult(subt(p,s.c),subt(p,s.a)))>eps;
}
int dot_inplane_ex(point3 p,point3 s1,point3 s2,point3 s3)
{
  return dot_inplane_in(p,s1,s2,s3)&&vlen(xmult(subt(p,s1),subt(p,s2)))>eps&&
    vlen(xmult(subt(p,s2),subt(p,s3)))>eps&&vlen(xmult(subt(p,s3),subt(p,s1)))>eps;
}
```

1.6.9 判两点在线段同侧, 点在线段上返回 0, 不共面无意义

```
int same_side(point3 p1,point3 p2,line3 l)
{
  return dmult(xmult(subt(l.a,l.b),subt(p1,l.b)),xmult(subt(l.a,l.b),subt(p2,l.b)))>eps;
}
int same_side(point3 p1,point3 p2,point3 l1,point3 l2)
{
  return dmult(xmult(subt(l1,l2),subt(p1,l2)),xmult(subt(l1,l2),subt(p2,l2)))>eps;
}
```

1.6.10 判两点在线段异侧, 点在线段上返回 0, 不共面无意义

```
int opposite_side(point3 p1,point3 p2,line3 l)
{
  return dmult(xmult(subt(l.a,l.b),subt(p1,l.b)),xmult(subt(l.a,l.b),subt(p2,l.b)))<-eps;
}
int opposite_side(point3 p1,point3 p2,point3 l1,point3 l2)
{
  return dmult(xmult(subt(l1,l2),subt(p1,l2)),xmult(subt(l1,l2),subt(p2,l2)))<-eps;
}
```

1.6.11 判两点在平面同侧, 点在平面上返回 0

```
int same_side(point3 p1,point3 p2,plane3 s)
{
  return dmult(pvec(s),subt(p1,s.a))*dmult(pvec(s),subt(p2,s.a))>eps;
}
int same_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3)
{
  return dmult(pvec(s1,s2,s3),subt(p1,s1))*dmult(pvec(s1,s2,s3),subt(p2,s1))>eps;
}
```

1.6.12 判两点在平面异侧, 点在平面上返回 0

```
int opposite_side(point3 p1,point3 p2,plane3 s)
{
  return dmult(pvec(s),subt(p1,s.a))*dmult(pvec(s),subt(p2,s.a))<-eps;
}
int opposite_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3)
{
  return dmult(pvec(s1,s2,s3),subt(p1,s1))*dmult(pvec(s1,s2,s3),subt(p2,s1))<-eps;
}
```

1.6.13 判两直线平行

```
int parallel(line3 u,line3 v)
{
  return vlen(xmult(subt(u.a,u.b),subt(v.a,v.b)))<eps;
}
int parallel(point3 u1,point3 u2,point3 v1,point3 v2)
{
  return vlen(xmult(subt(u1,u2),subt(v1,v2)))<eps;
}
```

1.6.14 判两平面平行

```
int parallel(plane3 u,plane3 v)
{
  return vlen(xmult(pvec(u),pvec(v)))<eps;
}
int parallel(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)
{
  return vlen(xmult(pvec(u1,u2,u3),pvec(v1,v2,v3)))<eps;
}
```

1.6.15 判直线与平面平行

```
int parallel(line3 l,plane3 s)
{
  return zero(dmult(subt(l.a,l.b),pvec(s)));
}
int parallel(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
{
  return zero(dmult(subt(l1,l2),pvec(s1,s2,s3)));
}
```

1.6.16 判两直线垂直

```
int perpendicular(line3 u,line3 v)
{
  return zero(dmult(subt(u.a,u.b),subt(v.a,v.b)));
}
int perpendicular(point3 u1,point3 u2,point3 v1,point3 v2)
{
  return zero(dmult(subt(u1,u2),subt(v1,v2)));
}
```

1.6.17 判两平面垂直

```
int perpendicular(plane3 u,plane3 v)
{
  return zero(dmult(pvec(u),pvec(v)));
}
int perpendicular(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)
{
  return zero(dmult(pvec(u1,u2,u3),pvec(v1,v2,v3)));
}
```

1.6.18 判直线与平面垂直

```
int perpendicular(line3 l,plane3 s)
{
  return vlen(xmult(subt(l.a,l.b),pvec(s)))<eps;
}
int perpendicular(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
{
  return vlen(xmult(subt(l1,l2),pvec(s1,s2,s3)))<eps;
}
```

1.6.19 判两线段相交, 包括端点和部分重合

```
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);
}
int intersect_in(point3 u1,point3 u2,point3 v1,point3 v2)
{
  if (!dots_onplane(u1,u2,v1,v2))
    return 0;
  if (!dots_inline(u1,u2,v1)||!dots_inline(u1,u2,v2))
    return !same_side(u1,u2,v1,v2)&&!same_side(v1,v2,u1,u2);
  return dot_online_in(u1,v1,v2)||dot_online_in(u2,v1,v2)||dot_online_in(v1,u1,u2)||dot_online_in(v2,u1,u2
      );
}
```

1.6.20 判两线段相交, 不包括端点和部分重合

```
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);
}
int intersect_ex(point3 u1,point3 u2,point3 v1,point3 v2)
{
  return dots_onplane(u1,u2,v1,v2)&&opposite_side(u1,u2,v1,v2)&&opposite_side(v1,v2,u1,u2);
}
```

1.6.21 判线段与空间三角形相交, 包括交于边界和 (部分) 包含

```
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);
}
int intersect_in(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
{
  return !same_side(l1,l2,s1,s2,s3)&&!same_side(s1,s2,l1,l2,s3)&&
    !same_side(s2,s3,l1,l2,s1)&&!same_side(s3,s1,l1,l2,s2);
}
```

1.6.22 判线段与空间三角形相交, 不包括交于边界和 (部分) 包含

```
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);
}
int intersect_ex(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
{
  return opposite_side(l1,l2,s1,s2,s3)&&opposite_side(s1,s2,l1,l2,s3)&&
    opposite_side(s2,s3,l1,l2,s1)&&opposite_side(s3,s1,l1,l2,s2);
}
```

1.6.23 计算两直线交点

```
//注意事先判断直线是否共面和平行!
//线段交点请另外判线段相交同时还是要判断是否平行(!)
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.x+=(u.b.x-u.a.x)*t;
  ret.y+=(u.b.y-u.a.y)*t;
  ret.z+=(u.b.z-u.a.z)*t;
  return ret;
}
point3 intersection(point3 u1,point3 u2,point3 v1,point3 v2)
{
  point3 ret=u1;
  double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
      /((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
  ret.x+=(u2.x-u1.x)*t;
  ret.y+=(u2.y-u1.y)*t;
  ret.z+=(u2.z-u1.z)*t;
  return ret;
}
```

1.6.24 计算直线与平面交点

```
//注意事先判断是否平行并保证三点不共线,!
//线段和空间三角形交点请另外判断
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.x=l.a.x+(l.b.x-l.a.x)*t;
  ret.y=l.a.y+(l.b.y-l.a.y)*t;
  ret.z=l.a.z+(l.b.z-l.a.z)*t;
  return ret;
}
point3 intersection(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
{
  point3 ret=pvec(s1,s2,s3);
  double t=(ret.x*(s1.x-l1.x)+ret.y*(s1.y-l1.y)+ret.z*(s1.z-l1.z))/
    (ret.x*(l2.x-l1.x)+ret.y*(l2.y-l1.y)+ret.z*(l2.z-l1.z));
  ret.x=l1.x+(l2.x-l1.x)*t;
  ret.y=l1.y+(l2.y-l1.y)*t;
  ret.z=l1.z+(l2.z-l1.z)*t;
  return ret;
}
```

1.6.25 计算两平面交线

```
//注意事先判断是否平行并保证三点不共线,!
line3 intersection(plane3 u,plane3 v)
```

```
{
  line3 ret;
  ret.a=parallel(v.a,v.b,u.a,u.b,u.c)?intersection(v.b,v.c,u.a,u.b,u.c):intersection(v.a,v.b,u.a,u.b,u.c);
  ret.b=parallel(v.c,v.a,u.a,u.b,u.c)?intersection(v.b,v.c,u.a,u.b,u.c):intersection(v.c,v.a,u.a,u.b,u.c);
  return ret;
}
line3 intersection(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)
{
  line3 ret;
  ret.a=parallel(v1,v2,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):intersection(v1,v2,u1,u2,u3);
  ret.b=parallel(v3,v1,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):intersection(v3,v1,u1,u2,u3);
  return ret;
}
```

1.6.26 点到直线距离

```
double ptoline(point3 p,line3 l)
{
  return vlen(xmult(subt(p,l.a),subt(l.b,l.a)))/dist3(l.a,l.b);
}
double ptoline(point3 p,point3 l1,point3 l2)
{
  return vlen(xmult(subt(p,l1),subt(l2,l1)))/dist3(l1,l2);
}
```

1.6.27 点到平面距离

```
double ptoplane(point3 p,plane3 s)
{
  return fabs(dmult(pvec(s),subt(p,s.a)))/vlen(pvec(s));
}
double ptoplane(point3 p,point3 s1,point3 s2,point3 s3)
{
  return fabs(dmult(pvec(s1,s2,s3),subt(p,s1)))/vlen(pvec(s1,s2,s3));
}
```

1.6.28 直线到直线距离

```
double linetoline(line3 u,line3 v)
{
  point3 n=xmult(subt(u.a,u.b),subt(v.a,v.b));
  return fabs(dmult(subt(u.a,v.a),n))/vlen(n);
}
double linetoline(point3 u1,point3 u2,point3 v1,point3 v2)
{
  point3 n=xmult(subt(u1,u2),subt(v1,v2));
  return fabs(dmult(subt(u1,v1),n))/vlen(n);
}
```

1.6.29 两直线夹角 cos 值

```
double angle_cos(line3 u,line3 v)
{
  return dmult(subt(u.a,u.b),subt(v.a,v.b))/vlen(subt(u.a,u.b))/vlen(subt(v.a,v.b));
}
double angle_cos(point3 u1,point3 u2,point3 v1,point3 v2)
{
  return dmult(subt(u1,u2),subt(v1,v2))/vlen(subt(u1,u2))/vlen(subt(v1,v2));
}
```

1.6.30 两平面夹角 cos 值

```
double angle_cos(plane3 u,plane3 v)
{
  return dmult(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)
{
  return dmult(pvec(u1,u2,u3),pvec(v1,v2,v3))/vlen(pvec(u1,u2,u3))/vlen(pvec(v1,v2,v3));
}
```

1.6.31 直线平面夹角 sin 值

```
double angle_sin(line3 l,plane3 s)
{
  return dmult(subt(l.a,l.b),pvec(s))/vlen(subt(l.a,l.b))/vlen(pvec(s));
}
```

```
double angle_sin(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
{
  return dmult(subt(l1,l2),pvec(s1,s2,s3))/vlen(subt(l1,l2))/vlen(pvec(s1,s2,s3));
}
```

1.7 网格

```
#define abs(x) ((x)>0?(x):-(x))
struct point
{
  int x,y;
};

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

//多边形上的网格点个数
int grid_onedge(int n,point* p)
{
  int i,ret=0;
  for (i=0;i<n;i++)
    ret+=gcd(abs(p[i].x-p[(i+1)%n].x),abs(p[i].y-p[(i+1)%n].y));
  return ret;
}

//多边形内的网格点个数
int grid_inside(int n,point* p)
{
  int i,ret=0;
  for (i=0;i<n;i++)
    ret+=p[(i+1)%n].y*(p[i].x-p[(i+2)%n].x);
  return (abs(ret)-grid_onedge(n,p))/2+1;
}
```

1.8 圆

1.8.1 头文件

```
#include <math.h>
#define eps 1e-8
struct point{double x,y;};

double xmult(point p1,point p2,point p0)
{
  return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}

double distance(point p1,point p2)
{
  return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}

double disptoline(point p,point l1,point l2)
{
  return fabs(xmult(p,l1,l2))/distance(l1,l2);
}

point intersection(point u1,point u2,point v1,point v2)
{
  point ret=u1;
  double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
      /((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
  ret.x+=(u2.x-u1.x)*t;
  ret.y+=(u2.y-u1.y)*t;
  return ret;
}
```

1.8.2 判直线和圆相交, 包括相切

```
int intersect_line_circle(point c,double r,point l1,point l2)
{
  return disptoline(c,l1,l2)<r+eps;
}
```

1.8.3 判线段和圆相交, 包括端点和相切

```
int intersect_seg_circle(point c,double r,point l1,point l2)
{
  double t1=distance(c,l1)-r,t2=distance(c,l2)-r;
  point t=c;
  if (t1<eps||t2<eps)
    return t1>-eps||t2>-eps;
  t.x+=l1.y-l2.y;
  t.y+=l2.x-l1.x;
  return xmult(l1,c,t)*xmult(l2,c,t)<eps&&disptoline(c,l1,l2)-r<eps;
}
```

1.8.4 判圆和圆相交, 包括相切

```
int intersect_circle_circle(point c1,double r1,point c2,double r2)
{
  return distance(c1,c2)<r1+r2+eps&&distance(c1,c2)>fabs(r1-r2)-eps;
}
```

1.8.5 计算圆上到点 p 最近点, 如 p 与圆心重合, 返回 p 本身

```
point dot_to_circle(point c,double r,point p)
{
  point u,v;
  if (distance(p,c)<eps)
    return p;
  u.x=c.x+r*fabs(c.x-p.x)/distance(c,p);
  u.y=c.y+r*fabs(c.y-p.y)/distance(c,p)*((c.x-p.x)*(c.y-p.y)<0?-1:1);
  v.x=c.x-r*fabs(c.x-p.x)/distance(c,p);
  v.y=c.y-r*fabs(c.y-p.y)/distance(c,p)*((c.x-p.x)*(c.y-p.y)<0?-1:1);
  return distance(u,p)<distance(v,p)?u:v;
}
```

1.8.6 计算直线与圆的交点

```
//计算线段与圆的交点可用这个函数后判点是否在线段上
void intersection_line_circle(point c,double r,point l1,point l2,point& p1,point& p2)
{
  point p=c;
  double t;
  p.x+=l1.y-l2.y;
  p.y+=l2.x-l1.x;
  p=intersection(p,c,l1,l2);
  t=sqrt(r*r-distance(p,c)*distance(p,c))/distance(l1,l2);
  p1.x=p.x+(l2.x-l1.x)*t;
  p1.y=p.y+(l2.y-l1.y)*t;
  p2.x=p.x-(l2.x-l1.x)*t;
  p2.y=p.y-(l2.y-l1.y)*t;
}
```

1.8.7 计算圆与圆的交点

```
void intersection_circle_circle(point c1,double r1,point c2,double r2,point& p1,point& p2)
{
  point u,v;
  double t;
  t=(1+(r1*r1-r2*r2)/distance(c1,c2)/distance(c1,c2))/2;
  u.x=c1.x+(c2.x-c1.x)*t;
  u.y=c1.y+(c2.y-c1.y)*t;
  v.x=u.x+c1.y-c2.y;
  v.y=u.y-c1.x+c2.x;
  intersection_line_circle(c1,r1,u,v,p1,p2);
}
```

1.9 凸包重心

```
#include<stdio.h>
#include<string.h>
#include<algorithm>
#include<math.h>
using namespace std;
#define maxn 100000
struct point{double x,y;}a[maxn],b[maxn];
int top;
int N;
point vex[maxn];
bool cmp(const point &x,const point &y)
{
  if(x.y==y.y) return x.x<y.x;
  return x.y<y.y;
}
double xmult(point a,point b,point c){return (b.x-a.x)*(c.y-a.y)-(b.y-a.y)*(c.x-a.x);}
void convex(int n,point P[])
{
  sort(P,P+n,cmp);
  vex[0]=P[0];
  vex[1]=P[1];
  top=1;
  for(int i=2;i<n;i++)
  {
    while(top&&xmult(vex[top],vex[top-1],P[i])<=0) top--;
    vex[++top]=P[i];
  }
  int len=top;
  vex[++top]=P[n-2];
  for(int i=n-3;i>=0;i--)
  {
    while(top!=len&&xmult(vex[top],vex[top-1],P[i])<=0) top--;
    vex[++top]=P[i];
  }
}
double cha(point a,point b)
{
  return a.x*b.y-b.x*a.y;
}
double dian(point a,point b,point c)
{
    return (c.x-b.x)*(a.x-b.x)+(c.y-b.y)*(a.y-b.y);
}
bool pan(point a,point b,point c)
{
    if(dian(a,b,c)>0&&dian(c,a,b)>0)
        return 1;
    return 0;
}
int main()
{
    int T;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d",&N);
        for(int i=0;i<N;i++)
        {
            scanf("%lf%lf",&a[i].x,&a[i].y);
            b[i]=a[i];
        }
        convex(N,b);
        double are=0;
        for(int i=0;i<N;i++)
        {
            are+=cha(a[i],a[(i+1)%N]);
          //  printf("%f\n",are);
        }
        are/=2.0;//fabs(are)/2.0;
        point zhong;
        zhong.x=0;
        zhong.y=0;
        for(int i=0;i<N;i++)
            zhong.x+=(a[i].x+a[(i+1)%N].x)*cha(a[i],a[(i+1)%N]);
        for(int i=0;i<N;i++)
            zhong.y+=(a[i].y+a[(i+1)%N].y)*cha(a[i],a[(i+1)%N]);
        zhong.x/=(6.0*are);
        zhong.y/=(6.0*are);
    }
}
```

1.10 半平面交

```
#include <cstdio>
#include <algorithm>
#include <cmath>
const double eps=1e-8;
using namespace std;
struct point
{
  double x,y;
  point(){}
  point(double x,double y)
  {
    this->x=x;
    this->y=y;
  }
  point operator -(const point &b) const
  {
    return point(x-b.x,y-b.y);
  }
  double operator *(const point &b) const
  {
    return x*b.y-y*b.x;
  }
};
struct line
{
  point s,e;
  double k;
  line(point a=point(),point b=point())
  {
    s=a,e=b;
    k=atan2(e.y-s.y,e.x-s.x);
  }
  point operator &(const line &b) const
  {
    point res=s;
    double t=((s-b.s)*(b.s-b.e))/((s-e)*(b.s-b.e));
    res.x+=(e.x-s.x)*t;
    res.y+=(e.y-s.y)*t;
    return res;
  }
};
line Q[1101];
double xmult(point p0,point p1,point p2)
{
  return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
bool HPIcmp(line a,line b)
{
  if(fabs(a.k-b.k)>eps)
    return a.k<b.k;
  return ((a.s-b.s)*(b.e-b.s))<0;
}
void HPI(line L[],int n,point res[],int &resn)
{
  int tot=n;
  sort(L,L+n,HPIcmp);
  tot=1;
  for(int i=1;i<n;i++)
    if(fabs(L[i].k-L[i-1].k)>eps)
      L[tot++]=L[i];
  int head=0,tail=1;
  Q[0]=L[0];
  Q[1]=L[1];
  resn=0;
  for(int i=2;i<tot;i++)
  {
    if(fabs((Q[tail].e-Q[tail].s)*(Q[tail-1].e-Q[tail-1].s))<eps
      ||fabs((Q[head].e-Q[head].s)*(Q[head+1].e-Q[head+1].s))<eps)
      return;
    while(head<tail&&(((Q[tail]&Q[tail-1])-L[i].s)*(L[i].e-L[i].s))>eps)
      tail--;
    while(head<tail&&(((Q[head]&Q[head+1])-L[i].s)*(L[i].e-L[i].s))>eps)
      head++;
    Q[++tail]=L[i];
  }
  while(head<tail&&(((Q[tail]&Q[tail-1])-Q[head].s)*(Q[head].e-Q[head].s))>eps)
    tail--;
  while(head<tail&&(((Q[head]&Q[head+1])-Q[tail].s)*(Q[tail].e-Q[tail].s))>eps)
    head++;
  if(tail<=head+1)
    return;
  for(int i=head;i<tail;i++)
    res[resn++]=Q[i]&Q[i+1];
  if(head<tail+1)
    res[resn++]=Q[head]&Q[tail];
}
int main()
{
  // fop;
  int t;
  scanf("%d",&t);
  while(t--)
  {
    int n;
    scanf("%d",&n);
    point P[1011];
    for(int i=0;i<n;i++)
      scanf("%lf%lf",&P[i].x,&P[i].y);
    int dir=0;
    double sum=0;
    for(int i=0;i<n;i++)
      sum+=xmult(P[i],P[(i+1)%n],P[(i+2)%n]);
    if(sum<0) dir=0;
    else dir=1;
    line LL[1011];
    if(dir)
      for(int i=0;i<n;i++)
        LL[i]=line(P[i],P[(i+1)%n]);
    else for(int i=n-1;i>=0;i--)
        LL[i]=line(P[i],P[(i-1+n)%n]);
    point res[1011];
    int resn=0;
    HPI(LL,n,res,resn);
    puts(resn?"YES":"NO");
  }
}
```

1.11 圆面积并

```
#include <bits/stdc++.h>
using namespace std;
const double eps=1e-12;
const double pi=acos(-1.0);
struct point
{
    double x,y;
    point(double _x=0,double _y=0)
    {
        x=_x;
        y=_y;
    };
    point(point s,point e)
    {
        x=e.x-s.x;
        y=e.y-s.y;
    };
    double length()
    {
        return sqrt(x*x+y*y);
    }
};
struct circle
{
    point c;
    double r;
};
struct event
{
    double tim;
    int typ;
    event(double _tim=0,int _typ=0)
    {
        tim=_tim;
        typ=_typ;
    }
};
int cmp(const double &a,const double &b)
{
    if(fabs(a-b)<eps)
        return 0;
    if(a<b)
        return -1;
    return 1;
}
bool eventcmp(const event &a,const event &b)
{
    return cmp(a.tim,b.tim)<0;
}
double area(double theta,double r)
{
    return 0.5*r*r*(theta-sin(theta));
}
double xmult(point a,point b)
{
    return a.x*b.y-a.y*b.x;
}
int n,cur,tote;
circle c[1001];
double ans[1011],pre[1011],AB,AC,BC,theta,fai,a0,a1;
event e[4004];
point lab;
int N;
bool del[1100];
void calc()
{
    for(int i=0;i<n;i++)
        del[i]=0;
    for(int i=0;i<n;i++)
        if(del[i]==0)
        {
            for(int j=0;j<n;j++)
                if(i!=j)
                {
                    if(del[j]==0)
                        if(cmp(point(c[i].c,c[j].c).length()+c[j].r,c[i].r)<=0)
                            del[j]=1;
                }
        }
    int tn=n;
    n=0;
    for(int i=0;i<tn;i++)
        if(del[i]==0)
            c[n++]=c[i];
    for(int i=1;i<=tn;i++)
        ans[i]=0.0;
    for(int i=0;i<n;i++)
    {
        tote=0;
        e[tote++]=event(-pi,1);
        e[tote++]=event(pi,-1);
        for(int j=0;j<n;j++)
            if(j!=i)
            {
                lab=point(c[j].c.x-c[i].c.x,c[j].c.y-c[i].c.y);
                AB=lab.length();
                AC=c[i].r;
                BC=c[j].r;
                if(cmp(AB+AC,BC)<=0)
                {
                    e[tote++]=event(-pi,1);
                    e[tote++]=event(pi,-1);
                    continue;
                }
                if(cmp(AB+BC,AC)<=0)
                    continue;
                if(cmp(AB,AC+BC)>0)
                    continue;
                theta=atan2(lab.y,lab.x);
                fai=acos((AC*AC+AB*AB-BC*BC)/(2.0*AC*AB));
                a0=theta-fai;
                if(cmp(a0,-pi)<0) a0+=2*pi;
                a1=theta+fai;
                if(cmp(a1,pi)>0) a1-=2*pi;
                if(cmp(a0,a1)>0)
                {
                    e[tote++]=event(a0,1);
                    e[tote++]=event(pi,-1);
                    e[tote++]=event(-pi,1);
                    e[tote++]=event(a1,-1);
                }
                else
                {
                    e[tote++]=event(a0,1);
                    e[tote++]=event(a1,-1);
                }
            }
        sort(e,e+tote,eventcmp);
        cur=0;
        for(int j=0;j<tote;j++)
        {
            if(cur!=0&&cmp(e[j].tim,pre[cur])!=0)
            {
                ans[cur]+=area(e[j].tim-pre[cur],c[i].r);
                ans[cur]+=xmult(point(c[i].c.x+c[i].r*cos(pre[cur]),c[i].c.y+c[i].r*sin(pre[cur])),
                                point(c[i].c.x+c[i].r*cos(e[j].tim),c[i].c.y+c[i].r*sin(e[j].tim)))/2.0;
            }
            cur+=e[j].typ;
            pre[cur]=e[j].tim;
        }
    }
    printf("%.3f\n",ans[1]);
    for(int i=1;i<n;i++)
        ans[i]-=ans[i+1];
}
int main()
{
    while(scanf("%d",&n)>0)
    {
        for(int i=0;i<n;i++)
        {
            scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
        }
        calc();
    }
}
```

1.12 圆与多边形交

```
#include <bits/stdc++.h>
#define pi acos(-1.0)
using namespace std;
const double eps=1e-10;
inline double max(double a,double b)
{
  if(a>b)
    return a;
  return b;
}
inline double min(double a,double b)
{
  if(a>b)
    return b;
  return a;
}
inline int fi(double a)
{
  if(a>eps)
    return 1;
  else if(a>=-eps)return 0;
  return -1;
}
class vector
{
  public:
  double x,y;
  vector(){}
  vector(double x0,double y0)
  {
    x=x0,y=y0;
  }
  double operator *(const vector& a) const
  {
    return x*a.y-y*a.x;
  }
  double operator %(const vector& a) const
  {
    return x*a.x+y*a.y;
  }
  vector verti() const
  {
    return vector(-y,x);
  }
  double length() const
  {
    return sqrt(x*x+y*y);
  }
  vector adjust(double len)
  {
    double o1=len/length();
    return vector(x*o1,y*o1);
  }
  vector oppose()
  {
    return vector(-x,-y);
  }
};
class point
{
public:
  double x,y;
  point (){}
  point (double x0,double y0)
  {
    x=x0,y=y0;
  }
  vector operator -(const point& a) const
  {
    return vector(x-a.x,y-a.y);
  }
  point operator +(const vector& a) const
  {
    return point(x+a.x,y+a.y);
  }
};
class segment
{
  public:
  point a,b;
  segment(){}
  segment(point a0,point b0)
  {
    a=a0,b=b0;
  }
  point intersert(const segment& s) const
  {
    vector v1=s.a-a,v2=s.b-a,v3=s.b-b,v4=s.a-b;
    double s1=v1*v2,s2=v3*v4;
    double se=s1+s2;
    s1/=se;
    s2/=se;
    return point(a.x*s2+b.x*s1,a.y*s2+b.y*s1);
  }
  point pverti(const point& p) const
  {
    vector t=(b-a).verti();
    segment uv(p,p+t);
    return intersert(uv);
  }
  bool on_seg(const point &p) const
  {
    if(fi(min(a.x,b.x)-p.x)<=0&&fi(p.x-max(a.x,b.x))<=0&&
      fi(min(a.y,b.y)-p.y)<=0&&fi(p.y-max(a.y,b.y))<=0)return true;
    else return false;
  }
};
double radius;
point polygon[10];
double kuras_area(point a,point b,point cir)
{
  point ori=point(cir.x,cir.y);
  // printf("%.2f %.2f\n",cir.x,cir.y);
  int sgn=fi((b-ori)*(a-ori));
  double da=(a-ori).length(),db=(b-ori).length();
  // printf("%.2f %.2f\n",da,db);
  int ra=fi(da-radius),rb=fi(db-radius);
  double angle = acos(((b-ori)%(a-ori))/(da*db));
  // printf("%.2f\n",angle);
  segment t(a,b); point h,u; vector seg;
  double ans,dlt,mov,tangle;
  if(fi(da)==0||fi(db)==0)
    return 0;
  else if(sgn==0)
    return 0;
  else if(ra<=0&&rb<=0)
    return fabs((b-ori)*(a-ori))/2*sgn;
  else if(ra>=0&&rb>=0)
  {
    h=t.pverti(ori);
    dlt=(h-ori).length();
    if(!t.on_seg(h)||fi(dlt-radius)>=0)
```

```
      return radius*radius*(angle/2)*sgn;
    else
    {
      ans=radius*radius*(angle/2);
      tangle=acos(dlt/radius);
      ans-=radius*radius*tangle;
      ans+=radius*sin(tangle)*dlt;
      // printf("%.2f\n",ans);
      return ans*sgn;
    }
  }
  else
  {
    h=t.pverti(ori);
    dlt=(h-ori).length();
    seg=b-a;
    mov=sqrt(radius*radius-dlt*dlt);
    seg=seg.adjust(mov);
    if(t.on_seg(h+seg)) u=h+seg;
    else u=h+seg.oppose();
    if(ra==1) swap(a,b);
    ans=fabs((a-ori)*(u-ori))/2;
    tangle=acos(((u-ori)%(b-ori))/((u-ori).length()*(b-ori).length()));
    ans+=radius*radius*(tangle/2);
    return ans*sgn;
  }
}
int main()
{
  int cas=0;
  double x1,x2,x3,y1,y2,y3;
  while(scanf("%lf%lf%lf",&x1,&y1,&radius)>0)
  {
    if(cas++)
      puts("");
    scanf("%lf%lf%lf%lf",&x2,&y2,&x3,&y3);
    polygon[0]=point(x2,y2);
    polygon[3]=point(x2,y3);
    polygon[2]=point(x3,y3);
    polygon[1]=point(x3,y2);
    double area=0;
    for(int i=0;i<4;i++)
    {
      area+=kuras_area(polygon[i],polygon[(i+1)%4],point(x1,y1));
    }
    printf("%.16f\n",fabs(area));
  }
}
```

1.13 三维凸包

```
#define PR 1e-8
#define N 510
struct TPoint
{
    double x,y,z;
    TPoint(){}
    TPoint(double _x,double _y,double _z):x(_x),y(_y),z(_z){}
    TPoint operator-(const TPoint p) {return TPoint(x-p.x,y-p.y,z-p.z);}
    TPoint operator*(const TPoint p) {return TPoint(y*p.z-z*p.y,z*p.x-x*p.z,x*p.y-y*p.x);}//叉积
    double operator^(const TPoint p) {return x*p.x+y*p.y+z*p.z;}//点积
};
struct fac//
{
    int a,b,c;//凸包一个面上的三个点的编号
    bool ok;//该面是否是最终凸包中的面
};
struct T3dhull
{
    int n;//初始点数
    TPoint ply[N];//初始点
    int trianglecnt;//凸包上三角形数
    fac tri[N];//凸包三角形
    int vis[N][N];//点到点是属于哪个面ij
    double dist(TPoint a){return sqrt(a.x*a.x+a.y*a.y+a.z*a.z);}//两点长度
    double area(TPoint a,TPoint b,TPoint c){return dist((b-a)*(c-a));}//三角形面积*2
    double volume(TPoint a,TPoint b,TPoint c,TPoint d){return (b-a)*(c-a)^(d-a);}//四面体有向体积*6
    double ptoplane(TPoint &p,fac &f)//正：点在面同向
    {
        TPoint m=ply[f.b]-ply[f.a],n=ply[f.c]-ply[f.a],t=p-ply[f.a];
        return (m*n)^t;
    }
    void deal(int p,int a,int b)
    {
        int f=vis[a][b];
        fac add;
        if(tri[f].ok)
        {
            if((ptoplane(ply[p],tri[f]))>PR) dfs(p,f);
            else
            {
                add.a=b,add.b=a,add.c=p,add.ok=1;
                vis[p][b]=vis[a][p]=vis[b][a]=trianglecnt;
                tri[trianglecnt++]=add;
            }
        }
    }
    void dfs(int p,int cnt)//维护凸包，如果点在凸包外更新凸包p
    {
        tri[cnt].ok=0;
        deal(p,tri[cnt].b,tri[cnt].a);
        deal(p,tri[cnt].c,tri[cnt].b);
        deal(p,tri[cnt].a,tri[cnt].c);
    }
    bool same(int s,int e)//判断两个面是否为同一面
    {
        TPoint a=ply[tri[s].a],b=ply[tri[s].b],c=ply[tri[s].c];
        return fabs(volume(a,b,c,ply[tri[e].a]))<PR
            &&fabs(volume(a,b,c,ply[tri[e].b]))<PR
            &&fabs(volume(a,b,c,ply[tri[e].c]))<PR;
    }
    void construct()//构建凸包
    {
        int i,j;
        trianglecnt=0;
        if(n<4) return ;
        bool tmp=true;
        for(i=1;i<n;i++)//前两点不共点
        {
            if((dist(ply[0]-ply[i]))>PR)
            {
                swap(ply[1],ply[i]); tmp=false; break;
            }
        }
        if(tmp) return;
        tmp=true;
        for(i=2;i<n;i++)//前三点不共线
        {
            if((dist((ply[0]-ply[1])*(ply[1]-ply[i])))>PR)
            {
                swap(ply[2],ply[i]); tmp=false; break;
            }
        }
        if(tmp) return ;
        tmp=true;
        for(i=3;i<n;i++)//前四点不共面
        {
            if(fabs((ply[0]-ply[1])*(ply[1]-ply[2])^(ply[0]-ply[i]))>PR)
            {
                swap(ply[3],ply[i]); tmp=false; break;
            }
        }
        if(tmp) return ;
        fac add;
        for(i=0;i<4;i++)//构建初始四面体
        {
            add.a=(i+1)%4,add.b=(i+2)%4,add.c=(i+3)%4,add.ok=1;
            if((ptoplane(ply[i],add))>0) swap(add.b,add.c);
            vis[add.a][add.b]=vis[add.b][add.c]=vis[add.c][add.a]=trianglecnt;
            tri[trianglecnt++]=add;
        }
        for(i=4;i<n;i++)//构建更新凸包
        {
            for(j=0;j<trianglecnt;j++)
            {
                if(tri[j].ok&&(ptoplane(ply[i],tri[j]))>PR)
                {
                    dfs(i,j); break;
                }
            }
        }
        int cnt=trianglecnt;
        trianglecnt=0;
        for(i=0;i<cnt;i++)
        {
            if(tri[i].ok)
                tri[trianglecnt++]=tri[i];
        }
    }
    double area()//表面积
    {
        double ret=0;
        for(int i=0;i<trianglecnt;i++)
            ret+=area(ply[tri[i].a],ply[tri[i].b],ply[tri[i].c]);
        return ret/2.0;
    }
    double volume()//体积
    {
        TPoint p(0,0,0);
        double ret=0;
        for(int i=0;i<trianglecnt;i++)
            ret+=volume(p,ply[tri[i].a],ply[tri[i].b],ply[tri[i].c]);
        return fabs(ret/6);
    }
    int facetri() {return trianglecnt;}//表面三角形数
    int facepolygon()//表面多边形数
    {
        int ans=0,i,j,k;
        for(i=0;i<trianglecnt;i++)
        {
            for(j=0,k=1;j<i;j++)
            {
                if(same(i,j)) {k=0;break;}
            }
            ans+=k;
        }
        return ans;
    }
}hull;
```

2 Graph

2.1 Tarjan

```
const int MAX=1605000;
struct node
{
    int v,next,w;
}g[MAX],g2[MAX];
int adj[MAX],low[MAX],dfn[MAX],bel[MAX];
int e,Index,cnt,n,m,x[MAX],y[MAX],inStack[MAX];
int maxx,vis[MAX],e2,adj2[MAX];
stack<int>s;
void add(int u,int v)
{
    g[e].v=v; g[e].next=adj[u]; adj[u]=e++;
}
void tarjan(int u)
{
    low[u]=dfn[u]=++Index;
    s.push(u);
    inStack[u]=1;
    int i,v;
    for(i=adj[u];i!=-1;i=g[i].next)
    {
        v=g[i].v;
        if(!dfn[v])
        {
            tarjan(v);
            low[u]=min(low[u],low[v]);
        }
        else if(inStack[v])
            low[u]=min(low[u],dfn[v]);
    }
    if(low[u]==dfn[u])
    {
        cnt++;
        do
        {
            v=s.top();
            s.pop();
            inStack[v]=0;
            bel[v]=cnt;
        }while(u!=v);
    }
}
```

2.2 Sap

```
#include <bits/stdc++.h>
using namespace std;
int m;
struct Edge
{
    int en,cap,flow,next;
} edge[E];
int head[N] , tot , now[N];
int source,sink,tot_num;
int pre[N] , dis[N] , gap[N];
int n;
struct node
{
     int x,y,z,f,l;
}p[122];
void add(int st,int en,int cap)
{
    edge[tot].en=en;
    edge[tot].cap=cap;
    edge[tot].flow=0;
    edge[tot].next=head[st];
    head[st]=tot++;

    edge[tot].en=st;
    edge[tot].cap=0;
    edge[tot].flow=0;
    edge[tot].next=head[en];
    head[en]=tot++;
}
```

```

void augment(int flow)
{
    for(int i=source;i!=sink;i=edge[now[i]].en)
    {
        edge[now[i]].flow+=flow;
        edge[now[i]^1].flow-=flow;
    }
}
int summ=0;
int sap()
{
    memset(dis,0,sizeof(dis));
    memset(gap,0,sizeof(gap));
    memset(pre,-1,sizeof(pre));
    for(int i=0;i<tot_num;i++)
        now[i]=head[i];
    gap[0]=tot_num;
    int point=source,flow=0,min_flow=INT_INF;
    while(dis[source]<tot_num)
    {
        bool fg=false;
        for(int i=now[point];i!=-1;i=edge[i].next)
            if(edge[i].cap-edge[i].flow>0 && dis[point]==dis[edge[i].en]+1)
            {
                min_flow=min(min_flow,edge[i].cap-edge[i].flow);
                now[point]=i;
                pre[edge[i].en]=point;
                point=edge[i].en;
                if(point==sink)
                {
                    flow+=min_flow;
                    augment(min_flow);
                    point=source;
                    min_flow=INT_INF;
                }
                fg=true;
                break;
            }
        if(fg) continue;
        if(--gap[dis[point]]==0) break;
        int Min=tot_num;
        for(int i=head[point];i!=-1;i=edge[i].next)
            if(edge[i].cap-edge[i].flow>0 && Min>dis[edge[i].en])
            {
                Min=dis[edge[i].en];
                now[point]=i;
            }
        gap[dis[point]=Min+1]++;
        if(point!=source) point=pre[point];
    }
    return flow;
}
int main()
{
    while(scanf("%d%d",&m,&n)>0)
    {
        tot=0;
        source=0;
        sink=n-1;
        tot_num=n;
        memset(head,-1,sizeof(head));
        for(int i=0;i<m;i++)
        {
            int a,b,c;
            scanf("%d%d%d",&a,&b,&c);
            add(a-1,b-1,c);
        }
        printf("%d\n",sap());
    }
}
```

2.3 费用流

```
using namespace std;
struct Edge
{
    int st,en,cap,flow,cost,next;
} edge[E];
int head[N] , tot , now[N];
int source,sink;
int pre[N] , dis[N];
queue<int> q;
bool vs[N];

void add(int st,int en,int cap,int cost)
{
    edge[tot].st=st;
    edge[tot].en=en;
    edge[tot].cap=cap;
    edge[tot].flow=0;
    edge[tot].cost=cost;
    edge[tot].next=head[st];
    head[st]=tot++;

    edge[tot].st=en;
    edge[tot].en=st;
    edge[tot].cap=0;
    edge[tot].flow=0;
    edge[tot].cost=-cost;
    edge[tot].next=head[en];
    head[en]=tot++;
}

bool SPFA()
{
    for(int i=0; i<N; i++)
        dis[i]=INF;
    memset(vs,0,sizeof(vs));
    memset(now,-1,sizeof(now));
    while(!q.empty()) q.pop();
    q.push(source); dis[source]=0; vs[source]=1;
    while(!q.empty())
    {
        int u=q.front(); q.pop(); vs[u]=0;
        for(int i=head[u],v; i!=-1; i=edge[i].next)
            if(edge[i].cap-edge[i].flow>0 && dis[v=edge[i].en]>dis[u]+edge[i].cost)
            {
                dis[v]=dis[u]+edge[i].cost;
                now[v]=i;
                if(!vs[v])
                {
                    vs[v]=1;
                    q.push(v);
                }
            }
    }
    if(dis[sink]!=INF) return true;
    else return false;
}

int MCMF()
{
    int cost=0;
    while(SPFA())
    {
        int flow=INF;
        for(int u=sink; u!=source; u=edge[now[u]].st)
            if(flow>edge[now[u]].cap-edge[now[u]].flow)
                flow=edge[now[u]].cap-edge[now[u]].flow;
        for(int u=sink; u!=source; u=edge[now[u]].st)
        {
            edge[now[u]].flow+=flow;
            edge[now[u]^1].flow-=flow;
        }
        cost+=flow*dis[sink];
    }
    return cost;
}
int n,m;
struct node
{
    int x;
    int y;
    node(int x=0,int y=0){this->x=x,this->y=y;};
};
node ph[111],pm[111];
char maz[111][111]={0};
int dist[111][111]={0};
void build()
```

```
{
    clr_1(head);
    tot=0;
    clr(dist);
    int cnt1=0;
    int cnt2=0;
    source=0; sink=(n+m)+1;
    for(int i=0;i<n;i++)
    {
       scanf("%s",maz[i]);
       for(int j=0;j<m;j++)
       {
           if(maz[i][j]=='H')
           {
               ph[++cnt1]=node(i,j);
               add(0,cnt1,1,0);
           }
           else if(maz[i][j]=='m')
           {
               pm[++cnt2]=node(i,j);
           }
       }
    }
    sink=cnt1+cnt2+1;
    for(int i=1;i<=cnt2;i++)
        add(cnt1+i,sink,1,0);
    for(int i=1;i<=cnt1;i++)
    {
        for(int j=1;j<=cnt2;j++)
        {
            int go=abs(ph[i].x-pm[j].x)+abs(ph[i].y-pm[j].y);
            add(i,cnt1+j,INF,go);
        }
    }
    printf("%d\n",MCMF());
    return;
}

int main()
{
   // fop;
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        if(n==m&&m==0)
           break;
        build();
    }
    return 0;
}
```

3 字符串

3.1 后缀数组 dc3

```
const int maxn = 150000;
int s[maxn*3];
int rank[maxn*3],height[maxn*3];
int r[maxn*3],sa[maxn*3];
int wa[maxn*3],wb[maxn*3],wv[maxn*3],ss[maxn*3],sa2[maxn*3];
int n,m,k;

#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)

int c0(int *r,int a,int b){
    return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];
}
int c12(int k,int *r,int a,int b){
    if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
    else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
}
void sort(int *r,int *a,int *b,int n,int m){
     int i;
     for(i=0;i<n;i++) wv[i]=r[a[i]];
     for(i=0;i<m;i++) ss[i]=0;
     for(i=0;i<n;i++) ss[wv[i]]++;
     for(i=1;i<m;i++) ss[i]+=ss[i-1];
     for(i=n-1;i>=0;i--) b[--ss[wv[i]]]=a[i];
     return;
}
//用n+1
void dc3(int *r,int *sa,int n,int m){
     int i,j,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p,*rn=r+n;
     r[n]=r[n+1]=0;
     for(i=0;i<n;i++) if(i%3!=0) wa[tbc++]=i;
     sort(r+2,wa,wb,tbc,m);
     sort(r+1,wb,wa,tbc,m);
     sort(r,wa,wb,tbc,m);
     for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
     rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
     if(p<tbc) dc3(rn,san,tbc,p);
     else for(i=0;i<tbc;i++) san[rn[i]]=i;
     for(i=0;i<tbc;i++) if(san[i]<tb) wb[ta++]=san[i]*3;
     if(n%3==1) wb[ta++]=n-1;
     sort(r,wb,wa,ta,m);
     for(i=0;i<tbc;i++) wv[wb[i]=G(san[i])]=i;
     for(i=0,j=0,p=0;i<ta && j<tbc;p++)
     sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
     for(;i<ta;p++) sa[p]=wa[i++];
     for(;j<tbc;p++) sa[p]=wb[j++];
     return;
}
//用n+1
void calheight(int *r,int *sa,int n){
    int i,j,k=0;
    for (i=0;i<n;++i) rank[sa[i]]=i;
    for (i=0;i<n;height[rank[i++]]=k)
        for (k?k--:0,j=sa[rank[i]-1];r[i+k]==r[j+k];k++);
}
```

3.2 后缀数组 da

```
const int maxn=100000;
int sa[maxn],rank[maxn],rank2[maxn],height[maxn],c[maxn],*x,*y,s[maxn],n;
void radix_sort(int n,int sz)
{
    memset(c,0,sizeof(c));
    for(int i=0;i<n;i++) c[x[y[i]]]++;
    for(int i=1;i<sz;i++) c[i]+=c[i-1];
    for(int i=n-1;i>=0;i--) sa[--c[x[y[i]]]]=y[i];
}
void get_sa(int *s,int n,int sz=222)
{
    x=rank,y=rank2;
    for(int i=0;i<n;i++)
        x[i]=s[i],y[i]=i;
    radix_sort(n,sz);
    for(int len=1;len<n;len<<=1)
    {
        int yid=0;
        for(int i=n-len;i<n;i++) y[yid++]=i;
        for(int i=0;i<n;i++) if(sa[i]>=len) y[yid++]=sa[i]-len;
        radix_sort(n,sz);
        swap(x,y);
        x[sa[0]]=yid=0;
        for(int i=1;i<n;i++)
        {
            if(y[sa[i]]==y[sa[i-1]]&&sa[i-1]+len<n&&sa[i]+len<n&&y[sa[i]+len]==y[sa[i-1]+len])
                x[sa[i]]=yid;
            else x[sa[i]]=++yid;
        }
        sz=yid+1;
        if(sz>=n) break;
    }
    for(int i=0;i<n;i++)
        rank[i]=x[i];
}
void get_height(int * s,int n)
{
    int k=0;height[0]=0;
    for(int i=0;i<n;i++)
    {
        if(rank[i]==0) continue;
        k=max(0,k-1);
        int j=sa[rank[i]-1];
        while(i+k<n&&j+k<n&&s[i+k]==s[j+k]) k++;
        height[rank[i]]=k;
    }
}
```

3.3 后缀自动机

```
#include <set>
#include <cmath>
#include <stack>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <algorithm>
#include <cstdlib>
#include <numeric>
#include <vector>
#include <ctime>
#include <queue>
#include <list>
#include <map>
#define pi acos(-1.0)
#define INF 0x3f3f3f3f
#define clr(x)  memset(x,0,sizeof(x));
#define clrto(x,siz,y)  for(int xx=0;xx<=siz;xx++)  x[xx]=y;
#define clrset(x,siz)  for(int xx=0;xx<=siz;xx++)  x[xx]=xx;
#define clr_1(x) memset(x,-1,sizeof(x));
#define clrmax(x) memset(x,0x3f,sizeof(x));
#define clrvec(x,siz) for(int xx=0;xx<=siz;xx++)  x[xx].clear();
#define fop2   freopen(".in","r",stdin); //freopen(".out","w",stdout);
#define fop   freopen("in.txt","r",stdin);//freopen("out.txt","w",stdout);
#define myprogram By_135678942570
#define clrcpy(x,siz,y)  for(int xx=0;xx<siz;xx++)  x[xx]=y[xx];
#define pb push_back
using namespace std;
const int INF = (-1u) >> 2;
const int maxn = 200010;
const int kind = 27;
struct NODE{
  int fail, next[kind], step;
  int pos; //该节点在字符串中的位置
  int id; //副本与本体拥有相同的 id
  void clear(){
    clr(next);
    id = step = fail = pos = 0;
  }
};
struct suffix_automation{ //支持删除的后缀自动机 !!!
  NODE node[maxn << 1]; //原串长+ 可能插入的 <= maxn 但每个可能还有一个副本，乘， 2
  int last, total;
  int len; //当前字符串长度
  int idx[maxn << 1]; //字符串中第 i 个字符对应结点 idx[i]
  bool is_del[maxn << 1];
  void init(){
    total = last = 0;
    len = 0;
    node[0].clear();
    is_del[0] = 0;
    clr(idx);
  }
  void push_back(int step_val){
    int p = ++total;
    node[p].clear();
    node[p].step = step_val;
    node[p].id = p;
    is_del[p] = 0;
  }
  bool unExistNorDel(int x){
    return x == 0 || is_del[node[x].id];
  }
  void insert(int ch){
    push_back(node[last].step + 1);
    node[total].pos = ++len;idx[len] = total; // 第 len 个字符对应第 total 个结点
    int p = last, np = total;
    for (;unExistNorDel(node[p].next[ch]); p = node[p].fail) node[p].next[ch] = np;
    if (node[p].next[ch] == np) //此时一定等于p 0
      node[np].fail = p;
    else {
      int q = node[p].next[ch];
      if (node[q].step == node[p].step + 1) node[np].fail = q;
      else {
        int nq = ++total;
        node[nq] = node[q];
        node[nq].step = node[p].step + 1;
        node[q].fail = node[np].fail = nq;
        for (;node[p].next[ch] == q; p = node[p].fail) node[p].next[ch] = nq;
      }
    }
    last = np;
  }
  void addString(char *str){
    int i;
    for (i = 0; str[i]; i++)
      insert(str[i] - 'a' + 1);
  }
  void delString(int k){ //从后往前删除 个字符k
    int  i;
    for (i = 1; i <= k; i++){
      is_del[node[idx[len - i + 1]].id] = 1;
    }
    last = idx[len - k];
    len -= k;
  }
  bool first;
  int dfs(int x, int &l){
    if (l == 0) return node[x].pos;
    int i;
    for (i = 0; i < kind; i++){
      if (i == 0 && first){
        first = 0;
        continue;
      }
      if (unExistNorDel(node[x].next[i]))continue;
      l--;
      return dfs(node[x].next[i], l);
    }
    return node[x].pos;
  }
  int query(int l){ //求长度为的子串中字典序最小的子串，返回位置l 若为则为所有子串中字典序最小的那个llen
    first = 1;
    insert(0);
    int tmp = l;
    int id = dfs(0, tmp);
    delString(1);
    int det = l - tmp;
    return id - det + 1;
  }
}obj;
int main()
{

}
```

4 杂物

4.1 Dancing Links

```
#include<bits/stdc++.h>
#define clr(x)  memset(x,0,sizeof(x));
using namespace std;
int U[333333];
int D[333333];
int L[333333];
int R[333333];
int S[333333];
int H[333333];
int C[333333];
int res[333333];
char mp[22][22];
void del(int CC)
{
  R[L[CC]]=R[CC];
  L[R[CC]]=L[CC];
  for(int i=D[CC];i!=CC;i=D[i])
    for(int j=R[i];j!=i;j=R[j])
    {
      D[U[j]]=D[j];
      U[D[j]]=U[j];
      S[C[j]]--;
    }
}
void bac(int CC)
{
  R[L[CC]]=CC;
  L[R[CC]]=CC;
  for(int i=D[CC];i!=CC;i=D[i])
    for(int j=R[i];j!=i;j=R[j])
    {
      D[U[j]]=j;
      U[D[j]]=j;
      S[C[j]]++;
    }
}
int dfs(int dep)
{
  if(R[0]==0)
  {
    for(int i=0;i<dep;i++)
      mp[res[i]/256][res[i]%256/16]=res[i]%16+'A';
    return 1;
  }
  if(R[0]>256)
  return 0;
  int w=INF;
  int now;
  int nxt;
  for(int i=R[0];i!=0;i=R[i])
    if(S[i]<w)
    {
      now=i;
      w=S[i];
    }
  nxt=D[now];
  del(now);
  while(nxt!=now)
  {
    res[dep]=H[nxt];
    for(int i=R[nxt];i!=nxt;i=R[i])
      del(C[i]);
    if(dfs(dep+1))
      return 1;
    for(int i=L[nxt];i!=nxt;i=L[i])
      bac(C[i]);
    nxt=D[nxt];
  }
  bac(now);
  return 0;
}
int last;
int cnt=256*4;
void linkinit(int u,int v)
```

```
{
  L[v]=u,R[u]=v,D[v]=U[v]=v,last=v;
}
void init()
{
  last=0;
  for(int i=1;i<=256;i++)
  {
    if(mp[(i-1)/16][(i-1)%16]!='-')
      continue;
    linkinit(last,i);
  }
  for(int i=1;i<=256;i++)
  {
    int flag=1;
    int h=(i-1)/16;
    int nn=(i-1)%16;
    for(int ii=0;ii<=15;ii++)
      if(mp[h][ii]==nn+'A')
      {
        flag=0;
        break;
      }
    if(flag)
      linkinit(last,i+256);
  }
  for(int i=1;i<=256;i++)
  {
    int flag=1;
    int h=(i-1)/16;
    int nn=(i-1)%16;
    for(int ii=0;ii<=15;ii++)
      if(mp[ii][h]==nn+'A')
      {
        flag=0;
        break;
      }
    if(flag)
      linkinit(last,i+256+256);
  }
  for(int i=1;i<=256;i++)
  {
    int flag=1;
    int nx=((i-1)/16/4*4);
    int ny=((i-1)/16%4*4);
    int nn=(i-1)%16;
    for(int ii=0;ii<=3;ii++)
      for(int jj=0;jj<=3;jj++)
        if(mp[ii+nx][jj+ny]==nn+'A')
        {
          flag=0;
          ii=4;
          break;
        }
    if(flag)
      linkinit(last,i+256+256+256);
  }
  R[last]=0;
  L[0]=last;
}
void linknode(int pos,int h)
{
    cnt++;
    H[cnt]=h;
    C[cnt]=pos;
    D[cnt]=C[cnt];
    U[cnt]=U[C[cnt]];
    D[U[C[cnt]]]=cnt;
    U[C[cnt]]=cnt;
    S[C[cnt]]++;
}
int main()
{
  while(scanf("%s",mp[0])>0)
  {
      cnt=256*4;
    clr(U);clr(D);clr(R);clr(L);clr(H);clr(C);clr(res);clr(S);
    for(int i=1;i<16;i++)
      scanf("%s",mp[i]);
    init();
    for(int i=0;i<16;i++)
      for(int j=0;j<16;j++)
        if(mp[i][j]=='-')
        for(int k=0;k<16;k++)
        {
          int flag=1;
          for(int ii=0;ii<16;ii++)
            if(mp[ii][j]=='A'+k||mp[i][ii]=='A'+k)
            {
              flag=0;
              break;
            }
          int nx=i/4*4;
          int ny=j/4*4;
          for(int ii=0;ii<4;ii++)
            for(int jj=0;jj<4;jj++)
              if(mp[nx+ii][ny+jj]=='A'+k)
              {
                flag=0;
                ii=5;
                break;
              }
          if(flag==0)
            continue;
          int h=i*256+j*16+k;
          int fg=0;
          if(D[i*16+j+1])
          {
              linknode(i*16+j+1,h);
            fg=cnt;
                    }
                    if(D[257+i*16+k])
                    {
              linknode(257+i*16+k,h);
            if(fg) L[cnt]=cnt-1,R[cnt-1]=cnt;
            else fg=cnt;
          }
          if(D[513+j*16+k])
          {
              linknode(513+j*16+k,h);
            if(fg) L[cnt]=cnt-1,R[cnt-1]=cnt;
            else fg=cnt;
          }
          if(D[769+(i/4*4+j/4)*16+k])
          {
              linknode(769+(i/4*4+j/4)*16+k,h);
            if(fg) L[cnt]=cnt-1,R[cnt-1]=cnt;
            else fg=cnt;
          }
          if(fg) L[fg]=cnt,R[cnt]=fg;
        }
    dfs(0);
    for(int i=0;i<16;i++)
    {
      mp[i][16]=0;
      printf("%s\n",mp[i]);
    }
    puts("");
  }
  return 0;
}
```