

---

# ACM TEMPLATE

---

135\_Morisummer

Last build at July 8, 2014

# Contents

<b>1</b>	<b>Geometry</b>	<b>3</b>
1.1	注意	3
1.2	几何公式	3
1.2.1	三角形	3
1.2.2	四边形	3
1.2.3	圆内接四边形	4
1.2.4	正 $N$ 边形	4
1.2.5	圆	4
1.2.6	棱柱	4
1.2.7	棱锥	4
1.2.8	正棱锥	4
1.2.9	棱台	4
1.2.10	正棱台	4
1.2.11	圆柱	5
1.2.12	圆锥	5
1.2.13	圆台	5
1.2.14	球	5
1.2.15	球台	5
1.2.16	球扇形	5
1.3	多边形	5
1.3.1	头文件	5
1.3.2	判定凸多边形, 允许相邻边共线	6
1.3.3	判定凸多边形, 不允许相邻边共线	6
1.3.4	判点在凸多边形内或多边形边上	6
1.3.5	判点在凸多边形内	6
1.3.6	判点在任意多边形内	6
1.3.7	判线段在任意多边形内	6
1.3.8	多边形重心	7
1.4	浮点函数	7
1.4.1	头文件	7
1.4.2	两点距离	8
1.4.3	判三点共线	8
1.4.4	判点在线段上, 包括端点	8
1.4.5	判点在线段上, 不包括端点	8
1.4.6	判两点在线段同侧, 点在线段上返回 0	8
1.4.7	判两点在线段异侧, 点在线段上返回 0	9
1.4.8	判两直线平行	9
1.4.9	判两直线垂直	9
1.4.10	判两线段相交, 包括端点和部分重合	9
1.4.11	判两线段相交, 不包括端点和部分重合	9
1.4.12	计算两直线交点	9
1.4.13	点到直线上的最近点	10
1.4.14	点到直线距离	10
1.4.15	点到线段上的最近点	10
1.4.16	点到线段距离	10
1.4.17	矢量 $V$ 以 $P$ 为顶点逆时针旋转 $angle$ 并放大 $scale$ 倍	11
1.5	三角形	11
1.5.1	头文件	11
1.5.2	外心	11
1.5.3	内心	11
1.5.4	垂心	11
1.5.5	重心	12
1.5.6	费马点	12
1.6	三维几何	12
1.6.1	头文件	12
1.6.2	取平面法向量	13
1.6.3	判三点共线	13
1.6.4	判四点共面	13
1.6.5	判点是否在线段上, 包括端点和共线	13
1.6.6	判点是否在线段上, 不包括端点	13
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	13
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	13

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	14
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	14
1.6.11	判两点在平面同侧, 点在平面上返回 0	14
1.6.12	判两点在平面异侧, 点在平面上返回 0	14
1.6.13	判两直线平行	14
1.6.14	判两平面平行	14
1.6.15	判直线与平面平行	15
1.6.16	判两直线垂直	15
1.6.17	判两平面垂直	15
1.6.18	判直线与平面垂直	15
1.6.19	判两线段相交, 包括端点和部分重合	15
1.6.20	判两线段相交, 不包括端点和部分重合	15
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	15
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	16
1.6.23	计算两直线交点	16
1.6.24	计算直线与平面交点	16
1.6.25	计算两平面交线	16
1.6.26	点到直线距离	17
1.6.27	点到平面距离	17
1.6.28	直线到直线距离	17
1.6.29	两直线夹角 $\cos$ 值	17
1.6.30	两平面夹角 $\cos$ 值	17
1.6.31	直线平面夹角 $\sin$ 值	17
1.7	网格	18
1.8	圆	18
1.8.1	头文件	18
1.8.2	判直线和圆相交, 包括相切	18
1.8.3	判线段和圆相交, 包括端点和相切	18
1.8.4	判圆和圆相交, 包括相切	19
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	19
1.8.6	计算直线与圆的交点	19
1.8.7	计算圆与圆的交点	19
1.9	凸包重心	19
1.10	半平面交	20
1.11	圆面积并	22
1.12	圆与多边形交	24
1.13	三维凸包	26
<b>2</b>	<b>Graph</b>	<b>29</b>
2.1	Tarjan	29
2.2	Sap	29
2.3	费用流	30
<b>3</b>	<b>字符串</b>	<b>33</b>
3.1	后缀数组 dc3	33
3.2	后缀数组 da	33
3.3	后缀自动机	34
<b>4</b>	<b>杂物</b>	<b>36</b>
4.1	Dancing Links	36

# 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.  $cross\ product = |u| \times |v| \times \sin(a)$   
 $dot\ product = |u| \times |v| \times \cos(a)$
- IX.  $(P1 - P0) \times (P2 - P0)$  结果的意义:  
正:  $\langle P0, P1 \rangle$  在  $\langle P0, P2 \rangle$  顺时针  $(0, \pi)$  内  
负:  $\langle P0, P1 \rangle$  在  $\langle P0, P2 \rangle$  逆时针  $(0, \pi)$  内  
0:  $\langle P0, P1 \rangle, \langle P0, P2 \rangle$  共线, 夹角为 0 或  $\pi$
- 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  $Ta = \frac{\sqrt{b \times c \times ((b+c)^2 - a^2)}}{b+c} = \frac{2 \times b \times c \times \cos(\frac{A}{2})}{b+c}$
- 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,  $M$  为对角线中点连线,  $A$  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

```

1 #define MAXN 1000
2 #define offset 10000
3 #define eps 1e-8
4 #define zero(x) (((x)>0?(x):-x))<eps)
5 #define _sign(x) ((x)>eps?1:((x)<=-eps?-1:0))
6 struct point{double x,y;};
7 struct line{point a,b;};
8 double xmult(point p1,point p2,point p0)
9 {
10     return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
11 }

```

### 1.3.2 判定凸多边形, 允许相邻边共线

```

1 int is_convex(int n, point* p)
2 {
3     int i, s[3] = {1, 1, 1};
4     for (i = 0; i < n && s[1] | s[2]; i++)
5         s[_sign(xmult(p[(i+1)%n], p[(i+2)%n], p[i]))] = 0;
6     return s[1] | s[2];
7 }

```

### 1.3.3 判定凸多边形, 不允许相邻边共线

```

1 int is_convex_v2(int n, point* p)
2 {
3     int i, s[3] = {1, 1, 1};
4     for (i = 0; i < n && s[0] && s[1] | s[2]; i++)
5         s[_sign(xmult(p[(i+1)%n], p[(i+2)%n], p[i]))] = 0;
6     return s[0] && s[1] | s[2];
7 }

```

### 1.3.4 判点在凸多边形内或多边形边上

```

1 int inside_convex(point q, int n, point* p)
2 {
3     int i, s[3] = {1, 1, 1};
4     for (i = 0; i < n && s[1] | s[2]; i++)
5         s[_sign(xmult(p[(i+1)%n], q, p[i]))] = 0;
6     return s[1] | s[2];
7 }

```

### 1.3.5 判点在凸多边形内

```

1 int inside_convex_v2(point q, int n, point* p)
2 {
3     int i, s[3] = {1, 1, 1};
4     for (i = 0; i < n && s[0] && s[1] | s[2]; i++)
5         s[_sign(xmult(p[(i+1)%n], q, p[i]))] = 0;
6     return s[0] && s[1] | s[2];
7 }

```

### 1.3.6 判点在任意多边形内

```

1 //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
2 int inside_polygon(point q, int n, point* p, int on_edge=1)
3 {
4     point q2;
5     int i=0, count;
6     while (i < n)
7         for (count=i=0, q2.x=rand()+offset, q2.y=rand()+offset; i < n; i++)
8             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)
9                 return on_edge;
10            else if (zero(xmult(q, q2, p[i])))
11                break;
12            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)
13                count++;
14            count++;
15    return count & 1;
16 }

```

### 1.3.7 判线段在任意多边形内

```

1 inline int opposite_side(point p1, point p2, point l1, point l2)
2 {
3     return xmult(l1, p1, l2)*xmult(l1, p2, l2) < -eps;
4 }
5 inline int dot_online_in(point p, point l1, point l2)
6 {
7     return zero(xmult(p, l1, l2)) && (l1.x-p.x)*(l2.x-p.x) < eps && (l1.y-p.y)*(l2.y-p.y) < eps;
8 }
9 //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
10 int inside_polygon_v2(point l1, point l2, int n, point* p)
11 {
12     point t[MXN], tt;
13     int i, j, k=0;
14     if (!inside_polygon(l1, n, p) || !inside_polygon(l2, n, p))

```

```

15     return 0;
16     for (i=0;i<n;i++)
17         if (opposite_side(l1,l2,p[i],p[(i+1)%n])&&opposite_side(p[i],p[(i+1)%n],l1,l2))
18             return 0;
19     else if (dot_online_in(l1,p[i],p[(i+1)%n]))
20         t[k++]=l1;
21     else if (dot_online_in(l2,p[i],p[(i+1)%n]))
22         t[k++]=l2;
23     else if (dot_online_in(p[i],l1,l2))
24         t[k++]=p[i];
25     for (i=0;i<k;i++)
26         for (j=i+1;j<k;j++){
27             tt.x=(t[i].x+t[j].x)/2;
28             tt.y=(t[i].y+t[j].y)/2;
29             if (!inside_polygon(tt,n,p))
30                 return 0;
31         }
32     return 1;
33 }

```

### 1.3.8 多边形重心

```

1 point intersection(line u,line v)
2 {
3     point ret=u.a;
4     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))
5         /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
6     ret.x+=(u.b.x-u.a.x)*t;
7     ret.y+=(u.b.y-u.a.y)*t;
8     return ret;
9 }
10
11 point barycenter(point a,point b,point c)
12 {
13     line u,v;
14     u.a.x=(a.x+b.x)/2;
15     u.a.y=(a.y+b.y)/2;
16     u.b=c;
17     v.a.x=(a.x+c.x)/2;
18     v.a.y=(a.y+c.y)/2;
19     v.b=b;
20     return intersection(u,v);
21 }
22
23 //多边形重心
24 point barycenter(int n,point* p)
25 {
26     point ret,t;
27     double t1=0,t2;
28     int i;
29     ret.x=ret.y=0;
30     for (i=1;i<n-1;i++){
31         if (fabs(t2=xmult(p[0],p[i],p[i+1]))>eps){
32             t=barycenter(p[0],p[i],p[i+1]);
33             ret.x+=t.x*t2;
34             ret.y+=t.y*t2;
35             t1+=t2;
36         }
37     }
38     if (fabs(t1)>eps)
39         ret.x/=t1,ret.y/=t1;
40     return ret;
41 }

```

## 1.4 浮点函数

### 1.4.1 头文件

```

1 #include <math.h>
2 #define eps 1e-8
3 #define zero(x) (((x)>0?(x):-x))<eps)
4 struct point
5 {
6     double x,y;
7     point(double a=0.0,b=0.0)
8     {
9         x=a,y=b;
10    }
11 };
12 struct line

```



```

13 {
14     point a,b;
15     line(point s=point(),point e=point())
16     {
17         a=s,b=e;
18     }
19 };
20 double xmult(point p1,point p2,point p0)
21 {
22     return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
23 }
24 double xmult(double x1,double y1,double x2,double y2,double x0,double y0)
25 {
26     return (x1-x0)*(y2-y0)-(x2-x0)*(y1-y0);
27 }
28 double dmult(point p1,point p2,point p0)
29 {
30     return (p1.x-p0.x)*(p2.x-p0.x)+(p1.y-p0.y)*(p2.y-p0.y);
31 }
32 double dmult(double x1,double y1,double x2,double y2,double x0,double y0)
33 {
34     return (x1-x0)*(x2-x0)+(y1-y0)*(y2-y0);
35 }

```

### 1.4.2 两点距离

```

1 double dis(point p1,point p2)
2 {
3     return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
4 }
5 double dis(double x1,double y1,double x2,double y2)
6 {
7     return sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
8 }

```

### 1.4.3 判三点共线

```

1 int dots_inline(point p1,point p2,point p3)
2 {
3     return zero(xmult(p1,p2,p3));
4 }
5 int dots_inline(double x1,double y1,double x2,double y2,double x3,double y3)
6 {
7     return zero(xmult(x1,y1,x2,y2,x3,y3));
8 }

```

### 1.4.4 判点在线段上，包括端点

```

1 int dot_online_in(point p,line l)
2 {
3     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;
4 }
5 int dot_online_in(point p,point l1,point l2)
6 {
7     return zero(xmult(p,l1,l2))&&(l1.x-p.x)*(l2.x-p.x)<eps&&(l1.y-p.y)*(l2.y-p.y)<eps;
8 }
9 int dot_online_in(double x,double y,double x1,double y1,double x2,double y2)
10 {
11     return zero(xmult(x,y,x1,y1,x2,y2))&&(x1-x)*(x2-x)<eps&&(y1-y)*(y2-y)<eps;
12 }

```

### 1.4.5 判点在线段上，不包括端点

```

1 int dot_online_ex(point p,line l)
2 {
3     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));
4 }
5 int dot_online_ex(point p,point l1,point l2)
6 {
7     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));
8 }
9 int dot_online_ex(double x,double y,double x1,double y1,double x2,double y2)
10 {
11     return dot_online_in(x,y,x1,y1,x2,y2)&&(!zero(x-x1)||!zero(y-y1))&&(!zero(x-x2)||!zero(y-y2));
12 }

```

### 1.4.6 判两点在线段同侧，点在线段上返回 0

```

1 int same_side(point p1,point p2,line l)
2 {
3     return xmult(l.a,p1,l.b)*xmult(l.a,p2,l.b)>eps;
4 }
5 int same_side(point p1,point p2,point l1,point l2)
6 {
7     return xmult(l1,p1,l2)*xmult(l1,p2,l2)>eps;
8 }

```

#### 1.4.7 判两点在线段异侧, 点在线段上返回 0

```

1 int opposite_side(point p1,point p2,line l)
2 {
3     return xmult(l.a,p1,l.b)*xmult(l.a,p2,l.b)<=-eps;
4 }
5 int opposite_side(point p1,point p2,point l1,point l2)
6 {
7     return xmult(l1,p1,l2)*xmult(l1,p2,l2)<=-eps;
8 }

```

#### 1.4.8 判两直线平行

```

1 int parallel(line u,line v)
2 {
3     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));
4 }
5 int parallel(point u1,point u2,point v1,point v2)
6 {
7     return zero((u1.x-u2.x)*(v1.y-v2.y)-(v1.x-v2.x)*(u1.y-u2.y));
8 }

```

#### 1.4.9 判两直线垂直

```

1 int perpendicular(line u,line v)
2 {
3     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));
4 }
5 int perpendicular(point u1,point u2,point v1,point v2)
6 {
7     return zero((u1.x-u2.x)*(v1.x-v2.x)+(u1.y-u2.y)*(v1.y-v2.y));
8 }

```

#### 1.4.10 判两线段相交, 包括端点和部分重合

```

1 int intersect_in(line u,line v)
2 {
3     if (!dots_inline(u.a,u.b,v.a)||!dots_inline(u.a,u.b,v.b))
4         return !same_side(u.a,u.b,v)&&!same_side(v.a,v.b,u);
5     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);
6 }
7 int intersect_in(point u1,point u2,point v1,point v2)
8 {
9     if (!dots_inline(u1,u2,v1)||!dots_inline(u1,u2,v2))
10        return !same_side(u1,u2,v1,v2)&&!same_side(v1,v2,u1,u2);
11    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);
12 }

```

#### 1.4.11 判两线段相交, 不包括端点和部分重合

```

1 int intersect_ex(line u,line v)
2 {
3     return opposite_side(u.a,u.b,v)&&opposite_side(v.a,v.b,u);
4 }
5 int intersect_ex(point u1,point u2,point v1,point v2)
6 {
7     return opposite_side(u1,u2,v1,v2)&&opposite_side(v1,v2,u1,u2);
8 }

```

#### 1.4.12 计算两直线交点

```

1 point intersection(line u,line v)
2 {
3     point ret=u.a;
4     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))
5         /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
6     ret.x+=(u.b.x-u.a.x)*t;

```

```

7   ret.y+=(u.b.y-u.a.y)*t;
8   return ret;
9 }
10 point intersection(point u1,point u2,point v1,point v2)
11 {
12     point ret=u1;
13     double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
14             /((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
15     ret.x+=(u2.x-u1.x)*t;
16     ret.y+=(u2.y-u1.y)*t;
17     return ret;
18 }

```

#### 1.4.13 点到直线上的最近点

```

1 point ptoline(point p,line l)
2 {
3     point t=p;
4     t.x+=l.a.y-l.b.y,t.y+=l.b.x-l.a.x;
5     return intersection(p,t,l.a,l.b);
6 }
7 point ptoline(point p,point l1,point l2)
8 {
9     point t=p;
10    t.x+=l1.y-l2.y,t.y+=l2.x-l1.x;
11    return intersection(p,t,l1,l2);
12 }

```

#### 1.4.14 点到直线距离

```

1 double disptoline(point p,line l)
2 {
3     return fabs(xmult(p,l.a,l.b))/distance(l.a,l.b);
4 }
5 double disptoline(point p,point l1,point l2)
6 {
7     return fabs(xmult(p,l1,l2))/distance(l1,l2);
8 }
9 double disptoline(double x,double y,double x1,double y1,double x2,double y2)
10 {
11     return fabs(xmult(x,y,x1,y1,x2,y2))/distance(x1,y1,x2,y2);
12 }

```

#### 1.4.15 点到线段上的最近点

```

1 point ptoseg(point p,line l)
2 {
3     point t=p;
4     t.x+=l.a.y-l.b.y,t.y+=l.b.x-l.a.x;
5     if (xmult(l.a,t,p)*xmult(l.b,t,p)>eps)
6         return distance(p,l.a)<distance(p,l.b)?l.a:l.b;
7     return intersection(p,t,l.a,l.b);
8 }
9 point ptoseg(point p,point l1,point l2)
10 {
11     point t=p;
12     t.x+=l1.y-l2.y,t.y+=l2.x-l1.x;
13     if (xmult(l1,t,p)*xmult(l2,t,p)>eps)
14         return distance(p,l1)<distance(p,l2)?l1:l2;
15     return intersection(p,t,l1,l2);
16 }

```

#### 1.4.16 点到线段距离

```

1 double disptoseg(point p,line l)
2 {
3     point t=p;
4     t.x+=l.a.y-l.b.y,t.y+=l.b.x-l.a.x;
5     if (xmult(l.a,t,p)*xmult(l.b,t,p)>eps)
6         return distance(p,l.a)<distance(p,l.b)?distance(p,l.a):distance(p,l.b);
7     return fabs(xmult(p,l.a,l.b))/distance(l.a,l.b);
8 }
9 double disptoseg(point p,point l1,point l2)
10 {
11     point t=p;
12     t.x+=l1.y-l2.y,t.y+=l2.x-l1.x;
13     if (xmult(l1,t,p)*xmult(l2,t,p)>eps)
14         return distance(p,l1)<distance(p,l2)?distance(p,l1):distance(p,l2);
15     return fabs(xmult(p,l1,l2))/distance(l1,l2);

```

16 | }

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

```

1 point rotate(point v,point p,double angle,double scale)
2 {
3     point ret=p;
4     v.x-=p.x,v.y-=p.y;
5     p.x=scale*cos(angle);
6     p.y=scale*sin(angle);
7     ret.x+=v.x*p.x-v.y*p.y;
8     ret.y+=v.x*p.y+v.y*p.x;
9     return ret;
10 }

```

## 1.5 三角形

### 1.5.1 头文件

```

1 #include <math.h>
2 struct point{double x,y};
3 struct line{point a,b};
4
5 double distance(point p1,point p2)
6 {
7     return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
8 }
9
10 point intersection(line u,line v)
11 {
12     point ret=u.a;
13     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))
14         /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
15     ret.x+=(u.b.x-u.a.x)*t;
16     ret.y+=(u.b.y-u.a.y)*t;
17     return ret;
18 }

```

### 1.5.2 外心

```

1 point circumcenter(point a,point b,point c)
2 {
3     line u,v;
4     u.a.x=(a.x+b.x)/2;
5     u.a.y=(a.y+b.y)/2;
6     u.b.x=u.a.x-a.y+b.y;
7     u.b.y=u.a.y+a.x-b.x;
8     v.a.x=(a.x+c.x)/2;
9     v.a.y=(a.y+c.y)/2;
10    v.b.x=v.a.x-a.y+c.y;
11    v.b.y=v.a.y+a.x-c.x;
12    return intersection(u,v);
13 }

```

### 1.5.3 内心

```

1 point incenter(point a,point b,point c)
2 {
3     line u,v;
4     double m,n;
5     u.a=a;
6     m=atan2(b.y-a.y,b.x-a.x);
7     n=atan2(c.y-a.y,c.x-a.x);
8     u.b.x=u.a.x+cos((m+n)/2);
9     u.b.y=u.a.y+sin((m+n)/2);
10    v.a=b;
11    m=atan2(a.y-b.y,a.x-b.x);
12    n=atan2(c.y-b.y,c.x-b.x);
13    v.b.x=v.a.x+cos((m+n)/2);
14    v.b.y=v.a.y+sin((m+n)/2);
15    return intersection(u,v);
16 }

```

### 1.5.4 垂心

```

1 point perpencenter(point a,point b,point c)
2 {

```

```

3   line u,v;
4   u.a=c;
5   u.b.x=u.a.x-a.y+b.y;
6   u.b.y=u.a.y+a.x-b.x;
7   v.a=b;
8   v.b.x=v.a.x-a.y+c.y;
9   v.b.y=v.a.y+a.x-c.x;
10  return intersection(u,v);
11 }

```

### 1.5.5 重心

```

1 point barycenter(point a,point b,point c)
2 {
3   line u,v;
4   u.a.x=(a.x+b.x)/2;
5   u.a.y=(a.y+b.y)/2;
6   u.b=c;
7   v.a.x=(a.x+c.x)/2;
8   v.a.y=(a.y+c.y)/2;
9   v.b=b;
10  return intersection(u,v);
11 }

```

### 1.5.6 费马点

```

1 point fermentpoint(point a,point b,point c)
2 {
3   point u,v;
4   double step=fabs(a.x)+fabs(a.y)+fabs(b.x)+fabs(b.y)+fabs(c.x)+fabs(c.y);
5   int i,j,k;
6   u.x=(a.x+b.x+c.x)/3;
7   u.y=(a.y+b.y+c.y)/3;
8   while (step>1e-10)
9     for (k=0;k<10;step/=2,k++)
10      for (i=-1;i<=1;i++)
11        for (j=-1;j<=1;j++)
12          {
13            v.x=u.x+step*i;
14            v.y=u.y+step*j;
15            if (distance(u,a)+distance(u,b)+distance(u,c)>distance(v,a)+distance(v,b)+distance(v,c))
16              u=v;
17          }
18   return u;
19 }

```

## 1.6 三维几何

### 1.6.1 头文件

```

1 #include <math.h>
2 #define eps 1e-8
3 #define zero(x) (((x)>0?(x):-x))<eps)
4 struct point3{double x,y,z;};
5 struct line3{point3 a,b;};
6 struct plane3{point3 a,b,c;};
7
8 point3 xmult(point3 u,point3 v)
9 {
10  point3 ret;
11  ret.x=u.y*v.z-v.y*u.z;
12  ret.y=u.z*v.x-u.x*v.z;
13  ret.z=u.x*v.y-u.y*v.x;
14  return ret;
15 }
16
17 double dmult(point3 u,point3 v)
18 {
19  return u.x*v.x+u.y*v.y+u.z*v.z;
20 }
21
22 point3 subtr(point3 u,point3 v)
23 {
24  point3 ret;
25  ret.x=u.x-v.x;
26  ret.y=u.y-v.y;
27  ret.z=u.z-v.z;
28  return ret;

```

```

29 }
30
31 double dist3(point3 p1,point3 p2)
32 {
33     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));
34 }
35
36 double vlen(point3 p)
37 {
38     return sqrt(p.x*p.x+p.y*p.y+p.z*p.z);
39 }

```

### 1.6.2 取平面法向量

```

1 point3 pvec(plane3 s)
2 {
3     return xmult(subt(s.a,s.b),subt(s.b,s.c));
4 }
5 point3 pvec(point3 s1,point3 s2,point3 s3)
6 {
7     return xmult(subt(s1,s2),subt(s2,s3));
8 }

```

### 1.6.3 判三点共线

```

1 int dots_inline(point3 p1,point3 p2,point3 p3)
2 {
3     return vlen(xmult(subt(p1,p2),subt(p2,p3)))<eps;
4 }

```

### 1.6.4 判四点共面

```

1 int dots_onplane(point3 a,point3 b,point3 c,point3 d)
2 {
3     return zero(dmult(pvec(a,b,c),subt(d,a)));
4 }

```

### 1.6.5 判点是否在线段上, 包括端点和共线

```

1 int dot_online_in(point3 p,line3 l)
2 {
3     return zero(vlen(xmult(subt(p,l.a),subt(p,l.b))))&&(l.a.x-p.x)*(l.b.x-p.x)<eps&&
4         (l.a.y-p.y)*(l.b.y-p.y)<eps&&(l.a.z-p.z)*(l.b.z-p.z)<eps;
5 }
6 int dot_online_in(point3 p,point3 l1,point3 l2)
7 {
8     return zero(vlen(xmult(subt(p,l1),subt(p,l2))))&&(l1.x-p.x)*(l2.x-p.x)<eps&&
9         (l1.y-p.y)*(l2.y-p.y)<eps&&(l1.z-p.z)*(l2.z-p.z)<eps;
10 }

```

### 1.6.6 判点是否在线段上, 不包括端点

```

1 int dot_online_ex(point3 p,line3 l)
2 {
3     return dot_online_in(p,l)&&(!zero(p.x-l.a.x)||!zero(p.y-l.a.y)||!zero(p.z-l.a.z))&&
4         (!zero(p.x-l.b.x)||!zero(p.y-l.b.y)||!zero(p.z-l.b.z));
5 }
6 int dot_online_ex(point3 p,point3 l1,point3 l2)
7 {
8     return dot_online_in(p,l1,l2)&&(!zero(p.x-l1.x)||!zero(p.y-l1.y)||!zero(p.z-l1.z))&&
9         (!zero(p.x-l2.x)||!zero(p.y-l2.y)||!zero(p.z-l2.z));
10 }

```

### 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

```

1 int dot_inplane_in(point3 p,plane3 s)
2 {
3     return zero(vlen(xmult(subt(s.a,s.b),subt(s.a,s.c)))-vlen(xmult(subt(p,s.a),subt(p,s.b)))-
4         vlen(xmult(subt(p,s.b),subt(p,s.c)))-vlen(xmult(subt(p,s.c),subt(p,s.a))));
5 }
6 int dot_inplane_in(point3 p,point3 s1,point3 s2,point3 s3)
7 {
8     return zero(vlen(xmult(subt(s1,s2),subt(s1,s3)))-vlen(xmult(subt(p,s1),subt(p,s2)))-
9         vlen(xmult(subt(p,s2),subt(p,s3)))-vlen(xmult(subt(p,s3),subt(p,s1))));
10 }

```

### 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

```

1 int dot_inplane_ex(point3 p,plane3 s)
2 {
3     return dot_inplane_in(p,s)&&vlen(xmult(subt(p,s.a),subt(p,s.b)))>eps&&
4         vlen(xmult(subt(p,s.b),subt(p,s.c)))>eps&&vlen(xmult(subt(p,s.c),subt(p,s.a)))>eps;
5 }
6 int dot_inplane_ex(point3 p,point3 s1,point3 s2,point3 s3)
7 {
8     return dot_inplane_in(p,s1,s2,s3)&&vlen(xmult(subt(p,s1),subt(p,s2)))>eps&&
9         vlen(xmult(subt(p,s2),subt(p,s3)))>eps&&vlen(xmult(subt(p,s3),subt(p,s1)))>eps;
10 }

```

### 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

```

1 int same_side(point3 p1,point3 p2,line3 l)
2 {
3     return dmult(xmult(subt(l.a,l.b),subt(p1,l.b)),xmult(subt(l.a,l.b),subt(p2,l.b)))>eps;
4 }
5 int same_side(point3 p1,point3 p2,point3 l1,point3 l2)
6 {
7     return dmult(xmult(subt(l1,l2),subt(p1,l2)),xmult(subt(l1,l2),subt(p2,l2)))>eps;
8 }

```

### 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

```

1 int opposite_side(point3 p1,point3 p2,line3 l)
2 {
3     return dmult(xmult(subt(l.a,l.b),subt(p1,l.b)),xmult(subt(l.a,l.b),subt(p2,l.b)))<-eps;
4 }
5 int opposite_side(point3 p1,point3 p2,point3 l1,point3 l2)
6 {
7     return dmult(xmult(subt(l1,l2),subt(p1,l2)),xmult(subt(l1,l2),subt(p2,l2)))<-eps;
8 }

```

### 1.6.11 判两点在平面同侧, 点在平面上返回 0

```

1 int same_side(point3 p1,point3 p2,plane3 s)
2 {
3     return dmult(pvec(s),subt(p1,s.a))*dmult(pvec(s),subt(p2,s.a))>eps;
4 }
5 int same_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3)
6 {
7     return dmult(pvec(s1,s2,s3),subt(p1,s1))*dmult(pvec(s1,s2,s3),subt(p2,s1))>eps;
8 }

```

### 1.6.12 判两点在平面异侧, 点在平面上返回 0

```

1 int opposite_side(point3 p1,point3 p2,plane3 s)
2 {
3     return dmult(pvec(s),subt(p1,s.a))*dmult(pvec(s),subt(p2,s.a))<-eps;
4 }
5 int opposite_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3)
6 {
7     return dmult(pvec(s1,s2,s3),subt(p1,s1))*dmult(pvec(s1,s2,s3),subt(p2,s1))<-eps;
8 }

```

### 1.6.13 判两直线平行

```

1 int parallel(line3 u,line3 v)
2 {
3     return vlen(xmult(subt(u.a,u.b),subt(v.a,v.b)))<eps;
4 }
5 int parallel(point3 u1,point3 u2,point3 v1,point3 v2)
6 {
7     return vlen(xmult(subt(u1,u2),subt(v1,v2)))<eps;
8 }

```

### 1.6.14 判两平面平行

```

1 int parallel(plane3 u,plane3 v)
2 {
3     return vlen(xmult(pvec(u),pvec(v)))<eps;
4 }
5 int parallel(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)
6 {
7     return vlen(xmult(pvec(u1,u2,u3),pvec(v1,v2,v3)))<eps;
8 }

```

### 1.6.15 判直线与平面平行

```

1 | int parallel(line3 l,plane3 s)
2 | {
3 |     return zero(dmult(subt(l.a,l.b),pvec(s)));
4 | }
5 | int parallel(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
6 | {
7 |     return zero(dmult(subt(l1,l2),pvec(s1,s2,s3)));
8 | }

```

### 1.6.16 判两直线垂直

```

1 | int perpendicular(line3 u,line3 v)
2 | {
3 |     return zero(dmult(subt(u.a,u.b),subt(v.a,v.b)));
4 | }
5 | int perpendicular(point3 u1,point3 u2,point3 v1,point3 v2)
6 | {
7 |     return zero(dmult(subt(u1,u2),subt(v1,v2)));
8 | }

```

### 1.6.17 判两平面垂直

```

1 | int perpendicular(plane3 u,plane3 v)
2 | {
3 |     return zero(dmult(pvec(u),pvec(v)));
4 | }
5 | int perpendicular(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)
6 | {
7 |     return zero(dmult(pvec(u1,u2,u3),pvec(v1,v2,v3)));
8 | }

```

### 1.6.18 判直线与平面垂直

```

1 | int perpendicular(line3 l,plane3 s)
2 | {
3 |     return vlen(xmult(subt(l.a,l.b),pvec(s)))<eps;
4 | }
5 | int perpendicular(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
6 | {
7 |     return vlen(xmult(subt(l1,l2),pvec(s1,s2,s3)))<eps;
8 | }

```

### 1.6.19 判两线段相交, 包括端点和部分重合

```

1 | int intersect_in(line3 u,line3 v)
2 | {
3 |     if (!dots_onplane(u.a,u.b,v.a,v.b))
4 |         return 0;
5 |     if (!dots_inline(u.a,u.b,v.a)||!dots_inline(u.a,u.b,v.b))
6 |         return !same_side(u.a,u.b,v)&&!same_side(v.a,v.b,u);
7 |     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);
8 | }
9 | int intersect_in(point3 u1,point3 u2,point3 v1,point3 v2)
10 | {
11 |     if (!dots_onplane(u1,u2,v1,v2))
12 |         return 0;
13 |     if (!dots_inline(u1,u2,v1)||!dots_inline(u1,u2,v2))
14 |         return !same_side(u1,u2,v1,v2)&&!same_side(v1,v2,u1,u2);
15 |     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);
16 | }

```

### 1.6.20 判两线段相交, 不包括端点和部分重合

```

1 | int intersect_ex(line3 u,line3 v)
2 | {
3 |     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);
4 | }
5 | int intersect_ex(point3 u1,point3 u2,point3 v1,point3 v2)
6 | {
7 |     return dots_onplane(u1,u2,v1,v2)&&opposite_side(u1,u2,v1,v2)&&opposite_side(v1,v2,u1,u2);
8 | }

```

### 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



```

1 int intersect_in(line3 l,plane3 s)
2 {
3     return !same_side(l.a,l.b,s)&&!same_side(s.a,s.b,l.a,l.b,s.c)&&
4         !same_side(s.b,s.c,l.a,l.b,s.a)&&!same_side(s.c,s.a,l.a,l.b,s.b);
5 }
6 int intersect_in(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
7 {
8     return !same_side(l1,l2,s1,s2,s3)&&!same_side(s1,s2,l1,l2,s3)&&
9         !same_side(s2,s3,l1,l2,s1)&&!same_side(s3,s1,l1,l2,s2);
10 }

```

### 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

```

1 int intersect_ex(line3 l,plane3 s)
2 {
3     return opposite_side(l.a,l.b,s)&&opposite_side(s.a,s.b,l.a,l.b,s.c)&&
4         opposite_side(s.b,s.c,l.a,l.b,s.a)&&opposite_side(s.c,s.a,l.a,l.b,s.b);
5 }
6 int intersect_ex(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
7 {
8     return opposite_side(l1,l2,s1,s2,s3)&&opposite_side(s1,s2,l1,l2,s3)&&
9         opposite_side(s2,s3,l1,l2,s1)&&opposite_side(s3,s1,l1,l2,s2);
10 }

```

### 1.6.23 计算两直线交点

```

1 //注意事先判断直线是否共面和平行!
2 //线段交点请另外判线段相交时还是要判断是否平行(!)
3 point3 intersection(line3 u,line3 v)
4 {
5     point3 ret=u.a;
6     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))/
7         (((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x)));
8     ret.x+=(u.b.x-u.a.x)*t;
9     ret.y+=(u.b.y-u.a.y)*t;
10    ret.z+=(u.b.z-u.a.z)*t;
11    return ret;
12 }
13 point3 intersection(point3 u1,point3 u2,point3 v1,point3 v2)
14 {
15     point3 ret=u1;
16     double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))/
17         (((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x)));
18     ret.x+=(u2.x-u1.x)*t;
19     ret.y+=(u2.y-u1.y)*t;
20     ret.z+=(u2.z-u1.z)*t;
21     return ret;
22 }

```

### 1.6.24 计算直线与平面交点

```

1 //注意事先判断是否平行并保证三点不共线,!
2 //线段和空间三角形交点请另外判断
3 point3 intersection(line3 l,plane3 s)
4 {
5     point3 ret=pvec(s);
6     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))/
7         (ret.x*(l.b.x-l.a.x)+ret.y*(l.b.y-l.a.y)+ret.z*(l.b.z-l.a.z));
8     ret.x=l.a.x+(l.b.x-l.a.x)*t;
9     ret.y=l.a.y+(l.b.y-l.a.y)*t;
10    ret.z=l.a.z+(l.b.z-l.a.z)*t;
11    return ret;
12 }
13 point3 intersection(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
14 {
15     point3 ret=pvec(s1,s2,s3);
16     double t=(ret.x*(s1.x-l1.x)+ret.y*(s1.y-l1.y)+ret.z*(s1.z-l1.z))/
17         (ret.x*(l2.x-l1.x)+ret.y*(l2.y-l1.y)+ret.z*(l2.z-l1.z));
18     ret.x=l1.x+(l2.x-l1.x)*t;
19     ret.y=l1.y+(l2.y-l1.y)*t;
20     ret.z=l1.z+(l2.z-l1.z)*t;
21     return ret;
22 }

```

### 1.6.25 计算两平面交线

```

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

```

```

3 {
4   line3 ret;
5   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);
6   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);
7   return ret;
8 }
9 line3 intersection(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)
10 {
11   line3 ret;
12   ret.a=parallel(v1,v2,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):intersection(v1,v2,u1,u2,u3);
13   ret.b=parallel(v3,v1,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):intersection(v3,v1,u1,u2,u3);
14   return ret;
15 }

```

### 1.6.26 点到直线距离

```

1 double ptoline(point3 p,line3 l)
2 {
3   return vlen(xmult(subt(p,l.a),subt(l.b,l.a)))/dist3(l.a,l.b);
4 }
5 double ptoline(point3 p,point3 l1,point3 l2)
6 {
7   return vlen(xmult(subt(p,l1),subt(l2,l1)))/dist3(l1,l2);
8 }

```

### 1.6.27 点到平面距离

```

1 double ptoplane(point3 p,plane3 s)
2 {
3   return fabs(dmult(pvec(s),subt(p,s.a)))/vlen(pvec(s));
4 }
5 double ptoplane(point3 p,point3 s1,point3 s2,point3 s3)
6 {
7   return fabs(dmult(pvec(s1,s2,s3),subt(p,s1)))/vlen(pvec(s1,s2,s3));
8 }

```

### 1.6.28 直线到直线距离

```

1 double linetoline(line3 u,line3 v)
2 {
3   point3 n=xmult(subt(u.a,u.b),subt(v.a,v.b));
4   return fabs(dmult(subt(u.a,v.a),n))/vlen(n);
5 }
6 double linetoline(point3 u1,point3 u2,point3 v1,point3 v2)
7 {
8   point3 n=xmult(subt(u1,u2),subt(v1,v2));
9   return fabs(dmult(subt(u1,v1),n))/vlen(n);
10 }

```

### 1.6.29 两直线夹角 cos 值

```

1 double angle_cos(line3 u,line3 v)
2 {
3   return dmult(subt(u.a,u.b),subt(v.a,v.b))/vlen(subt(u.a,u.b))/vlen(subt(v.a,v.b));
4 }
5 double angle_cos(point3 u1,point3 u2,point3 v1,point3 v2)
6 {
7   return dmult(subt(u1,u2),subt(v1,v2))/vlen(subt(u1,u2))/vlen(subt(v1,v2));
8 }

```

### 1.6.30 两平面夹角 cos 值

```

1 double angle_cos(plane3 u,plane3 v)
2 {
3   return dmult(pvec(u),pvec(v))/vlen(pvec(u))/vlen(pvec(v));
4 }
5 double angle_cos(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)
6 {
7   return dmult(pvec(u1,u2,u3),pvec(v1,v2,v3))/vlen(pvec(u1,u2,u3))/vlen(pvec(v1,v2,v3));
8 }

```

### 1.6.31 直线平面夹角 sin 值

```

1 double angle_sin(line3 l,plane3 s)
2 {
3   return dmult(subt(l.a,l.b),pvec(s))/vlen(subt(l.a,l.b))/vlen(pvec(s));
4 }

```

```

5 double angle_sin(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
6 {
7     return dmult(subt(l1,l2),pvec(s1,s2,s3))/vlen(subt(l1,l2))/vlen(pvec(s1,s2,s3));
8 }

```

## 1.7 网格

```

1 #define abs(x) ((x)>0?(x):- (x))
2 struct point
3 {
4     int x,y;
5 };
6
7 int gcd(int a,int b)
8 {
9     return b?gcd(b,a%b):a;
10 }
11
12 //多边形上的网格点个数
13 int grid_onedge(int n,point* p)
14 {
15     int i,ret=0;
16     for (i=0;i<n;i++)
17         ret+=gcd(abs(p[i].x-p[(i+1)%n].x),abs(p[i].y-p[(i+1)%n].y));
18     return ret;
19 }
20
21 //多边形内的网格点个数
22 int grid_inside(int n,point* p)
23 {
24     int i,ret=0;
25     for (i=0;i<n;i++)
26         ret+=p[(i+1)%n].y*(p[i].x-p[(i+2)%n].x);
27     return (abs(ret)-grid_onedge(n,p))/2+1;
28 }

```

## 1.8 圆

### 1.8.1 头文件

```

1 #include <math.h>
2 #define eps 1e-8
3 struct point{double x,y;};
4
5 double xmult(point p1,point p2,point p0)
6 {
7     return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
8 }
9
10 double distance(point p1,point p2)
11 {
12     return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
13 }
14
15 double disptoline(point p,point l1,point l2)
16 {
17     return fabs(xmult(p,l1,l2))/distance(l1,l2);
18 }
19
20 point intersection(point u1,point u2,point v1,point v2)
21 {
22     point ret=u1;
23     double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
24         /((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
25     ret.x+=(u2.x-u1.x)*t;
26     ret.y+=(u2.y-u1.y)*t;
27     return ret;
28 }

```

### 1.8.2 判直线和圆相交, 包括相切

```

1 int intersect_line_circle(point c,double r,point l1,point l2)
2 {
3     return disptoline(c,l1,l2)<r+eps;
4 }

```

### 1.8.3 判线段和圆相交, 包括端点和相切

```

1 int intersect_seg_circle(point c,double r,point l1,point l2)
2 {
3     double t1=distance(c,l1)-r,t2=distance(c,l2)-r;
4     point t=c;
5     if (t1<eps||t2<eps)
6         return t1>-eps||t2>-eps;
7     t.x+=l1.y-l2.y;
8     t.y+=l2.x-l1.x;
9     return xmult(l1,c,t)*xmult(l2,c,t)<eps&&disptoline(c,l1,l2)-r<eps;
10 }

```

#### 1.8.4 判圆和圆相交, 包括相切

```

1 int intersect_circle_circle(point c1,double r1,point c2,double r2)
2 {
3     return distance(c1,c2)<r1+r2+eps&&distance(c1,c2)>fabs(r1-r2)-eps;
4 }

```

#### 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

```

1 point dot_to_circle(point c,double r,point p)
2 {
3     point u,v;
4     if (distance(p,c)<eps)
5         return p;
6     u.x=c.x+r*fabs(c.x-p.x)/distance(c,p);
7     u.y=c.y+r*fabs(c.y-p.y)/distance(c,p)*((c.x-p.x)*(c.y-p.y)<0?-1:1);
8     v.x=c.x-r*fabs(c.x-p.x)/distance(c,p);
9     v.y=c.y-r*fabs(c.y-p.y)/distance(c,p)*((c.x-p.x)*(c.y-p.y)<0?-1:1);
10    return distance(u,p)<distance(v,p)?u:v;
11 }

```

#### 1.8.6 计算直线与圆的交点

```

1 //计算线段与圆的交点可用这个函数后判点是否在线段上
2 void intersection_line_circle(point c,double r,point l1,point l2,point& p1,point& p2)
3 {
4     point p=c;
5     double t;
6     p.x+=l1.y-l2.y;
7     p.y+=l2.x-l1.x;
8     p=intersection(p,c,l1,l2);
9     t=sqrt(r*r-distance(p,c)*distance(p,c))/distance(l1,l2);
10    p1.x=p.x+(l2.x-l1.x)*t;
11    p1.y=p.y+(l2.y-l1.y)*t;
12    p2.x=p.x-(l2.x-l1.x)*t;
13    p2.y=p.y-(l2.y-l1.y)*t;
14 }

```

#### 1.8.7 计算圆与圆的交点

```

1 void intersection_circle_circle(point c1,double r1,point c2,double r2,point& p1,point& p2)
2 {
3     point u,v;
4     double t;
5     t=(1+(r1*r1-r2*r2)/distance(c1,c2)/distance(c1,c2))/2;
6     u.x=c1.x+(c2.x-c1.x)*t;
7     u.y=c1.y+(c2.y-c1.y)*t;
8     v.x=u.x+c1.y-c2.y;
9     v.y=u.y-c1.x+c2.x;
10    intersection_line_circle(c1,r1,u,v,p1,p2);
11 }

```

### 1.9 凸包重心

```

1 #include<stdio.h>
2 #include<string.h>
3 #include<algorithm>
4 #include<math.h>
5 using namespace std;
6 #define maxn 100000
7 struct point{double x,y;}a[maxn],b[maxn];
8 int top;
9 int N;
10 point vex[maxn];
11 bool cmp(const point &x,const point &y)
12 {
13     if(x.y==y.y) return x.x<y.x;

```

```

14     return x.y<y.y;
15 }
16 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);}
17 void convex(int n,point P[])
18 {
19     sort(P,P+n,cmp);
20     vex[0]=P[0];
21     vex[1]=P[1];
22     top=1;
23     for(int i=2;i<n;i++)
24     {
25         while(top&& xmult(vex[top],vex[top-1],P[i])<=0) top--;
26         vex[++top]=P[i];
27     }
28     int len=top;
29     vex[++top]=P[n-2];
30     for(int i=n-3;i>=0;i--)
31     {
32         while(top!=len&& xmult(vex[top],vex[top-1],P[i])<=0) top--;
33         vex[++top]=P[i];
34     }
35 }
36 double cha(point a,point b)
37 {
38     return a.x*b.y-b.x*a.y;
39 }
40 double dian(point a,point b,point c)
41 {
42     return (c.x-b.x)*(a.x-b.x)+(c.y-b.y)*(a.y-b.y);
43 }
44 bool pan(point a,point b,point c)
45 {
46     if(dian(a,b,c)>0&&dian(c,a,b)>0)
47         return 1;
48     return 0;
49 }
50 int main()
51 {
52     int T;
53     scanf("%d",&T);
54     while(T--)
55     {
56         scanf("%d",&N);
57         for(int i=0;i<N;i++)
58         {
59             scanf("%lf%lf",&a[i].x,&a[i].y);
60             b[i]=a[i];
61         }
62         convex(N,b);
63         double are=0;
64         for(int i=0;i<N;i++)
65         {
66             are+=cha(a[i],a[(i+1)%N]);
67             // printf("%f\n",are);
68         }
69         are/=2.0;//fabs(are)/2.0;
70         point zhong;
71         zhong.x=0;
72         zhong.y=0;
73         for(int i=0;i<N;i++)
74             zhong.x+=(a[i].x+a[(i+1)%N].x)*cha(a[i],a[(i+1)%N]);
75         for(int i=0;i<N;i++)
76             zhong.y+=(a[i].y+a[(i+1)%N].y)*cha(a[i],a[(i+1)%N]);
77         zhong.x/=(6.0*are);
78         zhong.y/=(6.0*are);
79     }
80 }

```

## 1.10 半平面交

```

1 #include <cstdio>
2 #include <algorithm>
3 #include <cmath>
4 const double eps=1e-8;
5 using namespace std;
6 struct point
7 {
8     double x,y;
9     point(){}
10    point(double x,double y)

```

```

11 {
12     this→x=x;
13     this→y=y;
14 }
15 point operator -(const point &b) const
16 {
17     return point(x-b.x,y-b.y);
18 }
19 double operator *(const point &b) const
20 {
21     return x*b.y-y*b.x;
22 }
23 };
24 struct line
25 {
26     point s,e;
27     double k;
28     line(point a=point(),point b=point())
29     {
30         s=a,e=b;
31         k=atan2(e.y-s.y,e.x-s.x);
32     }
33     point operator &(const line &b) const
34     {
35         point res=s;
36         double t=((s-b.s)*(b.s-b.e))/((s-e)*(b.s-b.e));
37         res.x+=(e.x-s.x)*t;
38         res.y+=(e.y-s.y)*t;
39         return res;
40     }
41 };
42 line Q[1101];
43 double xmult(point p0,point p1,point p2)
44 {
45     return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
46 }
47 bool HPICmp(line a,line b)
48 {
49     if(fabs(a.k-b.k)>eps)
50         return a.k<b.k;
51     return ((a.s-b.s)*(b.e-b.s))<0;
52 }
53 void HPI(line L[],int n,point res[],int &resn)
54 {
55     int tot=n;
56     sort(L,L+n,HPICmp);
57     tot=1;
58     for(int i=1;i<n;i++)
59         if(fabs(L[i].k-L[i-1].k)>eps)
60             L[tot++]=L[i];
61     int head=0,tail=1;
62     Q[0]=L[0];
63     Q[1]=L[1];
64     resn=0;
65     for(int i=2;i<tot;i++)
66     {
67         if(fabs((Q[tail].e-Q[tail].s)*(Q[tail-1].e-Q[tail-1].s))<eps
68             || fabs((Q[head].e-Q[head].s)*(Q[head+1].e-Q[head+1].s))<eps)
69             return;
70         while(head<tail&&(((Q[tail]&Q[tail-1])-L[i].s)*(L[i].e-L[i].s))>eps)
71             tail--;
72         while(head<tail&&(((Q[head]&Q[head+1])-L[i].s)*(L[i].e-L[i].s))>eps)
73             head++;
74         Q[++tail]=L[i];
75     }
76     while(head<tail&&(((Q[tail]&Q[tail-1])-Q[head].s)*(Q[head].e-Q[head].s))>eps)
77         tail--;
78     while(head<tail&&(((Q[head]&Q[head+1])-Q[tail].s)*(Q[tail].e-Q[tail].s))>eps)
79         head++;
80     if(tail<=head+1)
81         return;
82     for(int i=head;i<tail;i++)
83         res[resn++]=Q[i]&Q[i+1];
84     if(head<tail+1)
85         res[resn++]=Q[head]&Q[tail];
86 }
87 int main()
88 {
89     // fop;
90     int t;

```

```

91 scanf("%d",&t);
92 while(t--)
93 {
94     int n;
95     scanf("%d",&n);
96     point P[1011];
97     for(int i=0;i<n;i++)
98         scanf("%lf%lf",&P[i].x,&P[i].y);
99     int dir=0;
100    double sum=0;
101    for(int i=0;i<n;i++)
102        sum+=xmult(P[i],P[(i+1)%n],P[(i+2)%n]);
103    if(sum<0) dir=0;
104    else dir=1;
105    line LL[1011];
106    if(dir)
107        for(int i=0;i<n;i++)
108            LL[i]=line(P[i],P[(i+1)%n]);
109    else for(int i=n-1;i>=0;i--)
110        LL[i]=line(P[i],P[(i-1+n)%n]);
111    point res[1011];
112    int resn=0;
113    HPI(LL,n,res,resn);
114    puts(resn?"YES":"NO");
115 }
116 }

```

## 1.11 圆面积并

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const double eps=1e-12;
4  const double pi=acos(-1.0);
5  struct point
6  {
7      double x,y;
8      point(double _x=0,double _y=0)
9      {
10         x=_x;
11         y=_y;
12     };
13     point(point s,point e)
14     {
15         x=e.x-s.x;
16         y=e.y-s.y;
17     };
18     double length()
19     {
20         return sqrt(x*x+y*y);
21     }
22 };
23 struct circle
24 {
25     point c;
26     double r;
27 };
28 struct event
29 {
30     double tim;
31     int typ;
32     event(double _tim=0,int _typ=0)
33     {
34         tim=_tim;
35         typ=_typ;
36     }
37 };
38 int cmp(const double &a,const double &b)
39 {
40     if(fabs(a-b)<eps)
41         return 0;
42     if(a<b)
43         return -1;
44     return 1;
45 }
46 bool eventcmp(const event &a,const event &b)
47 {
48     return cmp(a.tim,b.tim)<0;
49 }
50 double area(double theta,double r)
51 {

```

```

52     return 0.5*r*r*(theta-sin(theta));
53 }
54 double xmult(point a,point b)
55 {
56     return a.x*b.y-a.y*b.x;
57 }
58 int n,cur,tote;
59 circle c[1001];
60 double ans[1011],pre[1011],AB,AC,BC,theta,fai,a0,a1;
61 event e[4004];
62 point lab;
63 int N;
64 bool del[1100];
65 void calc()
66 {
67     for(int i=0;i<n;i++)
68         del[i]=0;
69     for(int i=0;i<n;i++)
70         if(del[i]==0)
71         {
72             for(int j=0;j<n;j++)
73                 if(i!=j)
74                 {
75                     if(del[j]==0)
76                         if(cmp(point(c[i].c,c[j].c).length()+c[j].r,c[i].r)<=0)
77                             del[j]=1;
78                 }
79         }
80     int tn=n;
81     n=0;
82     for(int i=0;i<tn;i++)
83         if(del[i]==0)
84             c[n++]=c[i];
85     for(int i=1;i<=tn;i++)
86         ans[i]=0.0;
87     for(int i=0;i<n;i++)
88     {
89         tote=0;
90         e[tote++]=event(-pi,1);
91         e[tote++]=event(pi,-1);
92         for(int j=0;j<n;j++)
93             if(j!=i)
94             {
95                 lab=point(c[j].c.x-c[i].c.x,c[j].c.y-c[i].c.y);
96                 AB=lab.length();
97                 AC=c[i].r;
98                 BC=c[j].r;
99                 if(cmp(AB+AC,BC)<=0)
100                 {
101                     e[tote++]=event(-pi,1);
102                     e[tote++]=event(pi,-1);
103                     continue;
104                 }
105                 if(cmp(AB+BC,AC)<=0)
106                     continue;
107                 if(cmp(AB,AC+BC)>0)
108                     continue;
109                 theta=atan2(lab.y,lab.x);
110                 fai=acos((AC*AC+AB*AB-BC*BC)/(2.0*AC*AB));
111                 a0=theta-fai;
112                 if(cmp(a0,-pi)<0) a0+=2*pi;
113                 a1=theta+fai;
114                 if(cmp(a1,pi)>0) a1-=2*pi;
115                 if(cmp(a0,a1)>0)
116                 {
117                     e[tote++]=event(a0,1);
118                     e[tote++]=event(pi,-1);
119                     e[tote++]=event(-pi,1);
120                     e[tote++]=event(a1,-1);
121                 }
122                 else
123                 {
124                     e[tote++]=event(a0,1);
125                     e[tote++]=event(a1,-1);
126                 }
127             }
128     }
129     sort(e,e+tote,eventcmp);
130     cur=0;
131     for(int j=0;j<tote;j++)
132     {

```



```

132         if(cur!=0&&cmp(e[j].tim,pre[cur])!=0)
133         {
134             ans[cur]+=area(e[j].tim-pre[cur],c[i].r);
135             ans[cur]+=xmult(point(c[i].c.x+c[i].r*cos(pre[cur]),c[i].c.y+c[i].r*sin(pre[cur])),
136                             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;
137         }
138         cur+=e[j].typ;
139         pre[cur]=e[j].tim;
140     }
141 }
142 printf("%.3f\n",ans[1]);
143 for(int i=1;i<n;i++)
144     ans[i]-=ans[i+1];
145 }
146 int main()
147 {
148     while(scanf("%d",&n)>0)
149     {
150         for(int i=0;i<n;i++)
151         {
152             scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
153         }
154         calc();
155     }
156 }

```

## 1.12 圆与多边形交

```

1  #include <bits/stdc++.h>
2  #define pi acos(-1.0)
3  using namespace std;
4  const double eps=1e-10;
5  inline double max(double a,double b)
6  {
7      if(a>b)
8          return a;
9      return b;
10 }
11 inline double min(double a,double b)
12 {
13     if(a>b)
14         return b;
15     return a;
16 }
17 inline int fi(double a)
18 {
19     if(a>eps)
20         return 1;
21     else if(a>=-eps)return 0;
22     return -1;
23 }
24 class vector
25 {
26 public:
27     double x,y;
28     vector(){}
29     vector(double x0,double y0)
30     {
31         x=x0,y=y0;
32     }
33     double operator *(const vector& a) const
34     {
35         return x*a.y-y*a.x;
36     }
37     double operator %(const vector& a) const
38     {
39         return x*a.x+y*a.y;
40     }
41     vector verti() const
42     {
43         return vector(-y,x);
44     }
45     double length() const
46     {
47         return sqrt(x*x+y*y);
48     }
49     vector adjust(double len)
50     {
51         double o1=len/length();
52         return vector(x*o1,y*o1);

```

```

53     }
54     vector oppose()
55     {
56         return vector(-x,-y);
57     }
58 };
59 class point
60 {
61 public:
62     double x,y;
63     point (){}
64     point (double x0,double y0)
65     {
66         x=x0,y=y0;
67     }
68     vector operator -(const point& a) const
69     {
70         return vector(x-a.x,y-a.y);
71     }
72     point operator +(const vector& a) const
73     {
74         return point(x+a.x,y+a.y);
75     }
76 };
77 class segment
78 {
79 public:
80     point a,b;
81     segment(){}
82     segment(point a0,point b0)
83     {
84         a=a0,b=b0;
85     }
86     point intersert(const segment& s) const
87     {
88         vector v1=s.a-a,v2=s.b-a,v3=s.b-b,v4=s.a-b;
89         double s1=v1*v2,s2=v3*v4;
90         double se=s1+s2;
91         s1/=se;
92         s2/=se;
93         return point(a.x*s2+b.x*s1,a.y*s2+b.y*s1);
94     }
95     point pverti(const point& p) const
96     {
97         vector t=(b-a).verti();
98         segment uv(p,p+t);
99         return intersert(uv);
100     }
101     bool on_seg(const point &p) const
102     {
103         if(fi(min(a.x,b.x)-p.x)<=0&&fi(p.x-max(a.x,b.x)<=0&&
104             fi(min(a.y,b.y)-p.y)<=0&&fi(p.y-max(a.y,b.y)<=0))return true;
105         else return false;
106     }
107 };
108 double radius;
109 point polygon[10];
110 double kuras_area(point a,point b,point cir)
111 {
112     point ori=point(cir.x,cir.y);
113     // printf("%.2f %.2f\n",cir.x,cir.y);
114     int sgn=fi((b-ori)*(a-ori));
115     double da=(a-ori).length(),db=(b-ori).length();
116     // printf("%.2f %.2f\n",da,db);
117     int ra=fi(da-radius),rb=fi(db-radius);
118     double angle = acos(((b-ori)*(a-ori))/(da*db));
119     // printf("%.2f\n",angle);
120     segment t(a,b); point h,u; vector seg;
121     double ans,dlt,mov,tangle;
122     if(fi(da)==0||fi(db)==0)
123         return 0;
124     else if(sgn==0)
125         return 0;
126     else if(ra<=0&&rb<=0)
127         return fabs((b-ori)*(a-ori))/2*sgn;
128     else if(ra>=0&&rb>=0)
129     {
130         h=t.pverti(ori);
131         dlt=(h-ori).length();
132         if(!t.on_seg(h)||fi(dlt-radius)>=0)

```

```

133     return radius*radius*(angle/2)*sgn;
134 else
135 {
136     ans=radius*radius*(angle/2);
137     tangle=acos(dlt/radius);
138     ans-=radius*radius*tangle;
139     ans+=radius*sin(tangle)*dlt;
140     // printf("%.2f\n",ans);
141     return ans*sgn;
142 }
143 }
144 else
145 {
146     h=t.pverti(ori);
147     dlt=(h-ori).length();
148     seg=b-a;
149     mov=sqrt(radius*radius-dlt*dlt);
150     seg=seg.adjust(mov);
151     if(t.on_seg(h+seg)) u=h+seg;
152     else u=h+seg.oppose();
153     if(ra==1) swap(a,b);
154     ans=fabs((a-ori)*(u-ori))/2;
155     tangle=acos(((u-ori)*(b-ori))/((u-ori).length()*(b-ori).length()));
156     ans+=radius*radius*(tangle/2);
157     return ans*sgn;
158 }
159 }
160 int main()
161 {
162     int cas=0;
163     double x1,x2,x3,y1,y2,y3;
164     while(scanf("%lf%lf%lf",&x1,&y1,&radius)>0)
165     {
166         if(cas++)
167             puts("");
168         scanf("%lf%lf%lf%lf",&x2,&y2,&x3,&y3);
169         polygon[0]=point(x2,y2);
170         polygon[3]=point(x2,y3);
171         polygon[2]=point(x3,y3);
172         polygon[1]=point(x3,y2);
173         double area=0;
174         for(int i=0;i<4;i++)
175         {
176             area+=kuras_area(polygon[i],polygon[(i+1)%4],point(x1,y1));
177         }
178         printf("%.16f\n",fabs(area));
179     }
180 }

```

## 1.13 三维凸包

```

1  #define PR 1e-8
2  #define N 510
3  struct TPoint
4  {
5      double x,y,z;
6      TPoint(){}
7      TPoint(double _x,double _y,double _z):x(_x),y(_y),z(_z){}
8      TPoint operator-(const TPoint p) {return TPoint(x-p.x,y-p.y,z-p.z);}
9      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
10     double operator^(const TPoint p) {return x*p.x+y*p.y+z*p.z;} //点积
11 };
12 struct fac//
13 {
14     int a,b,c;//凸包一个面上的三个点的编号
15     bool ok;//该面是否是最终凸包中的面
16 };
17 struct T3dhull
18 {
19     int n;//初始点数
20     TPoint ply[N];//初始点
21     int trianglecnt;//凸包上三角形数
22     fac tri[N];//凸包三角形
23     int vis[N][N];//点到点是属于哪个面ij
24     double dist(TPoint a){return sqrt(a.x*a.x+a.y*a.y+a.z*a.z);} //两点长度
25     double area(TPoint a,TPoint b,TPoint c){return dist((b-a)*(c-a));} //三角形面积*2
26     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
27     double ptoplane(TPoint &p,fac &f)//正：点在面同向

```

```

28 {
29     TPoint m=ply[f.b]-ply[f.a],n=ply[f.c]-ply[f.a],t=p-ply[f.a];
30     return (m*n)^t;
31 }
32 void deal(int p,int a,int b)
33 {
34     int f=vis[a][b];
35     fac add;
36     if(tri[f].ok)
37     {
38         if((ptoplane(ply[p],tri[f]))>PR) dfs(p,f);
39         else
40         {
41             add.a=b,add.b=a,add.c=p,add.ok=1;
42             vis[p][b]=vis[a][p]=vis[b][a]=trianglecnt;
43             tri[trianglecnt++]=add;
44         }
45     }
46 }
47 void dfs(int p,int cnt)//维护凸包, 如果点在凸包外更新凸包p
48 {
49     tri[cnt].ok=0;
50     deal(p,tri[cnt].b,tri[cnt].a);
51     deal(p,tri[cnt].c,tri[cnt].b);
52     deal(p,tri[cnt].a,tri[cnt].c);
53 }
54 bool same(int s,int e)//判断两个面是否为同一面
55 {
56     TPoint a=ply[tri[s].a],b=ply[tri[s].b],c=ply[tri[s].c];
57     return fabs(volume(a,b,c,ply[tri[e].a]))<PR
58         &&fabs(volume(a,b,c,ply[tri[e].b]))<PR
59         &&fabs(volume(a,b,c,ply[tri[e].c]))<PR;
60 }
61 void construct();//构建凸包
62 {
63     int i,j;
64     trianglecnt=0;
65     if(n<4) return ;
66     bool tmp=true;
67     for(i=1;i<n;i++)//前两点不共点
68     {
69         if((dist(ply[0]-ply[i]))>PR)
70         {
71             swap(ply[1],ply[i]); tmp=false; break;
72         }
73     }
74     if(tmp) return;
75     tmp=true;
76     for(i=2;i<n;i++)//前三点不共线
77     {
78         if((dist((ply[0]-ply[1])*(ply[1]-ply[i])))>PR)
79         {
80             swap(ply[2],ply[i]); tmp=false; break;
81         }
82     }
83     if(tmp) return ;
84     tmp=true;
85     for(i=3;i<n;i++)//前四点不共面
86     {
87         if(fabs((ply[0]-ply[1])*(ply[1]-ply[2])^(ply[0]-ply[i]))>PR)
88         {
89             swap(ply[3],ply[i]); tmp=false; break;
90         }
91     }
92     if(tmp) return ;
93     fac add;
94     for(i=0;i<4;i++)//构建初始四面体
95     {
96         add.a=(i+1)%4,add.b=(i+2)%4,add.c=(i+3)%4,add.ok=1;
97         if((ptoplane(ply[i],add))>0) swap(add.b,add.c);
98         vis[add.a][add.b]=vis[add.b][add.c]=vis[add.c][add.a]=trianglecnt;
99         tri[trianglecnt++]=add;
100     }
101     for(i=4;i<n;i++)//构建更新凸包
102     {
103         for(j=0;j<trianglecnt;j++)
104         {
105             if(tri[j].ok&&(ptoplane(ply[i],tri[j]))>PR)
106             {

```

```

107         dfs(i,j); break;
108     }
109 }
110 }
111 int cnt=trianglecnt;
112 trianglecnt=0;
113 for(i=0;i<cnt;i++)
114 {
115     if(tri[i].ok)
116         tri[trianglecnt++]=tri[i];
117 }
118 }
119 double area()//表面积
120 {
121     double ret=0;
122     for(int i=0;i<trianglecnt;i++)
123         ret+=area(ply[tri[i].a],ply[tri[i].b],ply[tri[i].c]);
124     return ret/2.0;
125 }
126 double volume()//体积
127 {
128     TPoint p(0,0,0);
129     double ret=0;
130     for(int i=0;i<trianglecnt;i++)
131         ret+=volume(p,ply[tri[i].a],ply[tri[i].b],ply[tri[i].c]);
132     return fabs(ret/6);
133 }
134 int facetri() {return trianglecnt;}//表面三角形数
135 int facepolygon()//表面多边形数
136 {
137     int ans=0,i,j,k;
138     for(i=0;i<trianglecnt;i++)
139     {
140         for(j=0,k=1;j<i;j++)
141         {
142             if(same(i,j)) {k=0;break;}
143         }
144         ans+=k;
145     }
146     return ans;
147 }
148 }hull;

```

## 2 Graph

### 2.1 Tarjan

```

1  const int MAX=1605000;
2  struct node
3  {
4      int v,next,w;
5  }g[MAX],g2[MAX];
6  int adj[MAX],low[MAX],dfn[MAX],bel[MAX];
7  int e,Index,cnt,n,m,x[MAX],y[MAX],inStack[MAX];
8  int maxx,vis[MAX],e2,adj2[MAX];
9  stack<int>s;
10 void add(int u,int v)
11 {
12     g[e].v=v; g[e].next=adj[u]; adj[u]=e++;
13 }
14 void tarjan(int u)
15 {
16     low[u]=dfn[u]=++Index;
17     s.push(u);
18     inStack[u]=1;
19     int i,v;
20     for(i=adj[u];i!=-1;i=g[i].next)
21     {
22         v=g[i].v;
23         if(!dfn[v])
24         {
25             tarjan(v);
26             low[u]=min(low[u],low[v]);
27         }
28         else if(inStack[v])
29             low[u]=min(low[u],dfn[v]);
30     }
31     if(low[u]==dfn[u])
32     {
33         cnt++;
34         do
35         {
36             v=s.top();
37             s.pop();
38             inStack[v]=0;
39             bel[v]=cnt;
40         }while(u!=v);
41     }
42 }

```

### 2.2 Sap

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  int m;
4  struct Edge
5  {
6      int en,cap,flow,next;
7  } edge[E];
8  int head[N] , tot , now[N];
9  int source,sink,tot_num;
10 int pre[N] , dis[N] , gap[N];
11 int n;
12 struct node
13 {
14     int x,y,z,f,l;
15 }p[122];
16 void add(int st,int en,int cap)
17 {
18     edge[tot].en=en;
19     edge[tot].cap=cap;
20     edge[tot].flow=0;
21     edge[tot].next=head[st];
22     head[st]=tot++;
23
24     edge[tot].en=st;
25     edge[tot].cap=0;
26     edge[tot].flow=0;
27     edge[tot].next=head[en];
28     head[en]=tot++;
29 }

```

```

30
31 void augment(int flow)
32 {
33     for(int i=source;i!=sink;i=edge[now[i]].en)
34     {
35         edge[now[i]].flow+=flow;
36         edge[now[i]^1].flow-=flow;
37     }
38 }
39 int summ=0;
40 int sap()
41 {
42     memset(dis,0,sizeof(dis));
43     memset(gap,0,sizeof(gap));
44     memset(pre,-1,sizeof(pre));
45     for(int i=0;i<tot_num;i++)
46         now[i]=head[i];
47     gap[0]=tot_num;
48     int point=source,flow=0,min_flow=INT_INF;
49     while(dis[source]<tot_num)
50     {
51         bool fg=false;
52         for(int i=now[point];i!=-1;i=edge[i].next)
53             if(edge[i].cap-edge[i].flow>0 && dis[point]==dis[edge[i].en]+1)
54             {
55                 min_flow=min(min_flow,edge[i].cap-edge[i].flow);
56                 now[point]=i;
57                 pre[edge[i].en]=point;
58                 point=edge[i].en;
59                 if(point==sink)
60                 {
61                     flow+=min_flow;
62                     augment(min_flow);
63                     point=source;
64                     min_flow=INT_INF;
65                 }
66                 fg=true;
67                 break;
68             }
69         if(fg) continue;
70         if(--gap[dis[point]]==0) break;
71         int Min=tot_num;
72         for(int i=head[point];i!=-1;i=edge[i].next)
73             if(edge[i].cap-edge[i].flow>0 && Min>dis[edge[i].en])
74             {
75                 Min=dis[edge[i].en];
76                 now[point]=i;
77             }
78         gap[dis[point]=Min+1]++;
79         if(point!=source) point=pre[point];
80     }
81     return flow;
82 }
83 int main()
84 {
85     while(scanf("%d%d",&m,&n)>0)
86     {
87         tot=0;
88         source=0;
89         sink=n-1;
90         tot_num=n;
91         memset(head,-1,sizeof(head));
92         for(int i=0;i<m;i++)
93         {
94             int a,b,c;
95             scanf("%d%d%d",&a,&b,&c);
96             add(a-1,b-1,c);
97         }
98         printf("%d\n",sap());
99     }
100 }

```

## 2.3 费用流

```

1 using namespace std;
2 struct Edge
3 {
4     int st,en,cap,flow,cost,next;
5 } edge[E];
6 int head[N] , tot , now[N];

```

```

7  int source,sink;
8  int pre[N] , dis[N];
9  queue<int> q;
10 bool vs[N];
11
12 void add(int st,int en,int cap,int cost)
13 {
14     edge[tot].st=st;
15     edge[tot].en=en;
16     edge[tot].cap=cap;
17     edge[tot].flow=0;
18     edge[tot].cost=cost;
19     edge[tot].next=head[st];
20     head[st]=tot++;
21
22     edge[tot].st=en;
23     edge[tot].en=st;
24     edge[tot].cap=0;
25     edge[tot].flow=0;
26     edge[tot].cost=-cost;
27     edge[tot].next=head[en];
28     head[en]=tot++;
29 }
30
31 bool SPFA()
32 {
33     for(int i=0; i<N; i++)
34         dis[i]=INF;
35     memset(vs,0,sizeof(vs));
36     memset(now,-1,sizeof(now));
37     while(!q.empty()) q.pop();
38     q.push(source); dis[source]=0; vs[source]=1;
39     while(!q.empty())
40     {
41         int u=q.front(); q.pop(); vs[u]=0;
42         for(int i=head[u],v; i!=-1; i=edge[i].next)
43             if(edge[i].cap-edge[i].flow>0 && dis[v=edge[i].en]>dis[u]+edge[i].cost)
44             {
45                 dis[v]=dis[u]+edge[i].cost;
46                 now[v]=i;
47                 if(!vs[v])
48                 {
49                     vs[v]=1;
50                     q.push(v);
51                 }
52             }
53     }
54     if(dis[sink]!=INF) return true;
55     else return false;
56 }
57
58 int MCMF()
59 {
60     int cost=0;
61     while(SPFA())
62     {
63         int flow=INF;
64         for(int u=sink; u!=source; u=edge[now[u]].st)
65             if(flow>edge[now[u]].cap-edge[now[u]].flow)
66                 flow=edge[now[u]].cap-edge[now[u]].flow;
67         for(int u=sink; u!=source; u=edge[now[u]].st)
68         {
69             edge[now[u]].flow+=flow;
70             edge[now[u]^1].flow-=flow;
71         }
72         cost+=flow*dis[sink];
73     }
74     return cost;
75 }
76 int n,m;
77 struct node
78 {
79     int x;
80     int y;
81     node(int x=0,int y=0){this->x=x,this->y=y;};
82 };
83 node ph[111],pm[111];
84 char maz[111][111]={0};
85 int dist[111][111]={0};
86 void build()

```



```

87 {
88     clr_1(head);
89     tot=0;
90     clr(dist);
91     int cnt1=0;
92     int cnt2=0;
93     source=0; sink=(n+m)+1;
94     for(int i=0;i<n;i++)
95     {
96         scanf("%s",maz[i]);
97         for(int j=0;j<m;j++)
98         {
99             if(maz[i][j]=='H')
100             {
101                 ph[++cnt1]=node(i,j);
102                 add(0,cnt1,1,0);
103             }
104             else if(maz[i][j]=='m')
105             {
106                 pm[++cnt2]=node(i,j);
107             }
108         }
109     }
110     sink=cnt1+cnt2+1;
111     for(int i=1;i<=cnt2;i++)
112         add(cnt1+i,sink,1,0);
113     for(int i=1;i<=cnt1;i++)
114     {
115         for(int j=1;j<=cnt2;j++)
116         {
117             int go=abs(ph[i].x-pm[j].x)+abs(ph[i].y-pm[j].y);
118             add(i,cnt1+j,INF,go);
119         }
120     }
121     printf("%d\n",MCMF());
122     return;
123 }
124
125 int main()
126 {
127     // fop;
128     while(scanf("%d%d",&n,&m)!=EOF)
129     {
130         if(n==m&&m==0)
131             break;
132         build();
133     }
134     return 0;
135 }

```

## 3 字符串

### 3.1 后缀数组 dc3

```

1  const int maxn = 150000;
2  int s[maxn*3];
3  int rank[maxn*3], height[maxn*3];
4  int r[maxn*3], sa[maxn*3];
5  int wa[maxn*3], wb[maxn*3], wv[maxn*3], ss[maxn*3], sa2[maxn*3];
6  int n, m, k;
7
8  #define F(x) ((x)/3+((x)%3==1?0:tb))
9  #define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
10
11 int c0(int *r, int a, int b){
12     return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];
13 }
14 int c12(int k, int *r, int a, int b){
15     if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1, r, a+1, b+1);
16     else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
17 }
18 void sort(int *r, int *a, int *b, int n, int m){
19     int i;
20     for(i=0; i<n; i++) wv[i]=r[a[i]];
21     for(i=0; i<m; i++) ss[i]=0;
22     for(i=0; i<n; i++) ss[wv[i]]++;
23     for(i=1; i<m; i++) ss[i]+=ss[i-1];
24     for(i=n-1; i>=0; i--) b[ss[wv[i]]]=a[i];
25     return;
26 }
27 //用n+1
28 void dc3(int *r, int *sa, int n, int m){
29     int i, j, *san=sa+n, ta=0, tb=(n+1)/3, tbc=0, p, *rn=r+n;
30     r[n]=r[n+1]=0;
31     for(i=0; i<n; i++) if(i%3!=0) wa[tbc++]=i;
32     sort(r+2, wa, wb, tbc, m);
33     sort(r+1, wb, wa, tbc, m);
34     sort(r, wa, wb, tbc, m);
35     for(p=1, rn[F(wb[0])]=0, i=1; i<tbc; i++)
36         rn[F(wb[i])]=c0(r, wb[i-1], wb[i])?p-1:p++;
37     if(p<tbc) dc3(rn, san, tbc, p);
38     else for(i=0; i<tbc; i++) san[rn[i]]=i;
39     for(i=0; i<tbc; i++) if(san[i]<tb) wb[ta++]=san[i]*3;
40     if(n%3==1) wb[ta++]=n-1;
41     sort(r, wb, wa, ta, m);
42     for(i=0; i<tbc; i++) wv[wb[i]]=G(san[i]);
43     for(i=0, j=0, p=0; i<ta && j<tbc; p++)
44         sa[p]=c12(wb[j]%3, r, wa[i], wb[j])?wa[i++]:wb[j++];
45     for(; i<ta; p++) sa[p]=wa[i++];
46     for(; j<tbc; p++) sa[p]=wb[j++];
47     return;
48 }
49 //用n+1
50 void calheight(int *r, int *sa, int n){
51     int i, j, k=0;
52     for (i=0; i<n; ++i) rank[sa[i]]=i;
53     for (i=0; i<n; height[rank[i++]]=k)
54         for (k?k--:0, j=sa[rank[i]-1]; r[i+k]==r[j+k]; k++);
55 }

```

### 3.2 后缀数组 da

```

1  const int maxn=100000;
2  int sa[maxn], rank[maxn], rank2[maxn], height[maxn], c[maxn], *x, *y, s[maxn], n;
3  void radix_sort(int n, int sz)
4  {
5      memset(c, 0, sizeof(c));
6      for(int i=0; i<n; i++) c[x[y[i]]]++;
7      for(int i=1; i<sz; i++) c[i]+=c[i-1];
8      for(int i=n-1; i>=0; i--) sa[c[x[y[i]]]]=y[i];
9  }
10 void get_sa(int *s, int n, int sz=222)
11 {
12     x=rank, y=rank2;
13     for(int i=0; i<n; i++)
14         x[i]=s[i], y[i]=i;
15     radix_sort(n, sz);
16     for(int len=1; len<n; len<=1)

```

```

17 {
18     int yid=0;
19     for(int i=n-len;i<n;i++) y[yid++]=i;
20     for(int i=0;i<n;i++) if(sa[i]>=len) y[yid++]=sa[i]-len;
21     radix_sort(n,sz);
22     swap(x,y);
23     x[sa[0]]=yid=0;
24     for(int i=1;i<n;i++)
25     {
26         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])
27             x[sa[i]]=yid;
28         else x[sa[i]]=++yid;
29     }
30     sz=yid+1;
31     if(sz>=n) break;
32 }
33 for(int i=0;i<n;i++)
34     rank[i]=x[i];
35 }
36 void get_height(int * s,int n)
37 {
38     int k=0;height[0]=0;
39     for(int i=0;i<n;i++)
40     {
41         if(rank[i]==0) continue;
42         k=max(0,k-1);
43         int j=sa[rank[i]-1];
44         while(i+k<n&&j+k<n&&s[i+k]==s[j+k]) k++;
45         height[rank[i]]=k;
46     }
47 }

```

### 3.3 后缀自动机

```

1 #include <set>
2 #include <cmath>
3 #include <stack>
4 #include <cstdio>
5 #include <cstring>
6 #include <iostream>
7 #include <algorithm>
8 #include <cstdlib>
9 #include <numeric>
10 #include <vector>
11 #include <ctime>
12 #include <queue>
13 #include <list>
14 #include <map>
15 #define pi acos(-1.0)
16 #define INF 0x3f3f3f3f
17 #define clr(x) memset(x,0,sizeof(x));
18 #define clrto(x,siz,y) for(int xx=0;xx<=siz;xx++) x[xx]=y;
19 #define clrset(x,siz) for(int xx=0;xx<=siz;xx++) x[xx]=xx;
20 #define clr_1(x) memset(x,-1,sizeof(x));
21 #define clrmax(x) memset(x,0x3f,sizeof(x));
22 #define clrvec(x,siz) for(int xx=0;xx<=siz;xx++) x[xx].clear();
23 #define fop2 freopen(".in","r",stdin); //freopen(".out","w",stdout);
24 #define fop freopen("in.txt","r",stdin); //freopen("out.txt","w",stdout);
25 #define myprogram By_135678942570
26 #define clrcpy(x,siz,y) for(int xx=0;xx<siz;xx++) x[xx]=y[xx];
27 #define pb push_back
28 using namespace std;
29 const int INF = (-1u) >> 2;
30 const int maxn = 200010;
31 const int kind = 27;
32 struct NODE{
33     int fail, next[kind], step;
34     int pos; //该节点在字符串中的位置
35     int id; //副本与本体拥有相同的 id
36     void clear(){
37         clr(next);
38         id = step = fail = pos = 0;
39     }
40 };
41 struct suffix_automation{ //支持删除的后缀自动机 !!!
42     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
43     int last, total;
44     int len; //当前字符串长度
45     int idx[maxn << 1]; //字符串中第 i 个字符对应结点 idx[i]

```

```

46  bool is_del[maxn << 1];
47  void init(){
48      total = last = 0;
49      len = 0;
50      node[0].clear();
51      is_del[0] = 0;
52      clr(idx);
53  }
54  void push_back(int step_val){
55      int p = ++total;
56      node[p].clear();
57      node[p].step = step_val;
58      node[p].id = p;
59      is_del[p] = 0;
60  }
61  bool unExistNorDel(int x){
62      return x == 0 || is_del[node[x].id];
63  }
64  void insert(int ch){
65      push_back(node[last].step + 1);
66      node[total].pos = ++len; idx[len] = total; // 第 len 个字符对应第 total 个结点
67      int p = last, np = total;
68      for (; unExistNorDel(node[p].next[ch]); p = node[p].fail) node[p].next[ch] = np;
69      if (node[p].next[ch] == np) //此时一定等于p 0
70          node[np].fail = p;
71      else {
72          int q = node[p].next[ch];
73          if (node[q].step == node[p].step + 1) node[np].fail = q;
74          else {
75              int nq = ++total;
76              node[nq] = node[q];
77              node[nq].step = node[p].step + 1;
78              node[q].fail = node[np].fail = nq;
79              for (; node[p].next[ch] == q; p = node[p].fail) node[p].next[ch] = nq;
80          }
81      }
82      last = np;
83  }
84  void addString(char *str){
85      int i;
86      for (i = 0; str[i]; i++)
87          insert(str[i] - 'a' + 1);
88  }
89  void delString(int k){ //从后往前删除 个字符k
90      int i;
91      for (i = 1; i <= k; i++){
92          is_del[node[idx[len - i + 1]].id] = 1;
93      }
94      last = idx[len - k];
95      len -= k;
96  }
97  bool first;
98  int dfs(int x, int &l){
99      if (l == 0) return node[x].pos;
100     int i;
101     for (i = 0; i < kind; i++){
102         if (i == 0 && first){
103             first = 0;
104             continue;
105         }
106         if (unExistNorDel(node[x].next[i])) continue;
107         l--;
108         return dfs(node[x].next[i], l);
109     }
110     return node[x].pos;
111 }
112 int query(int l){ //求长度为l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l len
113     first = 1;
114     insert(0);
115     int tmp = l;
116     int id = dfs(0, tmp);
117     delString(1);
118     int det = l - tmp;
119     return id - det + 1;
120 }
121 }obj;
122 int main()
123 {
124
125 }

```

## 4 杂物

### 4.1 Dancing Links

```

1  #include<bits/stdc++.h>
2  #define clr(x)  memset(x,0,sizeof(x));
3  using namespace std;
4  int U[333333];
5  int D[333333];
6  int L[333333];
7  int R[333333];
8  int S[333333];
9  int H[333333];
10 int C[333333];
11 int res[333333];
12 char mp[22][22];
13 void del(int CC)
14 {
15     R[L[CC]]=R[CC];
16     L[R[CC]]=L[CC];
17     for(int i=D[CC];i!=CC;i=D[i])
18         for(int j=R[i];j!=i;j=R[j])
19             {
20                 D[U[j]]=D[j];
21                 U[D[j]]=U[j];
22                 S[C[j]]--;
23             }
24 }
25 void bac(int CC)
26 {
27     R[L[CC]]=CC;
28     L[R[CC]]=CC;
29     for(int i=D[CC];i!=CC;i=D[i])
30         for(int j=R[i];j!=i;j=R[j])
31             {
32                 D[U[j]]=j;
33                 U[D[j]]=j;
34                 S[C[j]]++;
35             }
36 }
37 int dfs(int dep)
38 {
39     if(R[0]==0)
40     {
41         for(int i=0;i<dep;i++)
42             mp[res[i]/256][res[i]%256/16]=res[i]%16+'A';
43         return 1;
44     }
45     if(R[0]>256)
46         return 0;
47     int w=INF;
48     int now;
49     int nxt;
50     for(int i=R[0];i!=0;i=R[i])
51         if(S[i]<w)
52         {
53             now=i;
54             w=S[i];
55         }
56     nxt=D[now];
57     del(now);
58     while(nxt!=now)
59     {
60         res[dep]=H[nxt];
61         for(int i=R[nxt];i!=nxt;i=R[i])
62             del(C[i]);
63         if(dfs(dep+1))
64             return 1;
65         for(int i=L[nxt];i!=nxt;i=L[i])
66             bac(C[i]);
67         nxt=D[nxt];
68     }
69     bac(now);
70     return 0;
71 }
72 int last;
73 int cnt=256*4;
74 void linkinit(int u,int v)

```

```

75 {
76     L[v]=u,R[u]=v,D[v]=U[v]=v,last=v;
77 }
78 void init()
79 {
80     last=0;
81     for(int i=1;i<=256;i++)
82     {
83         if(mp[(i-1)/16][(i-1)%16]!='-')
84             continue;
85         linkinit(last,i);
86     }
87     for(int i=1;i<=256;i++)
88     {
89         int flag=1;
90         int h=(i-1)/16;
91         int nn=(i-1)%16;
92         for(int ii=0;ii<=15;ii++)
93             if(mp[h][ii]==nn+'A')
94             {
95                 flag=0;
96                 break;
97             }
98         if(flag)
99             linkinit(last,i+256);
100     }
101     for(int i=1;i<=256;i++)
102     {
103         int flag=1;
104         int h=(i-1)/16;
105         int nn=(i-1)%16;
106         for(int ii=0;ii<=15;ii++)
107             if(mp[ii][h]==nn+'A')
108             {
109                 flag=0;
110                 break;
111             }
112         if(flag)
113             linkinit(last,i+256+256);
114     }
115     for(int i=1;i<=256;i++)
116     {
117         int flag=1;
118         int nx=((i-1)/16/4*4);
119         int ny=((i-1)/16%4*4);
120         int nn=(i-1)%16;
121         for(int ii=0;ii<=3;ii++)
122             for(int jj=0;jj<=3;jj++)
123                 if(mp[ii+nx][jj+ny]==nn+'A')
124                 {
125                     flag=0;
126                     ii=4;
127                     break;
128                 }
129         if(flag)
130             linkinit(last,i+256+256+256);
131     }
132     R[last]=0;
133     L[0]=last;
134 }
135 void linknode(int pos,int h)
136 {
137     cnt++;
138     H[cnt]=h;
139     C[cnt]=pos;
140     D[cnt]=C[cnt];
141     U[cnt]=U[C[cnt]];
142     D[U[C[cnt]]]=cnt;
143     U[C[cnt]]=cnt;
144     S[C[cnt]]++;
145 }
146 int main()
147 {
148     while(scanf("%s",mp[0])>0)
149     {
150         cnt=256*4;
151         clr(U);clr(D);clr(R);clr(L);clr(H);clr(C);clr(res);clr(S);
152         for(int i=1;i<16;i++)
153             scanf("%s",mp[i]);
154         init();

```

```

155     for(int i=0;i<16;i++)
156         for(int j=0;j<16;j++)
157             if(mp[i][j]=='-')
158                 for(int k=0;k<16;k++)
159                     {
160                         int flag=1;
161                         for(int ii=0;ii<16;ii++)
162                             if(mp[ii][j]=='A'+k||mp[i][ii]=='A'+k)
163                                 {
164                                     flag=0;
165                                     break;
166                                 }
167                         int nx=i/4*4;
168                         int ny=j/4*4;
169                         for(int ii=0;ii<4;ii++)
170                             for(int jj=0;jj<4;jj++)
171                                 if(mp[nx+ii][ny+jj]=='A'+k)
172                                     {
173                                         flag=0;
174                                         ii=5;
175                                         break;
176                                     }
177                         if(flag==0)
178                             continue;
179                         int h=i*256+j*16+k;
180                         int fg=0;
181                         if(D[i*16+j+1])
182                             {
183                                 linknode(i*16+j+1,h);
184                                 fg=cnt;
185                             }
186                         if(D[257+i*16+k])
187                             {
188                                 linknode(257+i*16+k,h);
189                                 if(fg) L[cnt]=cnt-1,R[cnt-1]=cnt;
190                                 else fg=cnt;
191                             }
192                         if(D[513+j*16+k])
193                             {
194                                 linknode(513+j*16+k,h);
195                                 if(fg) L[cnt]=cnt-1,R[cnt-1]=cnt;
196                                 else fg=cnt;
197                             }
198                         if(D[769+(i/4*4+j/4)*16+k])
199                             {
200                                 linknode(769+(i/4*4+j/4)*16+k,h);
201                                 if(fg) L[cnt]=cnt-1,R[cnt-1]=cnt;
202                                 else fg=cnt;
203                             }
204                         if(fg) L[fg]=cnt,R[cnt]=fg;
205                     }
206     dfs(0);
207     for(int i=0;i<16;i++)
208         {
209             mp[i][16]=0;
210             printf("%s\n",mp[i]);
211         }
212     puts("");
213 }
214 return 0;
215 }

```