ACM Template

UESTC_Flaghead

Salvare219

Last build at November 13, 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 三维几何

1.3.1 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;
```

```
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 subt(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  }
```

1.3.2 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  }
```

1.3.3 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  }
```

1.3.4 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  }
```

1.3.5 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 }
```

1.3.6 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 }
```

1.3.7 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))));
}
```

1.3.8 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;
}
```

1.3.9 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;
}
```

1.3.10 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;
}
```

1.3.11 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;
}
```

1.3.12 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;
}
```

1.3.13 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;
}
```

1.3.14 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;
}
```

1.3.15 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)));
}
```

1.3.16 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)));
}
```

1.3.17 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)));
}
```

1.3.18 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;
}
```

1.3.19 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
      );
}
```

1.3.20 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);
}
```

1.3.21 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);
}
```

1.3.22 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);
}
```

1.3.23 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;
}
```

1.3.24 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;
}
```

1.3.25 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;
}
```

1.3.26 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);
}
```

1.3.27 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));
}
```

1.3.28 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);
}
```

1.3.29 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));
}
```

1.3.30 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));
}
```

1.3.31 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));
}
```

1.4 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;
}
```

1.5 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;
```

```
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 };
```

```
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));
  }
}
```

2 Graph

2.1 二分图

2.1.1 最大匹配 Hungry

```
//O(n*m)
vector<int> gra[505];
int n;
bool vis[505];
int p[505];
bool find(int s)
{
  for(int i=0;i<gra[s].size();i++)
    if(vis[gra[s][i]]==0&&p[gra[s][i]]!=s)
    {
      vis[gra[s][i]]=1;
      if(p[gra[s][i]]==0 || find(p[gra[s][i]]))
      {
        p[gra[s][i]]=s;
        return 1;
      }
    }
  return 0;
}
int max_match()
{
  int ans=0;
  memset(p,0,sizeof(p));
  for(int i=1;i<=n;i++)
  {
    memset(vis,0,sizeof(vis));
    ans+=find(i);
  }
  return ans;
}

//找解
//删除该点 如果最大匹配数不变则不为最小覆盖集的点
vector<int>gra[15050];
int p[15050];
bool vis[15050];
int find(int s,int f)
{
    for(int i=0;i<gra[s].size();i++)
        if(vis[gra[s][i]]==0&&p[gra[s][i]]!=s)
        {
            vis[gra[s][i]]=1;
            if(p[gra[s][i]]==0||find(p[gra[s][i]],f))
            {
                f?p[gra[s][i]]=s:0;
                return 1;
            }
        }
    return 0;
}
void get_solution(int nn,int mm)
{
    for(int i=1;i<=nn;i++)
    {
        memset(vis,0,sizeof(vis));
        ans1+=find(i,1);
    }
    for(int i=1;i<=mm;i++)
        if(p[i])
        {
            memset(vis,0,sizeof(vis));
            if(find(p[i],0)==0)chy[i]=1;
        }
    for(int i=1;i<=nn;i++)
    {
        int f=1;
        for(int j=0;j<gra[i].size();j++)
            if(chy[gra[i][j]])
            {
                f=0;
                break;
            }
```

```
        chx[i]=f;
    }
}
```

2.1.2 最大匹配 HK

```
//O(m*n^0.5)  mm
vector<int>gra[505]; //初始化
int nn,mm;           // nn 左边点 mm 右边点
bool use[505];
int phx[505],phy[505],p[505];
int q[505];
bool relable()
{
    for(int i=1;i<=nn;i++)phx[i]=0;
    for(int i=1;i<=mm;i++)phy[i]=0;
    int tail=0,front=0,s,y;
    bool f=0;
    for(int i=1;i<=nn;i++)
        if(!use[i])q[tail++]=i;
    while(tail!=front)
    {
        s=q[front++];
        for(int i=0;i<gra[s].size();i++)
            if(!phy[y=gra[s][i]])
            {
                phy[y]=phx[s]+1;
                if(!p[y])f=1;
                else phx[p[y]]=phy[y]+1,q[tail++]=p[y];
            }
    }
    return f;
}
bool find(int s)
{
    int y;
    for(int i=0;i<gra[s].size();i++)
        if(phy[y=gra[s][i]]==phx[s]+1)
        {
            phy[y]=0;
            if(!p[y]||find(p[y]))return p[y]=s,1;
        }
    return 0;
}
int Hopcroft_Karp()
{
    int ans=0;
    memset(use,0,sizeof(use));
    memset(p,0,sizeof(p));
    while(relable())
        for(int i=1;i<=nn;i++)
            if(!use[i]&&find(i))use[i]=1,ans++;
    return ans;
}
```

2.1.3 最大权匹配

```
//O(n^3) 保证 nn <= mm 且必须有完备匹配建空点和空边
//从 1 开始标号
int nn,mm;
bool visx[105],visy[105];
int pre[105],gra[105][105];
int lx[105],ly[105],slack[105];
bool find(int s)
{
    visx[s]=1;
    for(int i=1;i<=mm;i++)
        if(visy[i]==0)
        {
            int f=lx[s]+ly[i]-gra[s][i];
            if(f==0)
            {
                visy[i]=1;
                if(pre[i]==0 || find(pre[i]))
                {
                    pre[i]=s;
                    return 1;
                }
            }
```

```
            else slack[i]=min(slack[i],f);
        }
    return 0;
}
int Kuhn_Munkras()
{
    int i,j,d,ans=0;
    memset(ly,0,sizeof(ly));
    memset(pre,0,sizeof(pre));
    for(i=1;i<=nn;i++)
        for(j=1,lx[i]=-0x7fffffff;j<=mm;j++)
            lx[i]=max(lx[i],gra[i][j]);
    for(i=1;i<=nn;i++)
    {
        memset(slack,0x7f,sizeof(slack));
        while(1)
        {
            memset(visx,0,sizeof(visx));
            memset(visy,0,sizeof(visy));
            if(find(i)) break;
            d=0x7fffffff;
            for(j=1;j<=mm;j++)
                if(visy[j]==0) d=min(d,slack[j]);
            for(j=1;j<=nn;j++)
                if(visx[j]) lx[j]-=d;
            for(j=1;j<=mm;j++)
                if(visy[j]) ly[j]+=d;
                else slack[j]-=d;
        }
    }
    for(i=1;i<=mm;i++)
        if(pre[i]) ans+=gra[pre[i]][i];
    return ans;
}
```

2.2 连通性

2.2.1 强联通

```
int n,m;
vector<int> gra[2005];
bool vis[2005];        //vis 初始化 0
int stack[2005],sk;
int low[2005],scc[2005];  // scc[i]=j 第 i 个点在第 j 个强连通分量里
int timer,cnt;     //每次调用 cnt 初始化 1 timer 初始化 0
void tarjan(int s)
{
    vis[s]=1;
    int to,temp=low[s]=++timer;
    stack[sk++]=s;
    for(int i=0;i<gra[s].size();i++)
    {
        to=gra[s][i];
        if(vis[to]==0)tarjan(to);
        low[s]=min(low[to],low[s]);
    }
    if(temp==low[s])
    {
        do
        {
            to=stack[--sk];
            low[to]=0x3fffffff;
            scc[to]=cnt;
        }while(stack[sk]!=s);
        cnt++;
    }
}
```

2.2.2 双联通分量

```
//边连通分量
struct edge
{
    int y,pdc,next;
    edge(int y_=0,int pdc_=0,int n_=0){y=y_,pdc=pdc_,next=n_;}
}e[2000005];
int head[1005],mm;  //mm=0  head 初始化 -1
vector<int>bri[2];  //桥 初始化
```

```
bool vis[1005];   //初始化全 0
int low[1005],dfn[1005];
int edc[1005];
int stack[1005],sk;
bool cut[1005];   //割点 初始化全 0
int ecnt,timer,root;    //timer 初始化为 0 cnt 初始化为 1 root 为根
void dfs(int s,int fa)
{
  dfn[s]=low[s]=++timer;
  stack[sk++]=s;
  int to,sz=0;
  bool flag=vis[s]=1;
  for(int i=head[s];~i;i=e[i].next)
    if((to=e[i].y)==fa&&flag)flag=0;
    else
        {
            if(vis[to])low[s]=min(dfn[to],low[s]);
            else
            {
                dfs(to,s);sz++;
                low[s]=min(low[to],low[s]);
                if(low[to]>=dfn[s])
                {
                    if(s!=root||sz>1)cut[s]=1;
                }
                if(low[to]>dfn[s])
                {
                    bri[0].push_back(s);
                    bri[1].push_back(to);
                }
            }
        }
    if(low[s]==dfn[s])
    {
        do edc[stack[--sk]]=ecnt;
        while(stack[sk]!=s);
        ecnt++;
    }
}
void add(int x,int y)
{
    e[mm]=edge(y,0,head[x]);head[x]=mm++;
    e[mm]=edge(x,0,head[y]);head[y]=mm++;
}


struct edge
{
    int y,pdc,next;
    edge(int y_=0,int pdc_=0,int n_=0){y=y_,pdc=pdc_,next=n_;}
}e[2000005];
int head[1005],mm;
bool vis[1005];
int low[1005],dfn[1005];
int stack2[2000005],sk2;
bool cut[1005];
int pcnt,timer,root;
void dfs(int s,int fa)
{
  dfn[s]=low[s]=++timer;
  int to,sz=0;
  bool flag=vis[s]=1;
  for(int i=head[s];~i;i=e[i].next)
        if(e[i].pdc==0)
        {
            stack2[sk2++]=i;
            if((to=e[i].y)==fa&&flag)flag=0;
            else
            {
                if(vis[to])low[s]=min(dfn[to],low[s]);
                else
                {
                    dfs(to,s);sz++;
                    low[s]=min(low[to],low[s]);
                    if(low[to]>=dfn[s])
                    {
                        if(s!=root||sz>1)cut[s]=1;
                        do to=stack2[--sk2],e[to].pdc=e[to^1].pdc=pcnt;
                        while(stack2[sk2]!=i);
                        pcnt++;
```

```
                    }
                }
            }
        }
}
void add(int x,int y)
{
    e[mm]=edge(y,0,head[x]);head[x]=mm++;
    e[mm]=edge(x,0,head[y]);head[y]=mm++;
}
```

2.2.3 桥 && 割点

```
//割点 & 桥
vector<int>gra[10005];
bool v[10005],cut[10005];  //初始化全 0
int low[10005],dfn[10005];
int timer,root;   //timer=1  root 初始化
void dfs(int s,int p)
{
    bool f=1;
    int y,cnt=0;
    v[s]=1;
    dfn[s]=low[s]=timer++;
    for(int i=0;i<gra[s].size();i++)
        if((y=gra[s][i])==p&&f)f=0;
        else
        {
            if(v[y])low[s]=min(low[s],dfn[y]);
            else
            {
                dfs(y,s);cnt++;
                low[s]=min(low[s],low[y]);
                if((s!=root&&low[y]>=dfn[s])||(s==root&&cnt>1))cut[s]=1;
                if(low[y]>dfn[s])
                {
                    //bri[0].push_back(s);
                    //bri[1].push_back(y);
                }
            }
        }
    if(low[s]==dfn[s])
    {
        int h;
        do
        {
            h=stack[--k];
            col[h]=cnt;
        }while(stack[k]!=s);
        cnt++;
    }
}
```

2.2.4 欧拉路径

```

void eur(int s,int p)
{
    for(int i=0;i<gra[s].size();i++)
        if(v[gra[s][i].second]==0)
        {
            v[gra[s][i].second]=1;
            eur(gra[s][i].first,gra[s][i].second);
        }
    ans[h++]=p;
}
```

2.2.5 仙人掌图

```
vector<int>gra[20050];
//判定

int vis[20050];
int dfn[20050],low[20050];
int timer;
bool Cactus(int s)
{
    int y,cnt=0;
    vis[s]=1;
```

```
    dfn[s]=low[s]=timer++;
    for(int i=0;i<gra[s].size();i++)
    {
        if(vis[y=gra[s][i]]==2)return 0;
        if(vis[y]==0)
        {
            if(!Cactus(y))return 0;
            if(low[y]>dfn[y])return 0;
            low[s]=min(low[s],low[y]);
        }
        else low[s]=min(low[s],dfn[y]);
        if(low[y]<dfn[s])cnt++;
        if(cnt>1)return 0;
    }
    vis[s]=2;
    return 1;
}
```

2.3 网络流

2.3.1 SAP

```
Ver 3.0 邻接表，注意边开到两倍()
// O(EV^2) 下标从   1 开始
struct edge
{
    int y,c,next;
    edge(){}
    edge(int y_,int c_,int n_){y=y_,c=c_,next=n_;}
}e[900005];
int head[20005];  //初始化 -1;
int nn,mm;     //初始化 mm=0  nn 为点数
int d[20005],cont[20005],q[20005];
int pre[20005],cur[20005];
bool vis[20005];
void bfs(int s)
{
    int x,to,tail=1,front=0;
    for(int i=1;i<=nn;i++) vis[i]=cont[i]=0,cur[i]=head[i],d[i]=0x3fffffff;
    d[s]=0;cont[0]=1;q[0]=s;vis[s]=1;
    while(front<tail)
        for(int i=head[x=q[front++]];i!=-1;i=e[i].next)
            if(!vis[to=e[i].y] && e[i^1].c)
            {
                d[to]=d[x]+1;
                vis[to]=1;
                q[tail++]=to;
                cont[d[to]]++;
            }
}
int SAP(int s,int t)
{
  if(s==t) return 0x7fffffff;
    bfs(t);pre[s]=-1;               //不用 bfs 的话把初始化复制过来  d[i]=0;
    int ans=0,x=s,y,len=0,flow,back;
    while(d[s]<nn)
    {
        y=-1;
        for(int i=cur[x];i!=-1;i=e[i].next)
            if(e[i].c && d[x]==d[e[i].y]+1)
            {
                y=e[i].y;
                cur[x]=i;
                break;
            }
        if(y!=-1)
        {
            pre[y]=x;x=y;
            if(x==t)
            {
                flow=0x3fffffff;
                for(y=pre[y];y!=-1;y=pre[y])
                    if(flow>=e[cur[y]].c) flow=e[cur[y]].c,back=y;
                for(x=pre[x];x!=-1;x=pre[x])
                    e[cur[x]^1].c+=flow,e[cur[x]].c-=flow;
                ans+=flow;x=back;
            }
        }
        else
```

```
        {
            y=nn;
            for(int i=head[x];i!=-1;i=e[i].next)
                if(e[i].c && y>d[e[i].y])
                    y=d[e[i].y],cur[x]=i;
            cont[d[x]]--;
            if(cont[d[x]]==0) break;
            cont[d[x]=y+1]++;
            if(x!=s) x=pre[x];
        }
    }
    return ans;
}
void add(int x,int y,int c)   //无向图 可以优化
{
    e[mm]=edge(y,c,head[x]);head[x]=mm++;
    e[mm]=edge(x,0,head[y]);head[y]=mm++;
}
```

2.3.2 全局最小割

```
//标号 0 开始全局最小割
int cap[505][505];
int node[505],dis[505];
int Stoer_Wagner(int n)
{
    int i,j,now,ans=0x3fffffff;
    for(int i=0;i<n;i++) node[i]=i;
    while(n>1)
    {
        for(now=0,i=1;i<n;i++) dis[node[i]]=0;
        for(i=1;i<n;i++)
        {
            swap(node[now],node[i-1]);
            for(now=j=i;j<n;j++)
            {
                dis[node[j]]+=cap[node[i-1]][node[j]];
                if(dis[node[now]]<dis[node[j]]) now=j;
            }
        }
        ans=min(ans,dis[node[now]]);
        for(j=0;j<n;j++)
            cap[node[j]][node[now-1]]=cap[node[now-1]][node[j]]+=cap[node[j]][node[now]];
        node[now]=node[--n];
    }
    return ans;
}
```

2.3.3 MCMF

```
//适合增广路长或费用范围大的图
Ver 3.0 支持重边
struct edge
{
    int y,c,f,next;
    edge(){}
    edge(int y_,int c_,int f_,int n_){y=y_,c=c_,f=f_,next=n_;}
}e[2007000];
int head[2005];
int nn,mm;
int q[2005],p[2005],dis[2005];
bool vis[2005];
bool spfa(int s,int t)
{
    int to,tail=1,front=0;
    for(int i=1;i<=nn;i++) vis[i]=0,dis[i]=0x3fffffff;
    dis[s]=0;q[0]=s;
    while(tail!=front)
    {
        s=q[front++];vis[s]=0;
        if(front>nn)front=0;
        for(int i=head[s];~i;i=e[i].next)
            if(e[i].c&&dis[to=e[i].y]>dis[s]+e[i].f)
            {
                dis[to]=dis[s]+e[i].f;p[to]=i^1;
                if(vis[to]==0)
                {
                    vis[to]=1;
                    if(dis[to]<dis[q[front]])
```

```
                        front==0?front=nn:front--,q[front]=to;
                    else
                        q[tail++]=to,tail>nn?tail=0:0;
                }
            }
    }
    if(dis[t]<0x3fffffff) return 1;
    else return 0;
}
int cost_flow(int s,int t)
{
  int ans=0,flow,x;
  while(spfa(s,t))
  {
      flow=0x3fffffff;
    for(x=t;x!=s;x=e[p[x]].y) flow=min(flow,e[p[x]^1].c);
    for(x=t;x!=s;x=e[p[x]].y)
      e[p[x]].c+=flow,e[p[x]^1].c-=flow,ans+=flow*e[p[x]^1].f;
  }
  return ans;
}
inline void add(int x,int y,int c,int f)
{
    e[mm]=edge(y,c,f,head[x]);head[x]=mm++;
    e[mm]=edge(x,0,-f,head[y]);head[y]=mm++;
}
```

2.3.4 Primal Dual

```
//zkw 和 spfa 的结合基本一般类型图都比较快

struct edge
{
    int y,c,f,next;
    edge(){}
    edge(int y_,int c_,int f_,int n_){y=y_,c=c_,f=f_,next=n_;}
}e[2008000];
int head[2005];   //初始化 -1
int nn,mm;        //初始化 mm 为0  nn 为点数
int phi[2005],q[2005];
int s_,t_,cost_,tot_;
bool v[2005];
inline bool relable()
{
    int x,to,tail=1,front=0;
    for(int i=1;i<=nn;i++)v[i]=0,phi[i]=0x3fffffff;
    phi[t_]=0;q[0]=t_;
    while(tail!=front)
    {
        x=q[front++];v[x]=0;
        if(front>nn)front=0;
        for(int i=head[x];~i;i=e[i].next)
            if(e[i^1].c&&phi[to=e[i].y]>phi[x]-e[i].f)
            {
                phi[to]=phi[x]-e[i].f;
                if(v[to]==0)
                {
                    v[to]=1;
                    if(phi[to]<phi[q[front]])
                        front==0?front=nn:front--,q[front]=to;
                    else
                        q[tail++]=to,tail>nn?tail=0:0;
                }
            }
    }
    for(int i=1;i<=nn;i++)
        for(int j=head[i];~j;j=e[j].next)
            e[j].f+=phi[e[j].y]-phi[i];
    tot_+=phi[s_];
    return phi[s_]<0x3fffffff;
}
inline int aug(int s,int flow)
{
    if(s==t_)return cost_+=tot_*flow,flow;
    int res=flow,te;v[s]=1;
    for(int i=head[s];~i;i=e[i].next)
        if(!v[e[i].y]&&!e[i].f&&e[i].c)
        {
            res-=(te=aug(e[i].y,min(res,e[i].c)));
            e[i].c-=te;e[i^1].c+=te;
```

```
            if(!res)return flow;
        }
    return flow-res;
}
int cost_flow(int s,int t)
{
    cost_=tot_=0;s_=s;t_=t;
    if(!relable())return 0;
    do memset(v,0,sizeof(v));
    while(aug(s,0x3fffffff)||relable());
    return cost_;
}
inline void add(int x,int y,int c,int f)
{
    e[mm]=edge(y,c,f,head[x]);head[x]=mm++;
    e[mm]=edge(x,0,-f,head[y]);head[y]=mm++;
}
```

2.4 最短路

2.4.1 Dijkstra

```
#include<queue>
vector<pair<int,int> > gra[160005];
int dis[160005];
int nn;
bool vis[160005];
inline int Dijkstra(int s,int t)
{
    int mdis,mx,k=1,to,w;
    priority_queue<pair<int,int>,vector<pair<int,int> >,greater<pair<int,int> > > q;
    for(int i=1;i<=nn;i++) dis[i]=0x3fffffff,vis[i]=0;
    q.push(make_pair(0,s));dis[s]=0;
    while(!q.empty())
    {
        mdis=q.top().first;
        mx=q.top().second;q.pop();
        if(vis[mx]) continue;
        if(mx==t) return mdis;
        vis[mx]=1;
        for(int i=0;i<gra[mx].size();i++)
            if(dis[to=gra[mx][i].first]>(w=gra[mx][i].second)+mdis)
                dis[to]=w+mdis,q.push(make_pair(dis[to],to));
    }
    return -1;
}
```

2.4.2 SPFA

```
//SLF 优化LLL
SPFA 算法有两个优化算法 SLF 和 LLL ：
SLF ： Small Label First 策略，设要加入的节点是 j ，队首元素为
 i ，若 dist(j)<dist(i) ，则将 j 插入队首，否则插入队尾。
LLL ： Large Label Last 策略，设队首元素为 i ，队列中所有 dist 值的平均值为 x ，若
 dist(i)>x 则将 i 插入到队尾，查找下一元素，直到找到某一 i 使得 dist(i)<=，x则将出对进行松弛操作。
i

//Parent Checking检查当前结点的父亲是否在队列中在的话直接抛弃当前节点的更新




//负环
//加 SLF 优化的话只能用  1维护路径长度大于
1. n 有负环入队次数大于
2. n 有负环


// spfa-dfs 专找负环     inq 要事先初始化
vector<pair<int,int> >gra[700];
int now[700];
bool inq[700];
bool spfa_dfs(int s)
{
    int to,w;
    inq[s]=1;
    for(int i=0;i<gra[s].size();i++)
    {
```

```
        to=gra[s][i].first;
        w=gra[s][i].second;
        if(now[to]<now[s]+w)
        {
            now[to]=now[s]+w;
            if(inq[to])return 1;
            else if(spfa_dfs(to))return 1;
        }
    }
    inq[s]=0;
    return 0;
}
```

2.5 树形图论

2.5.1 LCA

```
// tanjar 离线算法
void dfs(int s,int p)
{
    int x;
    for(int i=0;i<gra[s].size();i++)
        if((x=gra[s][i])!=p)dfs(x,s),com[x]=s;
    vis[s]=1;
    for(int i=0;i<q[s].size();i++)
        if(vis[x=q[s][i]]) ans[id[s][i]]=find(x);
}

//倍增 标号 1 开始根节点深度设为    1
// pre 数组初始化!!!
//预处理
int pre[50050][20];
int dep[50050];
void dfs(int s,int p)
{
    int t=0;
    pre[s][0]=p;
    while(pre[s][t]) pre[s][t+1]=pre[pre[s][t]][t],t++;
    for(int i=0;i<gra[s].size();i++)
        if((t=gra[s][i])!=p)
        {
            dep[t]=dep[s]+1;
            dfs(t,s);
        }
}
int lca(int x,int y)
{
    int s=19;
    if(dep[x]>dep[y]) swap(x,y);
    while(dep[x]<dep[y]) y=dep[pre[y][s]]<dep[x]?y:pre[y][s],s--;
    if(x==y) return x;
    else
    {
        s=19;
        while(s!=-1)
        {
            if(pre[x][s]!=pre[y][s])
                x=pre[x][s],y=pre[y][s];
            s--;
        }
        return pre[x][0];
    }
}
```

2.5.2 度限制最小生成树

```
//O(m*logm+k*m) 要求某个点度数小于或等于  k
#define INF 0x3fffffff
struct edge
{
    int x,y,w;
    bool operator<(const edge a)const
    {return w<a.w;}
}e[450];
int n,m;
int dis[25][25],temp[25];
int com[25],con[25];
int find(int s){return s==com[s]?s:com[s]=find(com[s]);}
int dfs(int s,int p,int x,int y)
```

```
{
    if(con[s]<INF)
    {
        if(temp[0]>con[s]-dis[x][y])
            temp[0]=con[s]-dis[x][y],temp[1]=x,temp[2]=y,temp[3]=s;
    }
    for(int i=0;i<n;i++)
        if(dis[i][s]<INF && i!=p)
        {
            if(dis[x][y]<dis[i][s]) dfs(i,s,s,i);
            else dfs(i,s,x,y);
        }
    return 0;
}
int K_MST(int root,int k)
{
    int x,y,z=n-1,s=0,ans=0;
    sort(e,e+m);
    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
            dis[i][j]=INF;
    for(int i=0;i<n;i++)
        com[i]=i,temp[i]=-1,con[i]=INF;
    for(int i=0;i<m;i++)
        if(e[i].x!=root && e[i].y!=root)
        {
            x=find(e[i].x),y=find(e[i].y);
            if(x!=y)
            {
                if(x>y) swap(x,y);
                com[x]=y;ans+=e[i].w;z--;
                dis[e[i].x][e[i].y]=dis[e[i].y][e[i].x]=e[i].w;
            }
        }
        else
        {
            x=e[i].x==root?e[i].y:e[i].x;
            con[x]=min(con[x],e[i].w);
        }
    for(int i=0;i<n;i++)
        if(i!=root)
        {
            x=find(i);
            if(temp[x]==-1) temp[x]=i;
            else temp[x]=con[i]<con[temp[x]]?i:temp[x];
            if(x==i)
            {
                if(con[temp[x]]!=INF)
                    dis[temp[x]][root]=dis[root][temp[x]]=con[temp[x]],ans+=con[temp[x]],s++;
            }
        }
    if(z>s || s>k) return -1;
    for(int i=s;i<k;i++)
    {
        temp[0]=INF;
        for(int j=0;j<n;j++)
            if(dis[root][j]<INF) dfs(j,root,root,j);
        if(temp[0]>=0) break;
        dis[root][temp[3]]=dis[temp[3]][root]=con[temp[3]];
        dis[temp[1]][temp[2]]=dis[temp[2]][temp[1]]=INF;ans+=temp[0];
    }
    return ans;
}
```

2.5.3 曼哈顿最小生成树

```
struct point
{
    int x,y,i;
    bool operator<(point p)const
    {
        return x!=p.x?x<p.x:y<p.y;
    }
}p[10050];
struct edge
{
    int x,y,w;
    bool operator<(edge p)const
    {
        return w<p.w;
```

```
    }
}e[40050];
int n,k,m,ans;
int a[10050],b[10050];
int bit[10050][2];
int query(int x)
{
    int s=inf,f=-1;
    while(x<=m)
    {
        if(s>bit[x][0]) s=bit[x][0],f=bit[x][1];
        x+=x&-x;
    }
    return f;
}
void updat(int x,int val,int i)
{
    while(x)
    {
        if(bit[x][0]>val) bit[x][0]=val,bit[x][1]=i;
        x-=x&-x;
    }
}
int com[10050];
int find(int s)
{
    return s==com[s]?s:com[s]=find(com[s]);
}
void Manhattan()
{
    int pos,f,tot=0;
    for(int j=0;j<4;j++)
    {
        if(j==1||j==3) for(int i=0;i<n;i++) swap(p[i].x,p[i].y);
        else if(j==2) for(int i=0;i<n;i++) p[i].x=-p[i].x;
        sort(p,p+n);
        for(int i=0;i<n;i++)
            a[i]=b[i]=p[i].y-p[i].x;
        sort(b,b+n);
        m=unique(b,b+n)-b;
        for(int i=1;i<=m;i++)
            bit[i][0]=inf,bit[i][1]=-1;
        for(int i=n-1;i>-1;i--)
        {
            pos=lower_bound(b,b+m,a[i])-b+1;
            f=query(pos);
            if(f!=-1)
            {
                e[tot].x=p[i].i;e[tot].y=p[f].i;
                e[tot++].w=p[f].x+p[f].y-p[i].x-p[i].y;
            }
            updat(pos,p[i].y+p[i].x,i);
        }
    }
    sort(e,e+tot);
    int cnt=0,x,y;
    for(int i=0;i<tot;i++)
    {
        x=find(e[i].x);y=find(e[i].y);
        if(x==y) continue;
        else
        {
            com[x]=y;
            cnt++;
            if(cnt==n-k)
            {
                ans=e[i].w;
                return ;
            }
        }
    }
}
int main()
{
    freopen("1.txt","r",stdin);
    scanf("%d%d",&n,&k);
    for(int i=0;i<n;i++)
        scanf("%d%d",&p[i].x,&p[i].y),p[i].i=i,com[i]=i;
    Manhattan();
    printf("%d\n",ans);
```

```
    return 0;
}
```

2.5.4 最小树形图

```
//标号从 1 开始返回  -1 无解
//O(n*m)
struct edge
{
    int x,y;
    double w;
}e[10005];  //注意去除自环
int nn,mm;
double in[105];
int vis[105],p[105],id[105];
double Minimum_Arborescence(int root)
{
    double ans=0;
    int cnt,nod=nn,x,y;
    while(1)
    {
        for(int i=1;i<=nod;i++)in[i]=0x3fffffff;
        for(int i=0;i<mm;i++)
            if(in[e[i].y]>e[i].w&&e[i].x!=e[i].y)
            {
                in[e[i].y]=e[i].w;
                p[e[i].y]=e[i].x;
            }
        in[root]=0;
        for(int i=1;i<=nod;i++)
            if(in[i]==0x3fffffff)return -1;
        cnt=1;
        for(int i=1;i<=nod;i++)
            id[i]=-1,vis[i]=-1;
        for(int i=1;i<=nod;i++)
        {
            ans+=in[i];x=i;
            while(vis[x]!=i && id[x]==-1 && x!=root)
                vis[x]=i,x=p[x];
            if(x!=root && id[x]==-1)
            {
                for(y=p[x];x!=y;y=p[y])
                    id[y]=cnt;
                id[x]=cnt++;
            }
        }
        if(cnt==1) return ans;
        for(int i=1;i<=nod;i++)
            if(id[i]==-1)id[i]=cnt++;
        for(int i=0;i<mm;i++)
        {
            y=e[i].y;
            e[i].x=id[e[i].x];
            e[i].y=id[e[i].y];
            if(e[i].x!=e[i].y) e[i].w-=in[y];
        }
        nod=cnt-1;
        root=id[root];
    }
}
```

2.5.5 一般图最大匹配

```
#include <iostream>
#include <stdio.h>
#include <string.h>
#include <cmath>
#include <algorithm>
#include <queue>
using namespace std;
#define MAXE 250*250*2
#define MAXN 250
#define SET(a,b) memset(a,b,sizeof(a))
deque<int> Q;
//g[i][j存放关系图：]i,是否有边j,match[i存放]所匹配的点i
bool g[MAXN][MAXN],inque[MAXN],inblossom[MAXN];
int match[MAXN],pre[MAXN],base[MAXN];

//找公共祖先
```

```
int findancestor(int u,int v){
    bool inpath[MAXN]={false};
    while(1){
        u=base[u];
        inpath[u]=true;
        if(match[u]==-1)break;
        u=pre[match[u]];
    }
    while(1){
        v=base[v];
        if(inpath[v])return v;
        v=pre[match[v]];
    }
}

//压缩花
void reset(int u,int anc){
    while(u!=anc){
        int v=match[u];
        inblossom[base[u]]=1;
        inblossom[base[v]]=1;
        v=pre[v];
        if(base[v]!=anc)pre[v]=match[u];
        u=v;
    }
}

void contract(int u,int v,int n){
    int anc=findancestor(u,v);
    //SET(inblossom,0);
    memset(inblossom,0,sizeof(inblossom));
    reset(u,anc);reset(v,anc);
    if(base[u]!=anc)pre[u]=v;
    if(base[v]!=anc)pre[v]=u;
    for(int i=1;i<=n;i++)
        if(inblossom[base[i]]){
            base[i]=anc;
            if(!inque[i]){
                Q.push_back(i);
                inque[i]=1;
            }
        }
}

bool dfs(int S,int n){
    for(int i=0;i<=n;i++)pre[i]=-1,inque[i]=0,base[i]=i;
    Q.clear();Q.push_back(S);inque[S]=1;
    while(!Q.empty()){
        int u=Q.front();Q.pop_front();
        for(int v=1;v<=n;v++){
            if(g[u][v]&&base[v]!=base[u]&&match[u]!=v){
                if(v==S||(match[v]!=-1&&pre[match[v]]!=-1))contract(u,v,n);
                else if(pre[v]==-1){
                    pre[v]=u;
                    if(match[v]!=-1)Q.push_back(match[v]),inque[match[v]]=1;
                    else{
                        u=v;
                        while(u!=-1){
                            v=pre[u];
                            int w=match[v];
                            match[u]=v;
                            match[v]=u;
                            u=w;
                        }
                        return true;
                    }
                }
            }
        }
    }
    return false;
}

int main(){
    int n,m,a,b,ans,i;
    while(scanf("%d",&n)!=EOF){
        ans=0;          //最多有几对匹配
        memset(match,-1,sizeof(match));
        memset(g,0,sizeof(g));
        while(scanf("%d%d",&a,&b)!=EOF&&a!=0){
```

```
97              g[a][b]=g[b][a]=1;
98          }
99          for(i=1;i<=n;i++){
100             if(match[i]==-1&&dfs(i,n)){
101                 ans++;
102             }
103         }
104         cout<<ans*2<<endl;
105         for(i=1;i<=n;i++){
106             if(match[i]!=-1){
107                 printf("%d %d\n",i,match[i]);
108                 match[i]=match[match[i]]=-1;
109             }
110         }
111     }
112     return 0;
113 }
```

2.5.6 一般图最大权匹配

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  typedef long long LL;
6  const int N = 100;
7
8
9  int G[N][N];
10 int cnt_node;
11
12 int  dist[N];
13 bool vst[N];
14 int rec[N],cr;
15 int M[N];
16 int P[N];
17
18 const int inf = 0x3f3f3f3f;
19
20 bool spfa(int u)
21 {
22     rec[cr++]=u;
23     if(vst[u]) return true;
24     vst[u]=true;
25     int v;
26     for(v=0;v<cnt_node;v++)
27     {
28         if(v!=u&&M[u]!=v&&!vst[v])
29         {
30             int w=M[v];
31             if(dist[w]>dist[u]+G[u][v]-G[v][w])
32             {
33                 dist[w]=dist[u]+G[u][v]-G[v][w];
34                 if(spfa(w))
35                 {
36                     return true;
37                 }
38             }
39         }
40     }
41     cr--;
42     vst[u]=false;
43     return false;
44 }
45
46 int match()
47 {
48     int i;
49     for(i=0;i<cnt_node;i++) P[i]=i;
50     for(i=0;i<cnt_node;i+=2) M[i]=i+1,M[i+1]=i;
51     int cnt=0;
52     while(1)
53     {
54         //memset(dist,0,sizeof(dist));
55         for (int i = 0; i < cnt_node; i++)
56             dist[i] = inf;
57         cr=0;
58         int i;
59         bool fd=false;
60         memset(vst,0,sizeof(vst));
```

```
        for(i=0;i<cnt_node;i++)
        {
            if(spfa(P[i]))
            {
                fd=true;
                int j;
                int nx=M[rec[cr-1]];
                for(j=cr-2;rec[j]!=rec[cr-1];j--)
                {
                    M[nx]=rec[j];
                    int tmp=nx;
                    nx=M[rec[j]];
                    M[rec[j]]=tmp;
                }
                M[nx]=rec[j];
                M[rec[j]]=nx;
                break;
            }
        }
        if(!fd)
        {
            cnt++;
            if(cnt>=3) break;
        //  random_shuffle(P,P+cnt_node);
            for(i=0;i<cnt_node;i++)
            {
                swap(P[i],P[i+rand()%(cnt_node-i)]);
            }
        }
    }
    int sum=0;
    for(i=0;i<cnt_node;i++)
    {
        int v=M[i];
        if(i<v)
        {
            sum+=G[i][v];
        }
    }
    return sum;
}

int main() {
    int T;
    scanf("%d", &T);
    for (int cas = 1; cas <= T; cas++) {
        int n, m, K;
        scanf("%d%d%d", &n, &m, &K);
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                G[i][j] = inf;
                if (i == j) G[i][j] = 0;
            }
        }
        for (int i = 0; i < m; i++) {
            int u, v, w;
            scanf("%d%d%d", &u, &v, &w);
            u--; v--;
            G[u][v] = G[v][u] = min(G[u][v], w);
        }
        for (int k = 0; k < n; k++) {
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    if (G[i][j] > G[i][k] + G[k][j]) {
                        G[i][j] = G[i][k] + G[k][j];
                    }
                }
            }
        }
        cnt_node = K;
        printf("Case␣%d:␣", cas);
        if (K % 2 == 1) {
            printf("Impossible\n");
        } else {
            printf("%d\n", match());
        }
    }
    return 0;
}
```

2.5.7 次小生成树

```
// O(n^2)
int e[105][105],cir[105][105]; //初始化无穷e 初始化cir0
int dis[105],ind[105],pre[105];
bool use[105][105];
void SEC_MST(int n)
{
    int temp,f,ans=0,ans2=0x3fffffff;
    for(int i=0;i<n;i++)
        dis[i]=e[0][i],ind[i]=i,pre[i]=0;
    for(int i=1;i<n;i++)
    {
        f=i;temp=dis[ind[i]];
        for(int j=i+1;j<n;j++)
            if(dis[ind[j]]<temp)
                f=j,temp=dis[ind[j]];
        swap(ind[f],ind[i]);f=ind[i];
        use[f][pre[f]]=use[pre[f]][f]=1;
        for(int j=0;j<i;j++)
            cir[f][ind[j]]=cir[ind[j]][f]=max(e[f][pre[f]],cir[ind[j]][pre[f]]);
        ans+=dis[f];
        for(int j=i+1;j<n;j++)
            if(dis[ind[j]]>e[f][ind[j]]) dis[ind[j]]=e[f][ind[j]],pre[ind[j]]=f;
    }
    for(int i=0;i<n;i++)
        for(int j=i+1;j<n;j++)
            if(use[i][j]==0)
                ans2=min(ans2,e[i][j]-cir[i][j]);
}
```

3 Datastruct

3.1 数据结构

3.1.1 BIT

```
//更新区间求点 或更新点求区间不能有       0 !!!!
//求出 [ 1 , x ] 所有数的和在    x 处加入 v
struct bit_tree
{
    int bit[100050],s;
    inline void updat(int x,int v)
    {
        while(x<100005) bit[x]+=v,x+=x&-x;
    }
    inline int query(int x)
    {
        s=0;
        while(x) s+=bit[x],x-=x&-x;
        return s;
    }
    inline int find(int x)
    {
        int cnt=0,ans=0,p=1<<18;
        while(p)
        {
            ans|=p;
            if(ans>100001||cnt+bit[ans]>=x)ans^=p;
            else cnt+=bit[ans];p>>=1;
        }
        return ans+1;
    }
};
```

3.1.2 Partition Tree

```

int tree[20][MAXN];//表示每层上每个位置的值
int sorted[MAXN];//排好序的数
int toleft[20][MAXN];//每层 1 — i 有多少个数到左边

void build(int l,int r,int dep)
{
    if(l==r)return;
    int mid=(l+r)>>1;
    int same=mid-l+1;// same 表示等于中间值且到左边的数的个数
    for(int i=l;i<=r;i++)
      if(tree[dep][i]<sorted[mid])
        same--;
    int lpos=l;
    int rpos=mid+1;
    for(int i=l;i<=r;i++)
    {
        if(tree[dep][i]<sorted[mid])//去左边
        {
            tree[dep+1][lpos++]=tree[dep][i];

        }
        else if(tree[dep][i]==sorted[mid]&&same>0)//去左边
        {
            tree[dep+1][lpos++]=tree[dep][i];
            same--;
        }
        else//去右边
           tree[dep+1][rpos++]=tree[dep][i];
        toleft[dep][i]=toleft[dep][l-1]+lpos-l;//从 1 到 i 放左边的个数
    }
    build(l,mid,dep+1);//递归建树
    build(mid+1,r,dep+1);
}
int query(int L,int R,int l,int r,int dep,int k)
{
    if(l==r)return tree[dep][l];
    int mid=(L+R)>>1;
    int cnt=toleft[dep][r]-toleft[dep][l-1];
    if(cnt>=k)
    {
```

```
42          // L 查询区间前去左边的数的个数+
43          int newl=L+toleft[dep][l-1]-toleft[dep][L-1];
44          //左端点查询区间会分入左边的数的个数+
45          int newr=newl+cnt-1;
46          return query(L,mid,newl,newr,dep+1,k);//注意
47      }
48      else
49      {
50          // r 区间后分入左边的数的个数+
51          int newr=r+toleft[dep][R]-toleft[dep][r];
52          //右端点减去区间分入右边的数的个数
53          int newl=newr-(r-l-cnt);
54          return query(mid+1,R,newl,newr,dep+1,k-cnt);//注意
55      }
56  }
57  int main()
58  {
59     // freopen("in.txt","r",stdin);
60      //freopen("out.txt","w",stdout);
61      int iCase=0;
62      int n,m;
63      int A,B;
64      while(scanf("%d",&n)!=EOF)
65      {
66          iCase++;
67          for(int i=1;i<=n;i++)
68          {
69              scanf("%d",&tree[0][i]);
70              sorted[i]=tree[0][i];
71          }
72          sort(sorted+1,sorted+n+1);//建树
73          build(1,n,0);
74          scanf("%d",&m);
75          printf("Case␣%d:\n",iCase);
76          while(m--)
77          {
78              scanf("%d%d",&A,&B);
79              int k=(B-A)/2+1;
80              printf("%d\n",query(1,n,A,B,0,k));
81          }
82      }
83      return 0;
84  }
```

3.1.3 Seg Tree

```
1  一维
2  :
3  //求区间和的线段树 支持区间更新
4  struct seg_tree    // i 从 1 开始标
5  {
6      int num[100005];
7      long long laz[400050],dat[400050];
8      long long build(int i,int l,int r)
9      {
10         if(l<r)dat[i]=build(i<<1,l,l+r>>1)+build(i<<1|1,(l+r>>1)+1,r);
11         else dat[i]=num[i];
12         return dat[i];
13     }
14     inline void push_up(int i)
15     {
16         dat[i]=dat[i<<1]+dat[i<<1|1];
17     }
18     inline void push_down(int i,int l,int r)
19     {
20         laz[i]+=laz[i>>1];
21         dat[i]+=laz[i>>1]*(r-l+1);
22     }
23     inline long long query(int i,int l,int r,int l1,int r1)
24     {
25         if(l==l1 && r==r1) return dat[i];
26         else
27         {
28             int mid=l+r>>1;
29             if(laz[i])push_down(i<<1,l,mid),push_down(i<<1|1,mid+1,r),laz[i]=0;
30             if(l1>mid)return query(i<<1|1,mid+1,r,l1,r1);
31             else if(r1<=mid)return query(i<<1,l,mid,l1,r1);
32             else return query(i<<1,l,mid,l1,mid)+query(i<<1|1,mid+1,r,mid+1,r1);
33         }
```

```
    }
    inline void updat(int i,int l,int r,int l1,int r1,long long x)
    {
        if(l==l1 && r==r1) dat[i]+=(r1-l1+1)*x,laz[i]+=x;
        else
        {
            int mid=l+r>>1;
            if(laz[i])push_down(i<<1,l,mid),push_down(i<<1|1,mid+1,r),laz[i]=0;
            if(l1>mid)updat(i<<1|1,mid+1,r,l1,r1,x);
            else if(r1<=mid)updat(i<<1,l,mid,l1,r1,x);
            else updat(i<<1,l,mid,l1,mid,x),updat(i<<1|1,mid+1,r,mid+1,r1,x);
            push_up(i);
        }
    }
};
```

3.1.4 RMQ

```
int num[100050];
int dp[100050][30];
int to[100050];
void rmq()
{
    for(int i=0;i<n;i++)dp[i][0]=num[i];
    for(int l=2,s=0;l<=n;l<<=1,s++)
        for(int i=0;i+l<=n;i++)
      dp[i][s+1]=max(dp[i][s],dp[i+l/2][s]);
}
void pre_log()
{
    to[1]=0;
    for(int i=2;i<n;i+=2)
        to[i]=to[i+1]=to[i>>1]+1;
}
const double C=log(2.0);
int query(int l,int r)  //求 l 到 r 最值从( 0 标号)
{
    if(l>r) swap(l,r);
    int mid=log(r-l+1.0)/C;
  //mid=to[r-l+1];
    int len=1<<mid;
    return min(dp[l][mid],dp[r-len+1][mid]);
}二维

rmq

int num[305][305];
int dp[305][305][9][9];
void rmq_2d(int n,int m)
{
    for(int i=0;i<n;i++)
        for(int j=0;j<m;j++)
            dp[i][j][0][0]=num[i][j];
    for(int l1=1,x=0;l1<=n;l1<<=1,x++)
        for(int l2=1,y=0;l2<=m;l2<<=1,y++)
            for(int i=0;(x||y)&&i+l1<=n;i++)
                for(int j=0;j+l2<=m;j++)
                {
                    if(y==0)dp[i][j][x][y]=min(dp[i][j][x-1][y],dp[i+l1/2][j][x-1][y]);
                    else dp[i][j][x][y]=min(dp[i][j][x][y-1],dp[i][j+l2/2][x][y-1]);
                }
}
const double C=log(2.0);
int query_2d(int x,int y,int x1,int y1)
{
    if(x>x1)swap(x,x1);
    if(y>y1)swap(y,y1);
    int m1=log(x1-x+1.0)/C,m2=log(y1-y+1.0)/C;
  //m=to[x1-x+1];
    int l1=1<<m1,l2=1<<m2;
    return min(min(dp[x][y][m1][m2],dp[x1-l1+1][y1-l2+1][m1][m2]),
               min(dp[x1-l1+1][y][m1][m2],dp[x][y1-l2+1][m1][m2]));
}
```

3.1.5 Splay

```
const int maxn = 100010;
struct Splaytree
{
```

```
4     int sz[maxn];
5     int ch[maxn][2];
6     int pre[maxn];
7     int rt,top;
8
9     int cnt[maxn];
10    int val[maxn];
11    bool laz[maxn];
12    inline void push_up(int x)
13    {
14        sz[x]=cnt[x]+sz[ch[x][0]]+sz[ch[x][1]];
15    }
16    inline void push_down(int x)
17    {
18        if(laz[x])
19        {
20            swap(ch[x][0],ch[x][1]);
21            if(ch[x][0])laz[ch[x][0]]^=1;
22            if(ch[x][1])laz[ch[x][1]]^=1;
23            laz[x]=0;
24        }
25    }
26    inline void Rotate(int x,int f)//f=1 right
27    {
28        int y=pre[x];
29        push_down(y);
30        push_down(x);
31        ch[y][!f]=ch[x][f];
32        pre[ch[x][f]]=y;
33        pre[x]=pre[y];
34        if(pre[x])ch[pre[x]][ch[pre[y]][1]==y]=x;
35        ch[x][f]=y;
36        pre[y]=x;
37        push_up(y);
38    }
39    inline void Splay(int x,int to)  //把编号为 x 的节点旋到编号为 to 的节点的下面
40    {
41        while(pre[x]!=to)            //可以改单旋试试
42            if(pre[pre[x]]==to)Rotate(x,ch[pre[x]][0]==x);
43            else
44            {
45                bool f=(ch[pre[pre[x]]][0]==pre[x]);
46                if(ch[pre[x]][f]==x)Rotate(x,!f),Rotate(x,f);
47                else Rotate(pre[x],f),Rotate(x,f);
48            }
49        push_up(x);
50        if(to==0)rt=x;
51    }
52    inline int get_kth(int k)  //得到第 k 大的节点编号包含 -INF , INF 两个点
53    {
54        int x=rt;
55        push_down(x);
56        while(x)
57        {
58            if(k<=sz[ch[x][0]])x=ch[x][0];
59            else if(k>sz[ch[x][0]]+cnt[x])k-=sz[ch[x][0]]+cnt[x],x=ch[x][1];
60            else return x;
61            push_down(x);
62        }
63        return 0;
64    }
65    inline int get_val(int v)  //得到值为 v 的节点编号如果没有返回值小于 v 的第一个节点编号
66    {
67        int x=rt,y;
68        push_down(x);
69        while(x)
70        {
71            if(v<val[x])x=ch[x][0];
72            else if(v>val[x])x=ch[y=x][1];
73            else return x;
74            push_down(x);
75        }
76        return y;
77    }
78    inline int get_nxt()   //找到大于根的第一个节点编号
79    {
80        int x=ch[rt][1];
81        push_down(x);
82        while(ch[x][0])x=ch[x][0],push_down(x);
83        return x;
```

```
    }
    inline int get_pre()   //找到小于根的第一个节点编号
    {
        int x=ch[rt][0];
        push_down(x);
        while(ch[x][1])x=ch[x][1],push_down(x);
        return x;
    }
    inline void Del_rt()    //删除根节点
    {
        int t=rt;
        if(ch[rt][1])
        {
            rt=ch[rt][1];
            Splay(get_kth(1),0);
            ch[rt][0]=ch[t][0];
            if(ch[rt][0])pre[ch[rt][0]]=rt;
        }
        else rt=ch[rt][0];
        pre[rt]=0;
        push_up(rt);
    }
    inline void Newnode(int &x,int c,int f,int w)   //新建节点 &x=ch[f][?] c 为权值 f 为父亲 w 为大小一般是( 1 )
    {
        pre[x=++top]=f;
        ch[x][0]=ch[x][1]=0;
        sz[x]=cnt[x]=w;
        val[x]=c;
    }
    ////////////////////////////////////

    inline void init()
    {
        ch[0][0]=ch[0][1]=pre[0]=0;
        sz[0]=cnt[0]=0;
        rt=top=0;
        Newnode(rt,-0x3fffffff,0,1);
        Newnode(ch[rt][1],0x3fffffff,rt,1);

        //build_tree(ch[ch[rt][1]][0],0,n-1,ch[rt][1]); 初始化// num

        push_up(ch[rt][1]);
        push_up(rt);
    }
    void build_tree(int &x,int l,int r,int f)
    {
        if(l>r) return ;
        int mid=(l+r)>>1;
        Newnode(x,num[mid],f,1);
        build_tree(ch[x][0],l,mid-1,x);
        build_tree(ch[x][1],mid+1,r,x);
        push_up(x);
    }

}spt;
```

3.1.6 Heavy Light Decomposition

```
// 调用
//
//dep[1]=1;totw=0;
//dfs(1);
//top[1]=1;
//build(1);



int hs[50050],fa[50050],top[50050];
int dep[50050],cw[50050],totw;
int dfs(int s)
{
    int y,z=0,f=0,t,sum=1;
    for(int i=0;i<gra[s].size();i++)
        if((y=gra[s][i])!=fa[s])
        {
            dep[y]=dep[fa[y]=s]+1;
            t=dfs(y);sum+=t;
            if(t>z)z=t,f=y;
        }
    return hs[s]=f,sum;
```

```
}
void build(int s)
{
    int y;
    cw[s]=++totw;
    if(y=hs[s])top[y]=top[s],build(y);
    for(int i=0;i<gra[s].size();i++)
        if((y=gra[s][i])!=fa[s]&&y!=hs[s])
            build(top[y]=y);
}
void updat(int x,int y,int z)
{
    int f1=top[x],f2=top[y];
    while(f1!=f2)
    {
        if(dep[f1]<dep[f2])swap(f1,f2),swap(x,y);
        g.updat(1,1,n,cw[f1],cw[x],z);
        x=fa[f1];f1=top[x];
    }
    if(dep[x]<dep[y])swap(x,y);
    g.updat(1,1,n,cw[y],cw[x],z);  //边的话记得 cw[hs[y]]
}
int query(int s)
{
    return g.query(1,1,n,s);
}
```

3.1.7 Link Cut Tree

```


const int maxn = 10050;
namespace LCT      //深度大右儿子
{
    int ch[maxn][2];
    int pre[maxn];
    bool rt[maxn];

    bool rev[maxn];
    inline void push_up(int x)
    {

    }
    inline void push_down(int x)
    {
        if(rev[x])
        {
            int l=ch[x][0],r=ch[x][1];
            swap(ch[l][0],ch[l][1]);
            swap(ch[r][0],ch[r][1]);
            rev[l]^=1;rev[r]^=1;
            rev[x]=0;
        }
    }
    inline void Rotate(int x,int f)//f=1 right
    {
        int y=pre[x];
        push_down(y);
        push_down(x);
        ch[y][!f]=ch[x][f];
        pre[ch[x][f]]=y;
        pre[x]=pre[y];
        if(rt[y])rt[x]=1,rt[y]=0;
        else ch[pre[x]][ch[pre[y]][1]==y]=x;
        pre[y]=x;
        ch[x][f]=y;
        push_up(y);
    }
    inline void P(int x)
    {
        if(!rt[x])P(pre[x]);
        push_down(x);
    }
    inline void Splay(int x)
    {
        P(x);
        while(!rt[x])
            if(rt[pre[x]])Rotate(x,ch[pre[x]][0]==x);
            else
            {
```

```
52                  bool f=(ch[pre[pre[x]]][0]==pre[x]);
53                  if(ch[pre[x]][f]==x)Rotate(x,!f),Rotate(x,f);
54                  else Rotate(pre[x],f),Rotate(x,f);
55              }
56          push_up(x);
57      }
58      inline int Access(int x)
59      {
60          int y=0;
61          for(;x;x=pre[y=x])
62          {
63              Splay(x);
64              rt[ch[x][1]]=1;
65              rt[ch[x][1]=y]=0;
66              push_up(x);
67          }
68          return y;
69      }
70      inline void Make_root(int x)
71      {
72          Access(x);Splay(x);rev[x]^=1;
73          swap(ch[x][0],ch[x][1]);
74      }
75      inline int Find_root(int x)
76      {
77          Access(x);Splay(x);
78          while(ch[x][0])x=ch[x][0];
79          return x;
80      }
81      inline void Link(int x,int y)
82      {
83          Make_root(x);
84          pre[x]=y;
85      }
86      inline void Cut(int x,int y)
87      {
88          Access(x);
89          int z=Access(y);
90          Access(z);
91          Splay(z^=x^y);
92          pre[z]=0;
93      }
94      inline void Dest(int x,int y)
95      {
96          Make_root(x);
97          Access(y);
98          Splay(y);
99          pre[ch[y][0]]=pre[y];
100         rt[ch[y][0]]=1;
101         pre[y]=ch[y][0]=0;
102         push_up(y);
103     }
104     inline void init(int n)
105     {
106         for(int i=1;i<=n;i++)
107             ch[i][0]=ch[i][1]=pre[i]=rev[i]=0,rt[i]=1;
108     }
109   ***sample***
110     inline bool query(int x,int y)
111     {
112         while(pre[x])x=pre[x];
113         while(pre[y])y=pre[y];
114         return x==y;
115     }
116     inline int query1(int x,int y)
117     {
118         Access(x);
119         int z=Access(y);
120         Splay(x);Splay(z);
121         if(x==z)return max(val[x],dat1[ch[z][1]]);
122         else return max(val[z],max(dat1[x],dat1[ch[z][1]]));
123     }
124   ***sample***
125 }
126
127 vector<int>gra[10050],id[10050];
128 int how[10050],pos[10050];
129 void dfs(int s,int p)
130 {
131     int y;
```

```
    LCT::pre[s]=p;
    for(int i=0;i<gra[s].size();i++)
        if((y=gra[s][i])!=p)
        {
            pos[id[s][i]]=y;
            LCT::val[y]=LCT::dat[y]=how[id[s][i]];
            dfs(y,s);
        }
}
```

3.1.8 Crope

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <ext/rope>

using namespace std;
using namespace __gnu_cxx;

const int maxn=50500;

crope tmp,var[maxn],t;
int n,op,p,c,v,d,curv;
char str[maxn];

int main()
{
    while(scanf("%d",&n)!=EOF)
    {
        curv=0;d=0;
        while(n--)
        {
            scanf("%d",&op);
            if(op==1)
            {
                scanf("%d%s",&p,str);
                p-=d;
                t.insert(p,str);
                var[++curv]=t;
            }
            else if(op==2)
            {
                scanf("%d%d",&p,&c);
                p-=d;c-=d;
                t.erase(p-1,c);
                var[++curv]=t;
            }
            else if(op==3)
            {
                scanf("%d%d%d",&v,&p,&c);
                v-=d;p-=d;c-=d;
                tmp=var[v].substr(p-1,c);
                d+=count(tmp.begin(),tmp.end(),'c');
                cout<<tmp<<endl;
            }
        }
    }
    return 0;
}
```

3.1.9 K-D Tree

```
//找最近的 M 个点..
#include<queue>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int N=55555,K=5;
const int inf=0x3f3f3f3f;

#define sqr(x) (x)*(x)
int k,n,idx;   // k 为维数, n 为点数
struct point
{
    int x[K];
    bool operator < (const point &u) const
```

```
16      {
17          return x[idx]<u.x[idx];
18      }
19  }po[N];
20
21  typedef pair<double,point>tp;
22  priority_queue<tp>nq;
23
24  struct kdTree
25  {
26      point pt[N<<2];
27      int son[N<<2];
28
29      void build(int l,int r,int rt=1,int dep=0)
30      {
31          if(l>r) return;
32          son[rt]=r-l;
33          son[rt*2]=son[rt*2+1]=-1;
34          idx=dep%k;
35          int mid=(l+r)/2;
36          nth_element(po+l,po+mid,po+r+1);
37          pt[rt]=po[mid];
38          build(l,mid-1,rt*2,dep+1);
39          build(mid+1,r,rt*2+1,dep+1);
40      }
41      void query(point p,int m,int rt=1,int dep=0)
42      {
43          if(son[rt]==-1) return;
44          tp nd(0,pt[rt]);
45          for(int i=0;i<k;i++) nd.first+=sqr(nd.second.x[i]-p.x[i]);
46          int dim=dep%k,x=rt*2,y=rt*2+1,fg=0;
47          if(p.x[dim]>=pt[rt].x[dim]) swap(x,y);
48          if(~son[x]) query(p,m,x,dep+1);
49          if(nq.size()<m) nq.push(nd),fg=1;
50          else
51          {
52              if(nd.first<nq.top().first) nq.pop(),nq.push(nd);
53              if(sqr(p.x[dim]-pt[rt].x[dim])<nq.top().first) fg=1;
54          }
55          if(~son[y]&&fg) query(p,m,y,dep+1);
56      }
57  }kd;
58  int main()
59  {
60      while(scanf("%d%d",&n,&k)!=EOF)
61      {
62          for(int i=0;i<n;i++)
63              for(int j=0;j<k;j++)
64                  scanf("%d",&po[i].x[j]);
65          kd.build(0,n-1);
66          int t,m;
67          scanf("%d",&t);
68          while(t--)
69          {
70              point ask;
71              for(int j=0;j<k;j++) scanf("%d",&ask.x[j]);
72              scanf("%d",&m); kd.query(ask,m);
73              printf("the␣closest␣%d␣points␣are:\n", m);
74              point pt[20];
75              for(int j=0;!nq.empty();j++)
76                  pt[j]=nq.top().second,nq.pop();
77              for(int j=m-1;j>=0;j--,putchar(10))
78                  for(int z=0;z<k;z++)
79                      if(z) printf("␣%d",pt[j].x[z]);
80                      else printf("%d",pt[j].x[z]);
81          }
82      }
83      return 0;
84  }
```

3.2 字符串

3.2.1 AC automation

```
1  int to[260005][26],fail[500005],cont[500005];
2  int que[500005],k;
3  char str[1000005],lib[55];
4  void add_trie(char a[])
5  {
```

```
6       int index,p=0;
7       for(int i=0;a[i];i++)
8       {
9           index=a[i]-'a';
10          if(to[p][index]==0)
11          {
12              memset(to[k],0,sizeof(to[k]));
13              fail[k]=cont[k]=0;
14              to[p][index]=k++;
15          }
16          p=to[p][index];
17      }
18      cont[p]++;
19  }
20  void build_ac()
21  {
22      int tail=0,front=0,p;
23      for(int i=0;i<26;i++)
24          if(to[0][i])
25          {
26              fail[to[0][i]]=0;
27              que[front++]=to[0][i];
28          }
29      while(tail<front)
30      {
31          p=que[tail++];
32          for(int i=0;i<26;i++)
33              if(to[p][i])
34              {
35                  fail[to[p][i]]=to[fail[p]][i];
36                  que[front++]=to[p][i];
37              }
38              else to[p][i]=to[fail[p]][i];
39      }
40  }
41  int query(char a[])
42  {
43      int p=0,ans=0,temp;
44      for(int i=0;a[i];i++)
45      {
46          p=to[p][a[i]-'a'];
47          temp=p;
48          while(temp && cont[temp]!=-1)
49          {
50              ans+=cont[temp];
51              cont[temp]=-1;
52              temp=fail[temp];
53          }
54      }
55      return ans;
56  }
57  int main()
58  {
59      freopen("1.txt","r",stdin);
60      int t;
61      scanf("%d",&t);
62      while(t--)
63      {
64          int n;
65          scanf("%d",&n);
66          memset(to[0],0,sizeof(to[0]));
67          cont[0]=0;k=1;
68          for(int i=0;i<n;i++)
69          {
70              scanf("%s",lib);
71              add_trie(lib);
72          }
73          build_ac();
74          scanf("%s",str);
75          printf("%d\n",query(str));
76      }
77      return 0;
78  }
```

3.2.2 KMP

```
1  //get_next
2
3  //优化最好不用()
4  int i=0,j=-1;
```

```
fail[0]=-1;
while(ori[i])
{
    while(j!=-1 && ori[i]!=ori[j]) j=fail[j];
    i++;j++;
    if(ori[j]==ori[i]) fail[i]=fail[j];
    else fail[i]=j;
}

//未优化
int i=0,j=-1;
fail[0]=-1;
while(str[i])
{
    while(j!=-1 && str[i]!=str[j]) j=fail[j];
    i++;j++;
    fail[i]=j;
}


//kmp

i=0;j=0;
while(str[i])
{
    while(j!=-1 && str[i]!=ori[j] ) j=fail[j];
    i++;j++;
    if(ori[j]==0) ans++;
}

//EX-kmp  QAQ

char str[200005],ori[200005];
int fail[200005],extend[200005];
void extend_kmp()
{
    int i,j=-1,l,r;
    for(i=1;ori[i];i++,j--)
    {
        if(j<0 || i+fail[i-l]>=r)
        {
            if(j<0) j=0,r=i;
            while(ori[r] && ori[j]==ori[r]) r++,j++;
            fail[i]=j;l=i;
        }
        else fail[i]=fail[i-l];
    }
    nex[0]=i;j=-1;
    for(i=0;str[i];i++,j--)
    {
        if(j<0 || i+fail[i-l]>=r)
        {
            if(j<0) j=0,r=i;
            while(str[r] && ori[j] && ori[j]==str[r]) r++,j++;
            extend[i]=j;l=i;
        }
        else extend[i]=fail[i-l];
    }
}
```

3.2.3 Suffix Array

```
Ver 2.1
//height[i] 为 sa[i] 与 sa[i-1] 比较从  1 开始
//sa[i] 为字典序排名为的下标是多少i 从 0 开始
//rank[i] 为下标为 i 的排名为多少从  0 开始
int sa[200050],tsa[400050],height[200050];
int rank[200050],tsb[400050],bas[200050];
char str[200050];
int n;
void get_sa()
{
  int i,j,m=128,p,*x=tsa,*y=tsb;
  memset(tsa,-1,sizeof(tsa));
  memset(tsb,-1,sizeof(tsb));
  for(i=0;i<m;i++) bas[i]=0;
  for(i=0;i<n;i++) bas[x[i]=str[i]]++;
  for(i=1;i<m;i++) bas[i]+=bas[i-1];
  for(i=n-1;i>-1;i--) sa[--bas[x[i]]]=i;
  for(j=1,p=1;p<n;j<<=1,m=p)
```

```
  {
    for(p=0,i=n-j;i<n;i++) y[p++]=i;
    for(i=0;i<n;i++) if(sa[i]>=j) y[p++]=sa[i]-j;
    for(i=0;i<m;i++) bas[i]=0;
    for(i=0;i<n;i++) bas[x[y[i]]]++;
    for(i=1;i<m;i++) bas[i]+=bas[i-1];
    for(i=n-1;i>-1;i--) sa[--bas[x[y[i]]]]=y[i];
    for(swap(x,y),p=1,x[sa[0]]=0,i=1;i<n;i++)
            x[sa[i]]=(y[sa[i-1]]==y[sa[i]]&&y[sa[i-1]+j]==y[sa[i]+j])?p-1:p++;
  }

  for(i=0;i<n;i++) rank[sa[i]]=i;
  for(i=0,j=0;i<n;height[rank[i++]]=j)
    if(rank[i]) for(j?j--:0,p=sa[rank[i]-1];str[i+j]==str[p+j];j++);
}
```

3.2.4 Palindrome

```
//回文串
    while(scanf("%s",temp)==1)
    {
        int k=0;
        for(int i=0;temp[i];k++)
            if(k%2) str[k]=temp[i++];
            else str[k]='$';
        if(str[k-1]!='$') str[k++]='$';
        str[k]='#';
        int mx=0;
        for(int i=0;i<k;i++)
        {
            if(pal[mx]+mx>i) pal[i]=min(pal[mx]+mx-i,pal[2*mx-i]);
            else pal[i]=1;
            while(str[i+pal[i]]==str[i-pal[i]]) pal[i]++;
            if(pal[i]+i>mx) mx=i;
        }
    }

//答案 pal[i]-1;
```

3.2.5 Minimal representation

```
//字符串循环 字典序最小返回开始下标
char str[1000005];
int get_min()
{
   int len=strlen(str),i=0,j=1,k=0,t;
   while(i<len && j<len && k<len)
   {
       t=str[(i+k)%len]-str[(j+k)%len];
       if(t)
       {
           if(t>0) i=i+k+1;
           else j=j+k+1;
           if(i==j) j++;
           k=0;
       }
       else k++;
   }
   return min(i,j);
}
```

4 Math

4.1 博弈

4.1.1 Nim 积

```
//快速 sg[a][b]=mex{sg[x][y]^sg[a][y]^sg[x][b]|x<a,y<b}


int m[4][4]={0,0,0,0,0,1,2,3,0,2,3,1,0,3,1,2};
int nim_multy_pow(int x,int y)
{
    if(x<4) return m[x][y];
    int m=1;
    while((1<<m)<=x) m<<=1;
    m=1<<(m>>1);
    int d=nim_multy_pow(x/m,y/m);
    return (m*(d^nim_multy_pow(x/m,y%m)))^nim_multy_pow(m>>1,d);
}
int nim_multy(int x,int y)
{
    if(x<y) swap(x,y);
    if(x<4) return m[x][y];
    int m=1;
    while((1<<m)<=x) m<<=1;
    m=1<<(m>>1);
    int c=nim_multy(x/m,y/m);
    return m*(c^nim_multy(x/m,y%m)^nim_multy(x%m,y/m))^nim_multy(x%m,y%m)^nim_multy_pow(m>>1,c);
}
```

4.1.2 总结

```
//EX 结论
//
//N 阶 Nim 游戏
//结论 当且仅当在每一个不同的二进制位上 x1,...x2xk 中在该位上 1 的个数
//是 N+1 的倍数时手方有必胜策略否则先手必胜
//
//
//不平等博弈
//结论 : 第一段同色权为 1 之后权按位置递减 1/2
//
//
//Anti-sg
//结论 :
//先手必胜当且仅当 :
//(1) 所有堆的石子数都为 1 且游戏的 SG 值为 0 ;
//(2) 有些堆的石子数大于 1 且游戏的 SG 值不为 0 .
//对于任意一个 Anti-SG 游戏，如果我们规定当局面中所有的单一游
//戏的 SG 值为 0 时，游戏结束，则先手必胜当且仅当：
//(1) 游戏的 SG 函数不为 0 且游戏中某个单一游戏的 SG 函数大于 1 ；
//(2) 游戏的 SG 函数为 0 且游戏中没有单一游戏的 SG 函数大于 1 。
//
//
//Every-sg
//形象的说就是红队和蓝队每个队 n 个人然后进行  n 个博弈最后结束的一场博弈的胜者胜利 .
//显然 每个博弈的胜者都想让时间坚持得更久每个败者都想让这场博弈早点结束，不难列出每个点到终止的步数的 DP 方程.
//如果 v 是先手必胜则  f[v]=max(f[u])+1 其中 u 为 v 的后继且 u 为先手必败
//否则 f[v]=min(f[u])+1,u 为 v 后继
//然后可以求出每一个博弈的步数
//求出这个最大值 如果最大值是奇数那么先手必胜
//
//
//
//树删边
//结论
//叶子节点的 SG 值为 0，中间节点的 SG 值为它的所有子节点的
//SG 值加 1 后的异或和
//有环
//对于长度为奇数的环1. 去掉其中任意一个边之后剩下的
//两个链长度同奇偶 抑或之后的 SG 值不可能为奇数所
//以它的 SG 值为 1;
//对于长度为偶数的环2. 去掉其中任意一个边之后剩下的
//两个链长度异奇偶 抑或之后的值不可能为 SG 0，所以它
//的 SG 值为 0;
```

```
//将图中的任意一个偶环缩成一个新点3. 任意一个奇
//环缩成一个新点加一个新边 所有连到原先环上的边全
//部改为与新点相连 这样的改动不会影响图的 SG 值.
//
//
//
//两堆 每次取一堆的任意个或两堆得任意个 ( 相同数量 )
//威佐夫博弈 ak =[k(1+sqrt(5))/2],bk= ak + k
//
//
//三堆 每次取一堆的任意个或三堆得任意个相同数量()
//sg=a^b^c
//
//
//取倍数博弈 (1—n 取 x, 那么 2x,3x...kx 都会被取)
//int sg[]={0,1,2,1,4,3,2,1,5,6,2,1,8,7,
//5,9,8,7,3,4,7,4,2,1,10,9,3,6,11,12};
//
//
//
//博弈Ferguson
//进行游戏需要用到两个盒子在游戏的开始第一个盒子中有.,枚石子n,
//第二个盒子中有个石子m (n, m > 0).
//参与游戏的两名玩家轮流执行这样的操作 : 清空一个盒子中的石子,
//然后从另一个盒子中拿若干石子到被清空的盒子中 使得最后两个盒子都不空
//当两个盒子中都只有一枚石子时 游戏结束最后成功执行操作的玩家获胜
//
//结论对于一个位置: (x, y) 来说如果, x, y 中有一个偶数,
//那么 (x, y) 是N ( 必胜 ) 位置如果.和都是奇数xy那么, (x, y) 是位置P ( 必败 ) .
//
//
//硬币博弈
//
//每次可以翻动一个二个或三个硬币,. ( Mock 游戏Turtles )
//初始编号从开始0.
//位置 x : 0  1  2  3  4   5    6   7    8     9  10  11  12  13  14...
// sg[x] : 1  2  4  7  8  11   13   14   16   19  21  22  25  26  28...
//看上去 sg 值为 2x 或者 2x+1 我们称一个非负整数为. odious,
//当且仅当该数的二进制形式的 1 出现的次数是奇数否则称作, evil.
//所以 1,2,4,7 是 odious 因为它们的二进制形式是 1,10,100,111.
//而 0,3,5,6 是 evil 因为它们的二进制形式是, 0,11,101,110.
//而上面那个表中貌似, sg 值都是 odious 数所以当. 2x 为 odious 时,
//sg 值是 2x, 当 2x 是 evil 时, sg 值是 2x+1.
//
//
//每次可以连续翻动任意个硬币至少翻一个,.( Ruler 游戏 )
//初始编号从 1 开始.
//那么这个游戏的 SG 函数是 g(n)=mex{0,g(n—1),g(n—1)^g(n—2)…,,g(n—1)…^^g(1)}
//根据 SG 函数可以得到 SG 值表如下.
//位置 x : 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15   16...
// g(x)  : 1  2  1  4  1  2  1  8  1  2    1   4   1   2   1   16...
//所以 sg 值为 x 的因数当中 2 的能达到的最大次幂比如. 14=2*7, 最大 1 次幂.
//即 2;16=2*2*2*2, 最大 4 次幂即, 16 .
```

4.2 数值计算

4.2.1 傅里叶变换

```
//FFT
//注意数组范围 可以使用万进制来加速
Ver 1.0
#include<complex>
typedef complex<double> cplx;
const double pi=acos(—1.0);
void FFT(cplx F[],int len,int f)
{
    int i,j,k;
    cplx x,y,w=1,wn;
    for(i=1,j=0,k=0;i<=len;i++)
    {
        if(j<k) swap(F[j],F[k]);
        j^=i&—i;
        k^=len/(i&—i)>>1;
    }
    for(i=1;i<len;i<<=1)
    {
```

```
        wn=cplx(cos(-f*pi/i),sin(-f*pi/i));
        for(j=0;j<len;j+=i<<1,w=1)
            for(k=j;k<j+i;k++,w*=wn)
            {
                x=F[k];y=w*F[k+i];
                F[k]=x+y;F[k+i]=x-y;
            }
    }
    if(f==-1)
        for(i=0;i<len;i++)F[i]/=len;
}
cplx a[500000],b[500000];
void conv(int ca[],int l1,int cb[],int l2,int c[],int &l)
{
    l=1<<(32-__builtin_clz(l1+l2));
    for(int i=0;i<l;i++)
    {
        a[i]=i<l1?cplx(ca[i],0):cplx(0,0);
        b[i]=i<l2?cplx(cb[i],0):cplx(0,0);
    }
    FFT(a,l,1);FFT(b,l,1);
    for(int i=0;i<l;i++)a[i]*=b[i];
    FFT(a,l,-1);
    for(int i=0;i<l;i++)c[i]=a[i].real()+0.5;
}
```

4.2.2 数论变换

```
//FNT 有点慢  = =
//g 为 mod 的原根

//1004535809 3
//211812353 3
//10000000025100289 22
//11000000009994241 17
//100000000135659521 3

const int mod=211812353,g=3;
int _g[30],_ig[30];
long long invl;
int pow_mod(long long a,int b)
{
    long long c=1;
    while(b)
    {
        if(b&1)c=c*a%mod;
        b>>=1;
        a=a*a%mod;
    }
    return c;
}
void init()
{
    for(int i=0;i<30;i++)
    {
        _g[i]=pow_mod(g,(mod-1)/(1<<i));
        _ig[i]=pow_mod(_g[i],mod-2);
    }
}
void FNT(int F[],int len,int f)
{
    int i,j,k,cnt=1;
    long long x,y,w=1,wn;
    for(i=1,j=0,k=0;i<=len;i++)
    {
        if(j<k) swap(F[j],F[k]);
        j^=i&-i;
        k^=len/(i&-i)>>1;
    }
    for(i=1;i<len;i<<=1)
    {
        wn=f?_g[cnt++]:_ig[cnt++];
        for(j=0;j<len;j+=i<<1,w=1)
            for(k=j;k<j+i;k++,w=w*wn%mod)
            {
                x=F[k];y=w*F[k+i]%mod;
                F[k]=x+y;F[k+i]=x-y;
                if(F[k]>=mod)F[k]-=mod;
                if(F[k+i]<0)F[k+i]+=mod;
            }
```

```
    }
    if(!f)
        for(i=0;i<len;i++)F[i]=F[i]*invl%mod;
}
int a[150000],b[150000];
void conv(int ca[],int l1,int cb[],int l2,int c[],int &l)
{
    l=1<<(32-__builtin_clz(l1+l2));
    init();invl=pow_mod(l,mod-2);
    for(int i=0;i<l;i++)
    {
        a[i]=i<l1?ca[i]:0;
        b[i]=i<l2?cb[i]:0;
    }
    FNT(a,l,1);FNT(b,l,1);
    for(int i=0;i<l;i++)a[i]=(long long)a[i]*b[i]%mod;
    FNT(a,l,0);
    for(int i=0;i<l;i++)c[i]=a[i];
}
```

4.2.3 位运算 FFT

```
异或
  : tf(x1,x2)=(tf(x1)-tf(x2),tf(x1)+tf(x2))
    utf(x1,x2)=(utf((x1+x2)/2),utf((x2-x1)/2))与
    : tf(x1,x2)=(tf(x1)+tf(x2),tf(x1))
    utf(x1,x2)=(utf(x1-x2),utf(x2))


//与
Ver. 递归
void tf(int a[],int l,int r)
{
    if(l+1==r)return ;
    int mid=(l+r)>>1,len=mid-l;
    tf(a,l,mid);tf(a,mid,r);
    for(int i=0;i<len;i++)
        a[l+i]+=a[mid+i];
}
void utf(int a[],int l,int r)
{
    if(l+1==r)return ;
    int mid=(l+r)>>1,len=mid-l;
    for(int i=0;i<len;i++)
        a[l+i]-=a[mid+i];
    utf(a,l,mid);utf(a,mid,r);
}
//异或
void tf(int a[],int l,int r)
{
    if(l+1==r)return ;
    int mid=(l+r)>>1,len=mid-l;
    tf(a,l,mid);tf(a,mid,r);
    for(int i=0;i<len;i++)
    {
        int x1=a[l+i];
        int x2=a[mid+i];
        a[l+i]=x1-x2;
        a[mid+i]=x1+x2;
    }
}
void utf(int a[],int l,int r)
{
    if(l+1==r)return ;
    int mid=(l+r)>>1,len=mid-l;
    for(int i=0;i<len;i++)
    {
        int x1=a[l+i];
        int x2=a[mid+i];
        a[l+i]=(x1+x2)/2;
        a[mid+i]=(x2-x1)/2;
    }
    utf(a,l,mid);utf(a,mid,r);
}
```

4.2.4 高斯消元

```
//一般方程高斯消元 好板
const int MAXN=50;
```

```
3  |
4  |int a[MAXN][MAXN];//增广矩阵
5  |int x[MAXN];//解集
6  |bool free_x[MAXN];//标记是否是不确定的变元 1 表示不确定
7  |inline int lcm(int n,int m)
8  |{
9  |    int a=n,b=m,c=n%m;
10 |    while(c)a=b,b=c,c=a%b;
11 |    return n/b*m;
12 |}
13 |int Gauss(int equ,int var)
14 |{
15 |    int i,j,k,max_r,col=0;
16 |    int temp,LCM,ta,tb;
17 |    int free_x_num,free_index;
18 |    for(i=0;i<=var;i++)
19 |        x[i]=0,free_x[i]=1;
20 |    for(k=0;k<equ && col<var;k++,col++)
21 |    {
22 |        max_r=k;
23 |        for(i=k+1;i<equ;i++)
24 |            if(abs(a[i][col])>abs(a[max_r][col])) max_r=i;
25 |        if(max_r!=k) for(j=k;j<var+1;j++) swap(a[k][j],a[max_r][j]);
26 |        if(a[k][col]==0)
27 |        {
28 |            k--;
29 |            continue;
30 |        }
31 |        for(i=k+1;i<equ;i++)
32 |            if(a[i][col]!=0)
33 |            {
34 |                LCM=lcm(abs(a[i][col]),abs(a[k][col]));
35 |                ta=LCM/abs(a[i][col]);tb=LCM/abs(a[k][col]);
36 |                if(a[i][col]*a[k][col]<0) tb=-tb;
37 |                for(j=col;j<var+1;j++)
38 |                    a[i][j]=a[i][j]*ta-a[k][j]*tb;
39 |            }
40 |    }
41 |    for(i=k;i<equ;i++)
42 |        if(a[i][col]!=0) return -1;
43 |    if(k<var)
44 |    {
45 |        for(i=k-1;i>=0;i--)
46 |        {
47 |            free_x_num=0;
48 |            for(j=0;j<var;j++)
49 |                if (a[i][j]!=0 && free_x[j]) free_x_num++,free_index=j;
50 |            if(free_x_num>1) continue;
51 |            temp=a[i][var];
52 |            for(j=0;j<var;j++)
53 |                if (a[i][j]!=0 && j!=free_index) temp-=a[i][j]*x[j];
54 |            x[free_index]=temp/a[i][free_index];
55 |            free_x[free_index]=0;
56 |        }
57 |        return var-k;  //自由变元个数
58 |    }
59 |    else  //唯一解
60 |    {
61 |        for(i=var-1;i>-1;i--)
62 |        {
63 |            temp=a[i][var];
64 |            for(j=i+1;j<var;j++)
65 |                if(a[i][j]!=0) temp-=a[i][j]*x[j];
66 |            if(temp%a[i][i]!=0) return -2; //浮点解
67 |            x[i]=temp/a[i][i];
68 |        }
69 |        return 0;
70 |    }
71 |}
```

4.2.5 矩阵行列式

```
1  |
2  |//取模版本
3  |int a[305][305];
4  |int mod=10007;
5  |int pow_mod(int a,int b)
6  |{
7  |    int c=1;
```

```
    a%=mod;
    while(b)
    {
        if(b&1) c=(c*a)%mod;
        b>>=1;
        a=(a*a)%mod;
    }
    return c;
}
int det(int n)
{
    int i,j,k,ans=1,div=1;
    for(i=0;i<n;i++)
    {
        k=i;
        for(j=i;j<n;j++)
            if(a[j][i])
            {
                k=j;
                break;
            }
        if(a[k][i]==0) return 0;
        if(k!=i) for(ans=mod-ans,j=i;j<n;j++) swap(a[i][j],a[k][j]);
        ans=(a[i][i]*ans)%mod;
        for(j=i+1;j<n;j++)
            if(a[j][i])
            {
                div=(div*a[i][i])%mod;
                for(k=n-1;k>=i;k--)
                    a[j][k]=((a[j][k]*a[i][i]-a[i][k]*a[j][i])%mod+mod)%mod;
            }
    }
    ans=(ans*pow_mod(div,mod-2))%mod;
    return ans;
}

//版本gcd
long long a[305][305];
long long det(int n)
{
    int i,j,k,pa,pb;
    long long ans=1,pt;
    for(i=0;i<n;i++)
    {
        k=i;
        for(j=i;j<n;j++)
            if(a[j][i])
            {
                k=j;
                break;
            }
        if(a[k][i]==0) return 0;
        if(k!=i)for(ans=-ans,j=i;j<n;j++) swap(a[i][j],a[k][j]);
        for(j=i+1;j<n;j++)
            if(a[j][i])
            {
                if(a[i][i]>a[j][i])pa=i,pb=j;
                else pa=j,pb=i;
                while(a[pb][i])
                {
                    swap(pa,pb);
                    pt=a[pb][i]/a[pa][i];
                    for(k=i;k<n;k++)
                        a[pb][k]-=a[pa][k]*pt;
                }
                if(i!=pa)for(ans=-ans,k=i;k<n;k++)swap(a[i][k],a[j][k]);
            }
        ans*=a[i][i];
    }
    return ans;
}
```

4.2.6 单纯形法

```
//sigma(j,a[i][j]*xj)+b[i]>=0  ans=max(sigma(i,c[i]*x[i]))
//所有其他形式的线性规划方程组都可以按照下列方式转化成标准形式：
//目标函数并非最大化将所有1.: ci 取负.
//约束条件中存在大于将约束两边取负2.:.
//约束条件中存在等式将其转化成两个不等式3.:.
//有的变量没有非负约束加入新变量4.: x' 并用, x-x' 替换原来的变量 x
```

```
7
8  //实数单纯形  z=a[0][0]+max(sigma(j,a[0][j]*x[j]))
9  //            0<=a[i][0]+sigma(j,a[i][j]*x[j])
10 //            n 为变元个数，m 为约束个数
11 //标准型要求  b[i]>=0
12
13 const int N=100;
14 const int M=100;
15 const double INF=1e20;
16 int sgn(double x)
17 {
18     return fabs(x)<1e-8?0:(x>0.0?1:-1);
19 }
20 struct simplex
21 {
22     int pf[N+M],m,n;
23     double a[M][N],c[N],z;
24     double x[N+M];
25     int next[N];
26
27     void pivot(int x,int y)
28     {
29         int las=-1;
30         swap(pf[x],pf[n+y]);
31         a[y][x]=1.0/a[y][x];
32         for(int i=0;i<=n;i++)
33             if(sgn(a[y][i])&&i!=x)next[i]=las,las=i,a[y][i]*=-a[y][x];
34         for(int i=0;i<=m;i++)
35             if(i!=y)
36             {
37                 for(int j=las;~j;j=next[j])
38                     a[i][j]+=a[y][j]*a[i][x];
39                 a[i][x]*=a[y][x];
40             }
41     }
42     bool opt()
43     {
44         int pc,tc,i;
45         double mc;
46         while(1)
47         {
48             for(tc=i=1;i<=n;i++)
49                 if(sgn(a[0][i]-a[0][tc])>0)tc=i;
50             if(sgn(a[0][tc])<=0)return z=a[0][0],1;
51             pc=-1;mc=INF;
52             for(i=1;i<=m;i++)
53                 if(sgn(a[i][tc])<0&&sgn(mc+a[i][0]/a[i][tc])>0)
54                     mc=-a[pc=i][0]/a[i][tc];
55             if(pc==-1)return 0;
56             pivot(tc,pc);
57         }
58     }
59     void pri()
60     {
61         for(int i=0;i<=m;i++,putchar(10))
62             for(int j=0;j<=n;j++)
63                 printf("%f ",a[i][j]);
64         putchar(10);
65     }
66     bool init()
67     {
68         double mc=1.0;
69         int tc;
70         for(int i=1;i<=n+m;i++)pf[i]=i;
71         for(int i=1;i<=m;i++)
72             if(sgn(mc-a[i][0])>0)mc=a[tc=i][0];
73         if(sgn(mc)<0)
74         {
75             for(int i=1;i<=n;i++)c[i]=a[0][i],a[0][i]=0;
76             a[0][++n]=-1;pf[n+m]=n+m;
77             for(int i=1;i<=m;i++)a[i][n]=1;
78             pivot(n,tc);opt();
79             if(sgn(z)<0)return 0;
80             for(int i=1;i<=n+m;i++)
81                 if(pf[i]==n)
82                     if(i>n)pivot(1,i-n),tc=1;
83                     else tc=i;
84             for(int i=0;i<=m;i++)a[i][tc]=0;
85             for(int i=1;i<=n;i++)if(pf[i]<n)a[0][i]+=c[pf[i]];
86             for(int i=n+1;i<=m+n;i++)
```

```
87                  if(pf[i]<n)
88                      for(int j=0;j<=n;j++)a[0][j]+=c[pf[i]]*a[i-n][j];
89          }
90          return 1;
91      }
92      int solv()
93      {
94          if(!init())return -1;      //无解
95          if(!opt())return 0;        //无限大
96          for(int i=1;i<=n;i++)
97              x[pf[i]]=0;
98          for(int i=n+1;i<=n+m;i++)
99              x[pf[i]]=a[i-n][0];
100         return 1;                  //有解
101     }
102 }g;
```

4.2.7 龙贝格积分

```
1  double f(double x)
2  {
3      return x*x;
4  }
5  const int MAXREPT=10;
6  const double eps=1e-5;
7  double y[MAXREPT];
8  double Romberg(double aa, double bb)
9  {
10     int m,n;
11     double h,x,s,q,ep,p;
12     h=bb-aa;
13     y[0]=h*(f(aa)+f(bb))/2.0;
14     m=n=1;ep=eps+1.0;
15     while ((ep > eps) && (m < MAXREPT))
16     {
17         p=0.0;
18         for(int i=0;i<n;i++)
19             x=aa+(i+0.5)*h,p+=f(x);
20         p=(y[0]+h*p)/2.0;s=1.0;
21         for(int k=1;k<=m;k++)
22         {
23             s*=4.0;
24             q=(s*p-y[k-1])/(s-1.0);
25             y[k-1]=p;p=q;
26         }
27         p=fabs(q-y[m-1]);
28         y[m++]=q;n<<=1;h/=2.0;
29     }
30     return q;
31 }
```

4.3 数论

4.3.1 快速幂

```
1  //a^b%c  b 特别大如果
2  eular(c)<b则
3   a^b%c=a ^(b%eular(c)+eular(c)) % c;
```

4.3.2 Pollard Rho

```
1  #include <cstdio>
2  #include <cstring>
3  #include <cstdlib>
4  #include <algorithm>
5  using namespace std;
6
7  unsigned __int64 factor[100005];
8  unsigned __int64 factorNumber;
9  unsigned __int64 factorCount[100005];
10
11 unsigned __int64 gcd(unsigned __int64 a,unsigned __int64 b)
12 {
13     if(b == 0)
14     {
15         return a;
16     }
17     return gcd(b, a % b);
```

```
}
unsigned __int64 mul_mod(unsigned __int64 a,unsigned __int64 b,unsigned __int64 n)
{
    unsigned __int64 exp = a % n;
    unsigned __int64 res = 0;
    while(b)
    {
        if(b&1)
        {
            res += exp;
            if(res > n)
            {
                res -= n;
            }
        }
        exp <<= 1;
        if(exp>n)
        {
            exp -= n;
        }
        b >>= 1;
    }
    return res;
}

unsigned __int64 exp_mod(unsigned __int64 a,unsigned __int64 b,unsigned __int64 c)
{
    unsigned __int64 k = 1;
    while(b)
    {
        if(b & 1)
        {
            k = mul_mod(k, a, c);
        }
        a = mul_mod(a, a, c);
        b >>= 1;
    }
    return k;
}

bool miller_rabbin(unsigned __int64 n, unsigned __int64 times)
{
    if(n == 2)
    {
        return 1;
    }
    if(n < 2 || !(n & 1))
    {
        return 0;
    }
    unsigned __int64 a, u = n - 1, x, y;
    int t = 0;
    while((u&1) == 0)
    {
        ++ t;
        u >>= 1;
    }
    for(int i=0;i<times;i++)
    {
        a = rand() % (n-1) + 1;
        x = exp_mod(a, u, n);
        for(int j=0;j<t;j++)
        {
            y = mul_mod(x, x, n);
            if(y == 1 && x != 1 && x != n-1)
            {
                return false;
            }
            x = y;
        }
        if(y != 1)
        {
            return false;
        }
    }
    return true;
}

unsigned __int64 pollard_rho(unsigned __int64 n,unsigned __int64 c)
{
```

```
98      unsigned __int64 x,y,d,i=1,k=2;
99      y = x = rand() % (n-1) + 1;
100     while(true)
101     {
102         ++ i;
103         x = (mul_mod(x, x, n) + c) % n;
104         d = gcd((x - y + n) % n, n);
105         if(d > 1 && d < n)
106         {
107             return d;
108         }
109         if(x == y)
110         {
111             return n;
112         }
113         if(i == k)
114         {
115             k <<= 1;
116             y = x;
117         }
118     }
119 }
120
121 void find_factor(unsigned __int64 n,unsigned __int64 c)
122 {
123     if(n == 1)
124     {
125         return;
126     }
127     if(miller_rabbin(n, 6))
128     {
129         factor[factorNumber++] = n;
130         return;
131     }
132     unsigned __int64 p = n;
133     unsigned __int64 k = c;
134     while(p>=n)
135     {
136         p = pollard_rho(p, c--);
137     }
138     find_factor(p, k);
139     find_factor(n / p, k);
140 }
141
142 int main()
143 {
144     int caseNumber;
145     scanf("%d", &caseNumber);
146     while(caseNumber--)
147     {
148         unsigned __int64 n;
149         scanf("%I64u", &n);
150         factorNumber = 0;
151         find_factor(n, 120);
152         sort(factor, factor + factorNumber);
153         factorCount[0] = 1;
154         int k = 1;
155         for(int i=1;i<factorNumber;++i)
156         {
157             if(factor[i] == factor[i-1])
158             {
159                 ++ factorCount[k - 1];
160             }
161             else
162             {
163                 factorCount[k] = 1;
164                 factor[k++] = factor[i];
165             }
166         }
167         factorNumber = k;
168         printf("%d␣", k);
169         unsigned __int64 sum = 0;
170         for(int i=0;i<factorNumber;++i)
171         {
172             unsigned __int64 temp = 1;
173             for(int j=0;j<factorCount[i];++j)
174             {
175                 temp *= factor[i];
176             }
177             sum += temp;
```

```
        }
        if(factorNumber == 1)
        {
            sum /= factor[0];
        }
        printf("%I64u\n", sum);
    }
    return 0;
}
```

4.3.3 扩展欧几里得

```
void e_gcd(int a,int b,int &x,int &y)
{
  if(b==0)x=1,y=0;
  else e_gcd(b,a%b,y,x),y-=a/b*x;
}
```

4.3.4 欧拉函数 & 莫比乌斯函数

```
//欧拉函数 与 N 互质的个数
//预处理
//Ver 2.3;
int prim[100005],phi[100005];
int pk;
bool vis[100005];
void cal()
{
    phi[1]=1;
    for(int i=2;i<100001;i++)
    {
        if(!vis[i]) prim[pk++]=i,phi[i]=i-1;
        for(int j=0;j<pk && i*prim[j]<100001;j++)
        {
            vis[i*prim[j]]=1;
            if(i%prim[j]) phi[i*prim[j]]=phi[i]*(prim[j]-1);
            else
            {
                phi[i*prim[j]]=phi[i]*prim[j];
                break;
            }
        }
    }
}
//Ver 1.0;
void cal()
{
  phi[1]=1;
  for(int i=2;i<5000005;i++)
     if(!phi[i])
       for(int j=i;j<5000005;j+=i)
      {
        if(!phi[j])phi[j]=j;
        phi[j]=phi[j]/i*(i-1);
      }
}
//直接算 可先筛素数来优化
int phi(int s)
{
  int ret=1;
  for(int i=2;i*i<=s;i++)
    if(s%i==0)
    {
      s/=i,ret*=i-1;
      while(s%i==0) s/=i,ret*=i;
    }
    if(s>1) ret*=s-1;
    return ret;    //n 的欧拉数
}
ans=N*phi(N)/2      //求小于 N 的与 N 互质的数的和
phi(p^k)=p^k-p^(k-1)=(p-1)*p^(k-1)
phi(n)=phi(p1^a1)*phi(p2^a2)*phi(p3^a3)*...*phi(pn^an)


//莫比乌斯函数
//预处理
int prim[100005],mu[100005];
int pk;
bool vis[100005];
```

```
void cal()
{
    mu[1]=1;
    for(int i=2;i<100001;i++)
    {
        if(!vis[i]) prim[pk++]=i,mu[i]=-1;
        for(int j=0;j<pk && i*prim[j]<100001;j++)
        {
            vis[i*prim[j]]=1;
            if(i%prim[j]) mu[i*prim[j]]=-mu[i];
            else
            {
                mu[i*prim[j]]=0;
                break;
            }
        }
    }
}
//直接算 可先筛素数来优化
int mu(int s)
{
    int ret=1;
    for(int i=2;i*i<=s;i++)
        if(s%i==0)
        {
            s/=i,ret=-ret;
            if(s%i==0)return 0;
        }
    if(s>1) ret=-ret;
    return ret;
}
//F(n) = sigma (G(d))   d | n
//那么 G(n) = sigma (F(d) * miu (n / d))  d | n
//还有另外一个表达形式
//F(n) = sigma (G(d))  n | d
//G(n) = sigma (F(d) * miu (d / n))  n | d
```

4.3.5 区间 gcd 统计

```
//统计 x=[1,a],y=[1,b],gcd(x,y)=k 的对数这里保证类似   (1,3),(3,1) 只算一次法一


//枚举 x 分解素因子，通过容斥原理求出，[1,b] 与 x 互素的个数

int dfs(int b,int s) //容斥原理
{
    if(b<2) return 0;
    int ans=0;
    while(s<fk)
    {
        ans+=b/fac[s]-dfs(b/fac[s],s+1);
        s++;
    }
    return ans;
}

long long ans;
if(k==0) ans=0;
else
{
    a/=k;b/=k;
    if(a>b) swap(a,b);
    ans=0;
    for(int i=1;i<=a;i++)
    {
        fk=0;k=i;
        for(int j=0;vis[k];j++)
            if(k%prim[j]==0)
            {
                fac[fk++]=prim[j];
                while(k%prim[j]==0) k/=prim[j];
            }
        if(k>1) fac[fk++]=k;
        ans+=b-i+1-(dfs(b,0)-dfs(i-1,0));
    }
}法二


//fac[i] 表示 gcd(x,y)=i 的对数利用  fac[i]=(n/i)*(m/i)-fac[2*i]-fac[3*i] 去重..
```

```
if(k==0) ans=0;
else
{
    long long n,m;
    a/=k;b/=k;
    if(a>b) swap(a,b);
    fac[1]=0;
    for(int i=a;i>0;i--)
    {
        n=a/i,m=b/i;
        n=(2*m-n+1)*n/2;
        for(int j=i+i;j<=a;j+=i)
            n-=fac[j];
        fac[i]=n;
    }
    ans=fac[1];
}法三



//莫比乌斯反演 还有分块优化()
//定义
//F(n) = sigma (G(d))   d | n
//那么 G(n) = sigma (F(d) * miu (n / d))  d | n
//还有另外一个表达形式
//F(n) = sigma (G(d))  n | d
//G(n) = sigma (F(d) * miu (d / n))  n | d

int main()
{
    for(int i=1;i<100001;i++)mu[i]+=mu[i-1];
    int a,b,k;
    if(k==0)printf("Case␣%d:␣0\n",ti++);
    else
    {
        a/=k;b/=k;
        if(a>b)swap(a,b);
        long long sum=0,sum2=0;
        for(int i=1;i<=a;i=k+1)
        {
            k=min(a/(a/i),b/(b/i));
            sum+=1ll*(mu[k]-mu[i-1])*(a/i)*(b/i);
            sum2+=1ll*(mu[k]-mu[i-1])*(a/i)*(a/i);
        }
        printf("%I64d\n",sum-sum2/2);
    }
    return 0;
}
```

4.3.6 求原根

```
Ver 3.0

struct Root
{
    int prim[230000],k;
    bool vis[1000050];
    void cal()
    {
        for(int i=2;i<1000005;i++)
        {
            if(vis[i]==0) prim[k++]=i;
            for(int j=0;j<k && prim[j]*i<1000005;j++)
            {
                vis[prim[j]*i]=1;
                if(i%prim[j]==0) break;
            }
        }
    }
    int fac[40],fk;
    void getFactor(long long c)
    {
        fk=0;
        for(int i=0;i<k&&(long long)prim[i]*prim[i]<=c;i++)
            if(c%prim[i]==0)
            {
                fac[fk++]=prim[i];
                do c/=prim[i];
                while(c%prim[i]==0);
            }
```

```
        if(c!=1)
            fac[fk++]=c;
    }
    long long pow_mod(long long a,long long b,long long mod)
    {
        long long c=1;
        while(b)
        {
            if(b&1)c=c*a%mod;
            b>>=1;
            a=a*a%mod;
        }
        return c;
    }
    int getPriRoot(long long p)
    {
        if(p==2)return 1;
        long long phi=p-1;
        getFactor(phi);
        for(int g=2;g<p;++g)
        {
            bool f=1;
            for(int i=0;f&&i<fk;i++)
                if(pow_mod(g,phi/fac[i],p)==1)
                    f=0;
            if(f)
                return g;
        }
    }
}g;
```

4.3.7 取模 & 逆元

```
//一，普通
//(a/b) mod m 等于 a%(b*m)/b   //不能先取模a
//二，逆元
//1. e_gcd 取逆元
//调用 e_gcd(a,b,inv,d);
//inv 为 a 模 b 的逆元
//即有 (num/a)%b = (num*inv)%b;
//
//2. 欧拉函数取逆元
//费马定理 a^(p-1)=1(mod p) p 素数
//欧拉定理 a^phi[p]=1(mod p) a 与 p 互素
//
//3. 快速幂取逆元
//%1 如果 mod 为素数有   inv=pow_mod(a,mod-2,mod);
//%2 任意互素数有  inv=pow_mod(a,phi[mod]-1,mod);
//
//4. 不互素
//其实只需要把答案看做两部分的乘积：一部分是与 m 互素的，这一部分的乘法直接计算，
//除法改成乘逆元就行了；另一部分是若干个 m 的素因子的乘积，因为 m<1,000,000,000 ，
//所以 m 的不同素因子不会太多，用一个数组记录每一个素因子的数量就行。这一部分的
//乘法就是把记录的素因子数量相加，除法就是把记录的素因子数量相减。最后计算这两
//部分的乘积对 m 的取模，也就是 h(n)%m
//
//5. 线性求逆元
//f[i] 为 i 的逆元
fac[0]=inv[0]=f[0]=1;
fac[1]=inv[1]=f[1]=1;
for(int i=2;i<10100005;i++)
{
  f[i]=(-mod/i)*f[mod%i]%mod;
  if(f[i]<0)f[i]+=mod;
  inv[i]=inv[i-1]*f[i]%mod;
  fac[i]=fac[i-1]*i%mod;
}
```

4.3.8 中国剩余定理

```
 //x=a1 mod b1
//x=a2 mod b2
//x=a3 mod b3
//.....
//x=ai mod bi
//x=M1'M1a1+M2'.....M2a2+Mk'MKak(mod m)   m=b1*b2*.....b3bi
//Mi'Mi= 1(mod mi) 即    Mi'Mi+mi*y=1 求逆用扩展欧几里得解出 Mi'

```

```
//flag=0 无解最后解是  r
void e_gcd(long long a,long long b,long long &x,long long &y)
{
    if(b==0) x=1,y=0;
    else e_gcd(b,a%b,y,x),y-=a/b*x;
}

long long r=0,div=1,a,b,x,y,gcd;
bool flag=1;
for(int i=0;i<n;i++)
{
    scanf("%lld%lld",&b,&a); //b 是除数 a 是余数
    e_gcd(div,b,x,y);
    if(flag)
    {
        gcd=div*x+b*y;
        if((r-a)%gcd) flag=0;
        else
        {
            x=(a-r)/gcd*x%(b/gcd);
            r=(div*x+r)%(div/gcd*b);
            if(r<0) r+=div/gcd*b;
            div*=b/gcd;
        }
    }
}
if(flag) printf("%lld\n",r);
else printf("-1\n");
```

4.3.9 最小素因子

```
int prim[20005],k;
int mi[20005];
bool vis[20005];
void cal()
{
  for(int i=2;i<20001;i++)
  {
    if(vis[i]==0) prim[k++]=i,mi[i]=i;
    for(int j=0;j<k && prim[j]*i<20005;j++)
    {
      vis[prim[j]*i]=1;
      mi[prim[j]*i]=prim[j];
      if(i%prim[j]==0) break;
    }
  }
}
```

4.4 平面几何

4.4.1 常用函数

```
//asin(y/x) 范围 [-pi/2,pi/2]
//acos(y/x) 范围 [0,pi]
//atan(y/x) 范围 [-pi/2,pi/2]
//atan2(y,x) 范围 [-pi,pi] x 轴正向为 0

struct Point
{
    double x,y;
    Point(double a=0.0,double b=0.0){x=a;y=b;}
    Point operator+(const Point&a)const{return Point(x+a.x,y+a.y);}
    Point operator-(const Point&a)const{return Point(x-a.x,y-a.y);}
    Point operator*(const double&a)const{return Point(x*a,y*a);}
    Point operator/(const double&a)const{return Point(x/a,y/a);}
    double operator*(const Point&a)const{return x*a.y-y*a.x;}
    double operator/(const Point&a)const{return sqrt((a.x-x)*(a.x-x)+(a.y-y)*(a.y-y));}
    double operator%(const Point&a)const{return x*a.x+y*a.y;}
    double arg(){return atan2(y,x);}        //[-pi,pi] x 轴正向逆时针为正
};    //点类

struct Line
{
    int a,b,c;   //a*x+b*y+c=0 满足( a>0 或 a=0,b>0 且为最简 )
    Line(Point p0=Point(1,1),Point p1=Point())  //必须保证两点不重合
    {
        a=p1.y-p0.y;b=p0.x-p1.x;
        c=(p0.y-p1.y)*p0.x+(p1.x-p0.x)*p0.y;  //double 的话到这
```

```
27          int d=max(abs(a),abs(b));
28          d=a?__gcd(d,a):d;d=b?__gcd(d,b):d;
29          d=abs(c?__gcd(d,c):d);
30          if(a<0 || (a==0 && b<0)) d=-d;
31          a/=d;b/=d;c/=d;
32      }
33      Point intersect(Line l)    //求直线交点 要保证不平行或重合
34      {
35          return Point(1.0*(b*l.c-l.b*c)/(l.b*a-b*l.a),
36                       1.0*(l.a*c-a*l.c)/(a*l.b-l.a*b));
37      }
38  };   //直线类解析式()
39
40  struct Line
41  {
42      Point s,t;
43      double arg;
44      Line(){}
45      Line(Point a,Point b){s=a;t=b;arg=atan2(b.y-a.y,b.x-a.x);}
46      Point operator&(const Line&a)    //求直线交点 要保证不平行或重合
47      {return (t-s)*((s-a.s)*(a.s-a.t))/((s-t)*(a.s-a.t))+s;}
48      bool operator<(const Line&a)
49      {
50          if(fabs(arg-a.arg)>eps)return arg<a.arg;
51          else return (a.s-s)*(t-s)>0;
52      }
53  };   //直线类两点式()
54
55  const double eps = 1e-8;
56  inline int sgn(double x)
57  {
58      return fabs(x)<eps?0:(x>0.0?1:-1);
59  }
60
61  inline Point line_cro(Point p1,Point p2,Point p3,Point p4)  //直线求交点
62  {
63      return (p2-p1)*((p1-p3)*(p3-p4))/((p1-p2)*(p3-p4))+p1;
64  }
65
66  inline bool pos_line(Point p0,Point p1,Point p2)  //p0 是否在 p1p2 上
67  {
68      return sgn((p1-p0)*(p2-p0))==0 && sgn((p1-p0)%(p2-p0))<=0;
69  }
70
71  inline bool cross(Point p0,Point p1,Point p2,Point p3) //判断线段 p0p1,p2p3 是否相交端点相交也算
72  {
73      if(min(p0.x,p1.x) <= max(p2.x,p3.x) && min(p2.x,p3.x) <= max(p0.x,p1.x) &&
74         min(p0.y,p1.y) <= max(p2.y,p3.y) && min(p2.y,p3.y) <= max(p0.y,p1.y))
75         return sgn(((p1-p0)*(p2-p0))*((p1-p0)*(p3-p0)))<=0 && sgn(((p3-p2)*(p0-p2))*((p3-p2)*(p1-p2)))<=0;
76      else return 0;
77  }
78
79  inline bool cross(Point p0,Point p1,Point p2,Point p3) //判断线段 p0p1,p2p3 是否相交端点相交不算
80  {
81      return sgn(((p1-p0)*(p2-p0))*((p1-p0)*(p3-p0)))<0 && sgn(((p3-p2)*(p0-p2))*((p3-p2)*(p1-p2)))<0;
82  }
83
84  inline Point rota(Point p0,Point p1,double deg)  //p1 绕 p0 顺时针 deg 弧度
85  {
86      Point n=p1-p0;
87      return p0+Point(n.x*cos(deg)+n.y*sin(deg),n.y*cos(deg)-n.x*sin(deg));
88  }
89
90  bool inpol(Point pol[],int n,Point p1)
91  {
92      int cnt=0,i;
93      Point p2;pol[n]=pol[0];
94      for(i=n;i>0;i--)
95          if(sgn((pol[i-1]-p1)*(pol[i]-p1))==0&&sgn((pol[i-1]-p1)%(pol[i]-p1))!=1)
96              return 1;
97      while(i<n)
98      {
99          p2=Point(rand()+maxn,rand()+maxn);
100         for(i=0;i<n;i++)
101         {
102             if(!sgn((p2-p1)*(pol[i]-p1))){cnt=0;break;}
103             if(sgn((p2-pol[i])*(pol[i+1]-pol[i]))*sgn((p1-pol[i])*(pol[i+1]-pol[i]))==-1
104                &&sgn((p2-p1)*(pol[i+1]-p1))*sgn((p2-p1)*(pol[i]-p1))==-1)cnt++;
105         }
```

```
    }
    return cnt&1;
}

double R;  //球半径
double sph_dis(double lat1,double lng1,double lat2,double lng2) //纬度 经度
{
    double radLat1=lat1*pi/180.0;
    double radLat2=lat2*pi/180.0;
    double a=radLat1-radLat2;
    double b=lng1*pi/180.0-lng2*pi/180.0;
    double s=2*asin(sqrt(sin(a/2)*sin(a/2)+cos(radLat1)*cos(radLat2)*sin(b/2)*sin(b/2)));
    return s*R;
}
```

4.4.2 最近点对

```
//O(nlognlogn)
struct Point
{
    double x,y;
    bool operator<(Point a)const{return fabs(x-a.x)<1e-8?y<a.y:x<a.x;}
}po[100005],te[100005];
double dis(Point &a,Point &b)
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
double near_pair(int l,int r)
{
    if(l==r) return 1e10;
    if(l==r-1) return dis(po[l],po[r]);
    int mid=(l+r)>>1,k=0;
    double d=min(near_pair(l,mid),near_pair(mid+1,r));
    for(int i=mid;i>=l && po[mid].x-po[i].x<d;i--)
        te[k].x=po[i].y,te[k++].y=po[i].x;
    for(int i=mid+1;i<=r && po[i].x-po[mid].x<d;i++)
        te[k].x=po[i].y,te[k++].y=po[i].x;
    sort(te,te+k);
    for(int i=0;i<k;i++)
        for(int j=1;j<5 && i+j<k;j++)
            d=min(d,dis(te[i],te[i+j]));
    return d;
}
```

4.4.3 最近圆对

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

const double eps=1e-9;
int sgn(double x)
{
    return fabs(x)<eps?0:(x>0.0?1:-1);
}
struct p
{
    double x,y,r;
}po[50050];
int n;
pair<double,int>pa[50050],pb[50050];
#include<set>
struct sp
{
    int i;
    sp(int k=0){i=k;}
    bool operator<(const sp&a)const{return sgn(po[i].y-po[a.i].y)?po[i].y<po[a.i].y:po[i].x<po[a.i].x;}
};
inline bool cross(p a,p b,double mid)
{
    return sgn(sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y))-a.r-b.r-mid*2)<0;
}
set<sp>::iterator it,itt;
set<sp>st;
inline bool ok(double mid)
```

```
{
    int p=0,a,b;
    st.clear();
    for(int i=0;i<=n;i++)
    {
        while(p<n&&pb[p].first+mid<pa[i].first-mid)
        {
            it=itt=st.find(sp(pb[p++].second));
            itt++;
            if(it!=st.begin()&&itt!=st.end())
            {
                a=(--it)->i;
                b=itt->i;
                if(cross(po[a],po[b],mid))return 1;
            }
            st.erase(--itt);
        }
        st.insert(sp(b=pa[i].second));
        it=itt=st.find(sp(b));
        if(it!=st.begin())
        {
            a=(--it)->i;
            if(cross(po[a],po[b],mid))return 1;
        }
        itt++;
        if(itt!=st.end())
        {
            a=itt->i;
            if(cross(po[a],po[b],mid))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%lf",&po[i].x,&po[i].y,&po[i].r);
        for(int i=0;i<n;i++)
        {
            pa[i]=make_pair(po[i].x-po[i].r,i);
            pb[i]=make_pair(po[i].x+po[i].r,i);
        }
        sort(pa,pa+n);sort(pb,pb+n);
        pa[n]=make_pair(1e10,n);
        pb[n]=make_pair(1e10,n);
        double l=0,r=1e5,mid;
        while(r-l>1e-7)
        {
            mid=(l+r)/2;
            if(ok(mid))r=mid;
            else l=mid;
        }
        printf("%.6f\n",r*2);
    }
    return 0;
}
```

4.4.4 最小圆覆盖

```
//期望 O(n)

Point circumcenter(Point a,Point b,Point c)
{
    double a1=b.x-a.x,b1=b.y-a.y,c1 =(a1*a1+b1*b1)/2;
    double a2=c.x-a.x,b2=c.y-a.y,c2 =(a2*a2+b2*b2)/2;
    double d=a1*b2-a2*b1;
    return Point(a.x+(c1*b2-c2*b1)/d,a.y+(a1*c2-a2*c1)/d);
}
void min_circle(Point h[],int n,Point &c,double &r)
{
    int i,j,k;
    r=0.0;c=h[0];
    for(i=1;i<n;i++)
        if(h[i]/c>r)
            for(j=0,c=h[i],r=0.0;j<i;j++)
                if(h[j]/c>r)
```

```
                    for(k=0,c=(h[i]+h[j])/2.0,r=h[j]/c;k<j;k++)
                        if(h[k]/c>r)c=circumcenter(h[i],h[j],h[k]),r=h[k]/c;
}
```

4.4.5 凸包

```
//1. 求凸包
//longlong 版本
struct Point
{
    long long x,y;
    Point(long long a=0,long long b=0){x=a;y=b;}
    Point operator-(const Point&a)const{return Point(x-a.x,y-a.y);}
    Point operator+(const Point&a)const{return Point(x+a.x,y+a.y);}
    Point operator*(const long long&a)const{return Point(x*a,y*a);}
    long long operator*(const Point&a)const{return x*a.y-y*a.x;}
    Point operator/(const long long&a)const{return Point(x/a,y/a);}
    double operator/(const Point&a)const{return sqrt((a.x-x)*(a.x-x)+(a.y-y)*(a.y-y));}
    bool operator<(const Point&a)const
    {
        long long temp=*this*a;
        return temp>0||(temp==0&&x*x+y*y<a.x*a.x+a.y*a.y);
    }
};
int convex(Point h[],int n,Point ans[],int &k)
{
    for(int i=1;i<n;i++)
        if(h[0].y>h[i].y||(h[0].y==h[i].y&&h[0].x>h[i].x))swap(h[0],h[i]);
    Point sta=h[0];
    for(int i=0;i<n;i++)h[i]=h[i]-sta;
    sort(h+1,h+n);k=min(2,n);
    for(int i=0;i<n;i++)h[i]=h[i]+sta;
    ans[0]=h[0];ans[1]=h[1];
    for(int i=2;i<n;i++)
    {
        while(k>1&&(ans[k-1]-ans[k-2])*(h[i]-ans[k-2])<=0)k--;
        ans[k++]=h[i];
    }
}

//double 版本
struct Point
{
    double x,y;
    Point(double a=0,double b=0){x=a;y=b;}
    Point operator-(const Point&a)const{return Point(x-a.x,y-a.y);}
    Point operator+(const Point&a)const{return Point(x+a.x,y+a.y);}
    Point operator*(const double&a)const{return Point(x*a,y*a);}
    double operator*(const Point&a)const{return x*a.y-y*a.x;}
    Point operator/(const double&a)const{return Point(x/a,y/a);}
    double operator/(const Point&a)const{return sqrt((a.x-x)*(a.x-x)+(a.y-y)*(a.y-y));}
    bool operator<(const Point&a)const
    {
        double temp=*this*a;
        return sgn(temp)==1||(sgn(temp)==0&&sgn(a.x*a.x+a.y*a.y-x*x-y*y)==1);
    }
};
int convex(Point h[],int n,Point ans[],int &k)
{
    for(int i=1;i<n;i++)
        if(sgn(h[0].y-h[i].y)==1||(sgn(h[0].y-h[i].y)==0&&sgn(h[0].x-h[i].x)==1))swap(h[0],h[i]);
    Point sta=h[0];
    for(int i=0;i<n;i++)h[i]=h[i]-sta;
    sort(h+1,h+n);k=min(2,n);
    for(int i=0;i<n;i++)h[i]=h[i]+sta;
    ans[0]=h[0];ans[1]=h[1];
    for(int i=2;i<n;i++)
    {
        while(k>1&&sgn((ans[k-1]-ans[k-2])*(h[i]-ans[k-2]))!=1)k--;
        ans[k++]=h[i];
    }
}

//2. 最远点对(旋转卡壳)
long long ans=0;
int p=1;
hul[k]=hul[0];
for(int i=0;i<k;i++)
{
    while(fabs((hul[i+1]-hul[i])*(hul[p+1]-hul[i]))>fabs((hul[p]-hul[i])*(hul[i+1]-hul[i])))
```

```
        p==k-1?p=0:p++;
    ans=max(ans,max(hul[i+1]/hul[p],hul[i]/hul[p]));
}

//3. 最小矩形(三个旋转卡壳)
int p=1,p1=1,p2=1;
double ans=1e10,a,b;
hul[k]=hul[0];
for(int i=0;i<k;i++)
{
    while(abs((hul[i+1]-hul[i])*(hul[p+1]-hul[i]))>abs((hul[i+1]-hul[i])*(hul[p]-hul[i])))
        p==k-1?p=0:p++;
    while((hul[p1+1]-hul[p1])%(hul[i+1]-hul[i])>0)
        p1==k-1?p1=0:p1++;
    while((hul[i+1]-hul[i])*(hul[p2+1]-hul[p2])>0||(hul[p2+1]-hul[p2])%(hul[i+1]-hul[i])<0)
        p2==k-1?p2=0:p2++;
    a=2*abs((hul[i+1]-hul[i])*(hul[p]-hul[i]))/(hul[i]/hul[i+1]);
    b=2*abs((hul[p1]-hul[p2])%(hul[i+1]-hul[i]))/(hul[i]/hul[i+1]);
    ans=min(a+b,ans);
}

//4. 两个凸包最近点对
inline double p2l_dist(Point p1,Point p2,Point p3)
{
    Point d=Point(-(p3-p2).y,(p3-p2).x);
    if(sgn(d*(p2-p1))==sgn(d*(p3-p1)))return min(p1/p2,p1/p3);
    else
    {
        d=d/(d/Point(0,0));
        return fabs((p3-p1)%d);
    }
}
inline double seg_dist(Point p1,Point p2,Point p3,Point p4)
{
    double t1=min(p2l_dist(p1,p3,p4),p2l_dist(p2,p3,p4));
    double t2=min(p2l_dist(p3,p1,p2),p2l_dist(p4,p1,p2));
    return min(t1,t2);
}
double convex_dist(Point h1[],int n,Point h2[],int m)
{
    int p1=0,p2=0,cnt=n+1;
    for(int i=1;i<n;i++)
        if(sgn(h1[p1].y-h1[i].y)>0)p1=i;
    for(int i=1;i<m;i++)
        if(sgn(h2[p2].y-h2[i].y)<0)p2=i;
    h1[n]=h1[0];h2[m]=h2[0];
    double ans=1e10;
    while(cnt--)
    {
        ans=min(ans,seg_dist(h1[p1],h1[p1+1],h2[p2],h2[p2+1]));
        while(sgn((h2[p2+1]-h2[p2])*(h1[p1+1]-h1[p1]))<0)
        {
            p2==m-1?p2=0:p2++;
            ans=min(ans,seg_dist(h1[p1],h1[p1+1],h2[p2],h2[p2+1]));
        }
        p1==n-1?p1=0:p1++;
    }
    return ans;
}
```

4.4.6 三角形心

```
//□□□□
Point Circumcenter(Point a,Point b,Point c)
{
    double a1 = b.x - a.x, b1 = b.y - a.y, c1 = (a1*a1 + b1*b1)/2;
    double a2 = c.x - a.x, b2 = c.y - a.y, c2 = (a2*a2 + b2*b2)/2;
    double d = a1 * b2 - a2 * b1;
    return Point(a.x + (c1*b2 - c2*b1)/d,a.y + (a1*c2 - a2*c1)/d);
}

//′'□□
Point Orthocenter(Point p0,Point p1,Point p2)
{
    double a1,b1,a2,b2,c1,c2;
    a1 = p2.x-p1.x; b1=p2.y-p1.y;c1 = 0;
    a2 = p2.x-p0.x; b2=p2.y-p0.y;c2 = (p1.x-p0.x)*a2+(p1.y-p0.y)*b2;
    double d = a1 * b2 - a2 * b1;
    return Point(p0.x+(c1*b2-c2*b1)/d,p0.y+(a1*c2-a2*c1)/d);
}
```

```

//□□□□
//μ½□□½□□□□□¶¥μ□□□□□□ · ½°□□□□μj□
//
Point Barycenter(Point a,Point b,Point c)
{
  return (a+b+c)/3;
}

//□□□□
Point Incenter(Point p1,Point p2,Point p3)
{
    double a=dis(p2,p3),b=dis(p1,p3),c=dis(p1,p2);
    return Point((a*p1.x+b*p2.x+c*p3.x)/(a+b+c),(a*p1.y+b*p2.y+c*p3.y)/(a+b+c));
}
```

4.4.7 费马点

```

//(1)若三角形的ABC 3 个内角均小于 120 °，那么 3 条距离连线正好三
// 等分费马点所在的周角。所以三角形的费马点也称为三角形的等角中心。

//(2)若三角形有一内角不小于 120 度，则此钝角的顶点就是距离和最小的点。


const double eps=1e-8;
const double pi=acos(-1.0);
struct point_t
{
    double x, y;
    point_t() { }
    point_t(double tx, double ty) : x(tx), y(ty) { }
    point_t operator-(const point_t &r) const
    {
        return point_t(x - r.x, y - r.y);
    }
    point_t operator+(const point_t &r) const
    {
        return point_t(x + r.x, y + r.y);
    }
    point_t operator*(const double r) const
    {
        return point_t(x * r, y * r);
    }
    point_t operator/(const double r) const
    {
        return point_t(x / r, y / r);
    }
} p[4];

int dblcmp(double x)
{
    return (x < -eps ? -1 : x > eps);
}

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

double cross(point_t p1, point_t p2)
{
    return p1.x * p2.y - p1.y * p2.x;
}

double dot(point_t p1, point_t p2)
{
    return p1.x * p2.x + p1.y * p2.y;
}

double angle(point_t p1, point_t p2, point_t p3)
{
    double d = dot(p2 - p1, p3 - p1) / dist(p1, p2) / dist(p1, p3);
    return d;
}

point_t inter(point_t a, point_t b, point_t c, point_t d)
{
    point_t p1 = b - a, p2 = d - c;
    double a1 = p1.y, b1 = -p1.x, c1;
```

```
    double a2 = p2.y, b2 = -p2.x, c2;
    c1 = a1 * a.x + b1 * a.y;
    c2 = a2 * c.x + b2 * c.y;
    return point_t((c1 * b2 - c2 * b1) / (a1 * b2 - a2 * b1), (c1 * a2 - c2 * a1) / (b1 * a2 - b2 * a1));
}

point_t ver(point_t p1, point_t p2)
{
    point_t v = (p2 - p1) / 2 * sqrt(3.0);
    swap(v.x, v.y);
    v.x = -v.x;
    return v;
}

point_t fermat(point_t p1, point_t p2, point_t p3)
{
    if (2*angle(p1, p2, p3) < -1) return p1;
    if (2*angle(p2, p1, p3) < -1) return p2;
    if (2*angle(p3, p1, p2) < -1) return p3;
    point_t v1 = ver(p1, p2);
    point_t m1 = (p1 + p2) / 2;
    if (dblcmp(cross(p3 - p1, p2 - p1)) * dblcmp(cross(v1 + m1 - p1, p2 - p1)) > 0) v1.x = -v1.x, v1.y = -v1.y;
    m1 = m1 + v1;
    point_t v2 = ver(p1, p3);
    point_t m2 = (p1 + p3) / 2;
    if (dblcmp(cross(p2 - p1, p3 - p1)) * dblcmp(cross(v2 + m2 - p1, p3 - p1)) > 0) v2.x = -v2.x, v2.y = -v2.y;
    m2 = m2 + v2;
    return inter(p3, m1, p2, m2);
}

int main()
{
    freopen("1.txt","r",stdin);
    int a,b,c,d,e,f;
    while(scanf("%d%d%d%d%d%d",&a,&b,&c,&d,&e,&f)!=-1)
    {
        if(a==c && b==d && a==e && b==f) printf("%.6lf␣%.6lf\n",1.0*a,1.0*b);
        else if(a==c && b==d) printf("%.6lf␣%.6lf\n",1.0*a,1.0*b);
        else if(a==e && b==f) printf("%.6lf␣%.6lf\n",1.0*a,1.0*b);
        else if(c==e && d==f) printf("%.6lf␣%.6lf\n",1.0*c,1.0*d);
        else
        {
            p[0].x=1.0*a;p[0].y=1.0*b;p[1].x=1.0*c;p[1].y=1.0*d;p[2].x=1.0*e;p[2].y=1.0*f;
            point_t f=fermat(p[0],p[1],p[2]);
            printf("%.6lf␣%.6lf\n",f.x,f.y);
        }
    }
    return 0;
}



//另变步长法(近似算法):

struct point
{
    double x,y;
}p[3];
double dis(point a,point b)
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
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(dis(u,a)+dis(u,b)+dis(u,c)>dis(v,a)+dis(v,b)+dis(v,c))
                        u.x=v.x,u.y=v.y;
```

```
                }
    return u;
}
int main()
{
 //   freopen("1.txt","r",stdin);
    while(scanf("%lf%lf%lf%lf%lf%lf",&p[0].x,&p[0].y,&p[1].x,&p[1].y,&p[2].x,&p[2].y)==6)
    {
        point a=fermentpoint(p[0],p[1],p[2]);
        printf("%.6lf %.6lf\n",a.x,a.y);
    }
}
```

4.4.8 半平面交

```
// P0-P1 左侧半平面交离线   O(nlogn)
//判可行域 用面积比较好
const double pi=acos(-1.0);
struct Line
{
    Point s,t;
    double arg;
    Line(){}
    Line(Point a,Point b){s=a;t=b;arg=atan2(b.y-a.y,b.x-a.x);}
    Point operator&(const Line&a)const{return (t-s)*((s-a.s)*(a.s-a.t))/((s-t)*(a.s-a.t))+s;}
    bool operator<(const Line&a)const
    {
        if(fabs(arg-a.arg)>1e-6)return arg<a.arg;
        else return (a.s-s)*(t-s)>1e-9;
    }
};
int halfplaneintersection(Line v[],int n,Point ans[])
{
    int front=0,tail=0;
    sort(v,v+n);q[0]=v[0];
    for(int i=1;i<n;i++)
        if(fabs(v[i-1].arg-v[i].arg)>1e-6)
        {
            while(front<tail&&(ans[tail-1]-v[i].s)*(v[i].t-v[i].s)>-1e-9)tail--;
            while(front<tail&&(ans[front]-v[i].s)*(v[i].t-v[i].s)>-1e-9)front++;
            if(fabs(q[tail].arg+pi-v[i].arg)<1e-6)return -1;
            ans[tail]=v[i]&q[tail];q[++tail]=v[i];
        }
    while(front<tail&&(ans[tail-1]-q[front].s)*(q[front].t-q[front].s)>-1e-9)tail--;
    ans[tail]=q[tail]&q[front];
    for(int i=front;i<=tail;i++)ans[i-front]=ans[i];
    return tail-front+1;
}


// P1-P2 左侧半平面交在线   O(n^2)
//k 为半平面的点数为  0 无解

Point crosspoint(int i,Point p1,Point p2)
{
    double f=xmult(p1-p2,conv[i]-p2);
    f=f/(f-xmult(p1-p2,conv[i+1]-p2));
    return conv[i]+(conv[i+1]-conv[i])*f;
}
void cut(Point p1,Point p2)
{
    int s=0,f;
    conv[k]=conv[0];
    for(int i=0;i<k;i++)
    {
        f=sgn(xmult(p2-p1,conv[i]-p1));
        if(f!=-1) te[s++]=conv[i];
        if(f*sgn(xmult(p2-p1,conv[i+1]-p1))<0) te[s++]=crosspoint(i,p1,p2);
    }
    for(int i=0;i<s;i++) conv[i]=te[i];
    k=s;
}
```

4.4.9 很多圆

```
//两圆交点 返回   0 无解
bool cir_cross(Point p1,double r1,Point p2,double r2,Point ans[])
{
    double d=p1/p2,cosa,sina;
```

```
    if(sgn(d-r1-r2)==1||sgn(fabs(r1-r2)-d)==1)return 0;
    cosa=(r1*r1+d*d-r2*r2)/(2*r1*d);
    sina=sqrt(max(0.0,1.0-cosa*cosa));
    Point ve=(p2-p1)*r1/d;
    ans[0]=p1+Point(ve.x*cosa+ve.y*sina,ve.y*cosa-ve.x*sina);
    ans[1]=p1+Point(ve.x*cosa-ve.y*sina,ve.y*cosa+ve.x*sina);
    return 1;
}

//两圆面积交 两圆面积并可以减这个实现
double cir_inter(Point a,double ra,Point b,double rb)
{
    double d=a/b;
    if(ra>rb)swap(ra,rb);
    if(sgn(ra+rb-d)!=1)return 0.0;
    if(sgn(d+ra-rb)!=1)return pi*ra*ra;
    double p1=2*acos((ra*ra+d*d-rb*rb)/(2.0*ra*d));
    double p2=2*acos((rb*rb+d*d-ra*ra)/(2.0*rb*d));
    return ((p1-sin(p1))*ra*ra+(p2-sin(p2))*rb*rb)*0.5;
}

//圆与圆心三角形面积交
double cir_tri(Point c,double r,Point p1,Point p2)
{
    bool f=1;p1=p1-c;p2=p2-c;
    double sum=p1*p2,h,r1,r2,r3;
    if(sgn(sum)==0)return 0.0;
    if(sgn(sum)==-1)swap(p1,p2),f=0;
    r2=p1/Point(0,0);r3=p2/Point(0,0);r1=(p1%p2)/(r2*r3);
    Point v,s=p2-p1,t1=Point(-s.y,s.x),t2=t1*(p1*(p2-p1))/(t1*(p2-p1));
    v=s/(p1/p2);h=t2.x*t2.x+t2.y*t2.y;
    if(sgn(r*r-h)!=1)sum=acos(r1)*r*r;
    else
    {
        int d1,d2;h=sqrt(r*r-h);
        t1=t2-v*h;t2=t2+v*h;
        r2=(p1%t1)/((t1/Point(0,0))*r2);
        r3=(p2%t2)/((t2/Point(0,0))*r3);
        d1=sgn(p1*t2)==1?sgn(p1*t1)==1:-1;
        d2=sgn(p2*t2)==1?sgn(p2*t1)==1:-1;
        if(d1*d2==1)sum=acos(r1)*r*r;
        else if(d1*d2==-1)sum=acos(r2)*r*r+acos(r3)*r*r+t1*t2;
        else if(d1!=0)sum=acos(r2)*r*r+t1*p2;
        else if(d2!=0)sum=p1*t2+acos(r3)*r*r;
        else sum=p1*p2;
    }
    return (f?sum:-sum)*0.5;
}

//单位圆最大点覆盖
const double pi=acos(-1.0);
pair<double,bool>arg[605];
int cir_point(Point po[],int n,double r)
{
    double d,th,dth,p1,p2;
    int ans=0;
    for(int i=0;i<n;i++)
    {
        int cnt=0,k=0,tot=0;
        for(int j=0;j<n;j++)
            if(i!=j&&sgn((d=po[i]/po[j])-r-r)==-1)
            {
                dth=acos(d*0.5/r);
                th=atan2(po[j].y-po[i].y,po[j].x-po[i].x);
                p1=th-dth;p2=th+dth;
                if(sgn(p1+pi)==-1)p1+=2*pi,cnt++;
                if(sgn(p2-pi)==1)p2-=2*pi,cnt++;
                arg[k++]=make_pair(p1,0);arg[k++]=make_pair(p2,1);
            }
        sort(arg,arg+k);
        for(int j=0;j<k;j++)
            if(arg[j].second)cnt--;
            else tot=max(tot,++cnt);
        ans=max(tot+1,ans);
    }
    return ans;
}

//多圆面积并
const double eps = 1e-7;
```

```
85  const double pi=acos(-1.0);
86  pair<double,bool>arg[2005];
87  double cir_union(Point c[],double r[],int n)
88  {
89      double sum=0.0,sum1=0.0,d,p1,p2,p3;
90      for(int i=0;i<n;i++)
91      {
92          bool f=1;
93          for(int j=0;f&&j<n;j++)
94              if(i!=j&&sgn(r[j]-r[i]-c[i]/c[j])!=-1)f=0;
95          if(!f)swap(r[i],r[--n]),swap(c[i--],c[n]);
96      }
97      for(int i=0;i<n;i++)
98      {
99          int k=0,cnt=0;
100         for(int j=0;j<n;j++)
101             if(i!=j&&sgn((d=c[i]/c[j])-r[i]-r[j])<=0)
102             {
103                 p3=acos((r[i]*r[i]+d*d-r[j]*r[j])/(2.0*r[i]*d));
104                 p2=atan2(c[j].y-c[i].y,c[j].x-c[i].x);
105                 p1=p2-p3;p2=p2+p3;
106                 if(sgn(p1+pi)==-1)p1+=2*pi,cnt++;
107                 if(sgn(p2-pi)==1)p2-=2*pi,cnt++;
108                 arg[k++]=make_pair(p1,0);arg[k++]=make_pair(p2,1);
109             }
110         if(k)
111         {
112             sort(arg,arg+k);
113             p1=arg[k-1].first-2*pi;
114             p3=r[i]*r[i];
115             for(int j=0;j<k;j++)
116             {
117                 p2=arg[j].first;
118                 if(cnt==0)
119                 {
120                     sum+=(p2-p1-sin(p2-p1))*p3;
121                     sum1+=(c[i]+Point(cos(p1),sin(p1))*r[i])*(c[i]+Point(cos(p2),sin(p2))*r[i]);
122                 }
123                 p1=p2;
124                 arg[j].second?cnt--:cnt++;
125             }
126         }
127         else sum+=2*pi*r[i]*r[i];
128     }
129     return (sum+fabs(sum1))*0.5;
130 }
131
132
133 //多圆面积并Ver 2.0
134 //ans1 , ans2 初始化
135 double ans[1005],ans2[1005];
136 const double eps=1e-7;
137 const double pi=acos(-1.0);
138 pair<double,bool>arg[2005];
139 void cir_union(Point c[],double r[],int n)
140 {
141     double d,p1,p2,p3;
142     for(int i=0;i<n;i++)
143     {
144         int k=0,cnt=1;
145         for(int j=0;j<n;j++)
146             if(i!=j&&sgn((d=c[i]/c[j])-r[i]-r[j])<=0)
147             {
148                 if(sgn(d+r[i]-r[j])<=0)cnt++;
149                 else if(sgn(d+r[j]-r[i])<=0);
150                 else
151                 {
152                     p3=acos((r[i]*r[i]+d*d-r[j]*r[j])/(2.0*r[i]*d));
153                     p2=atan2(c[j].y-c[i].y,c[j].x-c[i].x);
154                     p1=p2-p3;p2=p2+p3;
155                     if(sgn(p1+pi)==-1)p1+=2*pi,cnt++;
156                     if(sgn(p2-pi)==1)p2-=2*pi,cnt++;
157                     arg[k++]=make_pair(p1,0);arg[k++]=make_pair(p2,1);
158                 }
159             }
160         if(k)
161         {
162             sort(arg,arg+k);
163             p1=arg[k-1].first-2*pi;
164             p3=r[i]*r[i];
```

```
            for(int j=0;j<k;j++)
            {
                p2=arg[j].first;
                ans[cnt]+=(c[i]+Point(cos(p1),sin(p1))*r[i])*(c[i]+Point(cos(p2),sin(p2))*r[i]);
                ans2[cnt]+=(p2-p1-sin(p2-p1))*p3;
                p1=p2;
                arg[j].second?cnt--:cnt++;
            }
        }
        else ans2[cnt]+=2*pi*r[i]*r[i];
    }
    for(int i=1;i<=n;i++)
        ans[i]=(fabs(ans[i])+ans2[i])*0.5;
    //ans[i为覆盖]次及以上的面积i
}
```

4.5 三维几何

4.5.1 常用函数

```
struct Point_3
{
    double x,y,z;
    Point_3(double a=0,double b=0,double c=0){x=a;y=b;z=c;}
    Point_3 operator+(const Point_3&a)const{return Point_3(x+a.x,y+a.y,z+a.z);}
    Point_3 operator-(const Point_3&a)const{return Point_3(x-a.x,y-a.y,z-a.z);}
    Point_3 operator*(const double&a)const{return Point_3(x*a,y*a,z*a);}
    Point_3 operator/(const double&a)const{return Point_3(x/a,y/a,z/a);}
    Point_3 operator*(const Point_3&a)const{return Point_3(y*a.z-z*a.y,z*a.x-x*a.z,x*a.y-y*a.x);}
    double operator/(const Point_3&a)const{return sqrt((a.x-x)*(a.x-x)+(a.y-y)*(a.y-y)+(a.z-z)*(a.z-z));}
    double operator%(const Point_3&a)const{return x*a.x+y*a.y+z*a.z;}
};

Point_3 rota_3(Point_3 p,Point_3 v, double arg)     //右手螺旋 arg 弧度 v 必须是单位向量
{
    double rot[3][3],cs=cos(arg),si=sin(arg);
    rot[0][0]=cs+(1-cs)*v.x*v.x;
    rot[0][1]=(1-cs)*v.x*v.y-si*v.z;
    rot[0][2]=(1-cs)*v.x*v.z+si*v.y;
    rot[1][0]=(1-cs)*v.y*v.x+si*v.z;
    rot[1][1]=cs+(1-cs)*v.y*v.y;
    rot[1][2]=(1-cs)*v.y*v.z-si*v.x;
    rot[2][0]=(1-cs)*v.z*v.x-si*v.y;
    rot[2][1]=(1-cs)*v.z*v.y+si*v.x;
    rot[2][2]=cs+(1-cs)*v.z*v.z;
    return Point_3(p.x*rot[0][0]+p.y*rot[0][1]+p.z*rot[0][2],
                   p.x*rot[1][0]+p.y*rot[1][1]+p.z*rot[1][2],
                   p.x*rot[2][0]+p.y*rot[2][1]+p.z*rot[2][2]);
}

Point_3 projection(Point_3 z,Point_3 vec,Point_3 p,Point_3 dir)  // vec 单位面法向量 dir 单位投影向量 *****
{
    int f=sgn(z*dir);
    if(f==0)return Point_3(1e20,1e20,1e20);
    else
    {
        double s=(z-p)%vec;
        return p+s*dir*f;
    }
}
```

4.5.2 三维凸包

```
const double eps=1e-9;
int sgn(double x){return fabs(x)<eps?0:(x>0.0?1:-1);}
struct Hull_3
{
    int n,cnt; //初始化 n 点在内部体积为正面法向量向内  ()
    Point_3 ply[1005];
    int a[5000],b[5000],c[5000];
    bool ok[5000];
    bool v[5000];
    int q[5000];
    int vis[1005][1005];
    inline double S(const Point_3&a,const Point_3&b,const Point_3&c)
    {return (b-a)*(c-a)/Point_3(0,0,0);}
    inline double V(const Point_3&a,const Point_3&b,const Point_3&c,const Point_3&d)
    {return (b-a)*(c-a)%(d-a);}
```

```
    inline int ply_p(int f,const Point_3&p)
    {return sgn(V(ply[a[f]],ply[b[f]],ply[c[f]],p));}/*
    void deal(int p,int p1,int p2)
    {
        int f=vis[p1][p2];
        if(ok[f])
        {
            if(ply_p(f,ply[p])==1) dfs(p,f);
            else
            {
                a[cnt]=p2;b[cnt]=p1;c[cnt]=p;ok[cnt]=1;
                vis[p][p2]=vis[p1][p]=vis[p2][p1]=cnt++;
            }
        }
    }
    void dfs(int p,int f)
    {
        ok[f]=0;
        deal(p,b[f],a[f]);
        deal(p,c[f],b[f]);
        deal(p,a[f],c[f]);
    }*/
    inline bool deal(int p,int p1,int p2)
    {
        int f=vis[p1][p2];
        if(ok[f]&&v[f]==0)
        {
            if(ply_p(f,ply[p])==1)return 1;
            else
            {
                a[cnt]=p2;b[cnt]=p1;c[cnt]=p;ok[cnt]=1;
                vis[p][p2]=vis[p1][p]=vis[p2][p1]=cnt++;
            }
        }
        return 0;
    }
    void bfs(int p,int f)
    {
        int tail=0,front=1;
        memset(v,0,sizeof(v));
        q[0]=f;v[f]=1;
        while(tail!=front)
        {
            f=q[tail++];ok[f]=0;
            if(deal(p,b[f],a[f]))v[q[front++]=vis[b[f]][a[f]]]=1;
            if(deal(p,c[f],b[f]))v[q[front++]=vis[c[f]][b[f]]]=1;
            if(deal(p,a[f],c[f]))v[q[front++]=vis[a[f]][c[f]]]=1;
        }
    }
    bool same(int s,int e)
    {
        return ply_p(s,ply[a[e]])==0
             &&ply_p(s,ply[b[e]])==0
             &&ply_p(s,ply[c[e]])==0;
    }
    bool construct()
    {
        int i,j;
        if(n<4)return 0;
        bool f=1;
        for(i=1;i<n;i++)
            if(sgn(ply[0]/ply[i]))
            {
                swap(ply[1],ply[i]);
                f=0;break;
            }
        if(f)return 0;f=1;
        for(i=2;i<n;i++)
            if(sgn(S(ply[0],ply[1],ply[i])))
            {
                swap(ply[2],ply[i]);
                f=0;break;
            }
        if(f)return 0;f=1;
        for(i=3;i<n;i++)
            if(sgn(V(ply[0],ply[1],ply[2],ply[i])))
            {
                swap(ply[3],ply[i]);
                f=0;break;
            }
```

```
96          if(f)return 0;cnt=0;
97          for(i=0;i<4;i++)
98          {
99              a[cnt]=(i+1)%4;b[cnt]=(i+2)%4;c[cnt]=(i+3)%4;ok[cnt]=1;
100             if(ply_p(cnt,ply[i])==1) swap(b[cnt],c[cnt]);
101             vis[a[cnt]][b[cnt]]=vis[b[cnt]][c[cnt]]=vis[c[cnt]][a[cnt]]=cnt;
102             cnt++;
103         }
104         for(i=4;i<n;i++)
105         {
106             for(j=0;j<cnt;j++)
107                 if(ok[j]&&ply_p(j,ply[i])==1)
108                 {
109                     bfs(i,j);break;
110                 }
111         }
112         int k=cnt;
113         cnt=0;
114         for(i=0;i<k;i++)
115             if(ok[i])
116                 a[cnt]=a[i],b[cnt]=b[i],c[cnt]=c[i],ok[cnt++]=1;
117         return 1;
118     }
119     double area()
120     {
121         double ret=0;
122         for(int i=0;i<cnt;i++)
123             ret+=S(ply[a[i]],ply[b[i]],ply[c[i]]);
124         return fabs(ret*0.5);
125     }
126     double volume()
127     {
128         double ret=0;
129         for(int i=1;i<cnt;i++)
130             ret+=V(ply[a[0]],ply[a[i]],ply[b[i]],ply[c[i]]);
131         return fabs(ret/6);
132     }
133     int facepolygon()
134     {
135         int ret=0,i,j;
136         bool f;
137         for(i=0;i<cnt;i++)
138         {
139             for(j=0,f=1;j<i;j++)
140                 if(same(i,j)){f=0;break;}
141             if(f)ret++;
142         }
143         return ret;
144     }
145     double dis(int f,const Point_3&p1)
146     {
147         return V(ply[a[f]],ply[b[f]],ply[c[f]],p1)/S(ply[a[f]],ply[b[f]],ply[c[f]]);
148     }
149     Point_3 Centre()
150     {
151         Point_3 ret=Point_3(0,0,0);
152         double sum=0.0,z;
153         for(int i=0;i<cnt;i++)
154         {
155             z=V(ply[a[0]],ply[a[i]],ply[b[i]],ply[c[i]]);
156             ret=(ply[a[0]]+ply[a[i]]+ply[b[i]]+ply[c[i]])/4.0*z+ret;
157             sum+=z;
158         }
159         return ret/sum;
160     }
161 }h;
```

4.6 序列

4.6.1 图数列

```
1 n 个点 n-1 条边 n^(n-2)
2
3 n 个点 n 条边
4
5     long long it=1,ans;
6     for(int i=3;i<n;i++)
7         it*=i;
8     ans=it;
```

```
    for(int i=1;i<=n-3;i++)
    {
        it=it*n/i;
        ans+=it;
    }
    printf("%I64d\n",ans);生成树计数


Krichhoof 矩阵的 n-1 阶行列式的值
Krichhoof 矩阵 G 是这样的：
Gii 等于点 i 的度数当
 i 和 j 有边时， Gij = -1 。否则   Gij 等于 0 。
```

4.6.2 组合数列

```
//卡特兰数
h[n∑]=h[i]*h[n-1-i]
  h[n]=c[2*n][n]/(n+1);
h(n)=c(2n,n)-c(2n,n+1);
h(n)=h(n-1)*(4*n-2)/(n+1);

//斯特林数
    for(int i=1;i<30;i++)
        S[i][1]=1;
    for(int i=2;i<30;i++)
        for(int j=2;j<30;j++)
            S[i][j]=S[i-1][j-1]+j*S[i-1][j];

//欧拉辗转数
dp[1][1]=1;
for(int i=2;i<23;i++)
    for(int j=1;j<i;j++)
        dp[i][j]=dp[i][j-1]+dp[i-1][i-j];
```

4.6.3 数列

```
//卡特兰数前几项为 : 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786
//斐波那契数列 F(n) 1 1 2 3 5 8 13 21 34 55
//卢卡斯数列 L(n) 1 3 4 7 11 18 29 47 76 123
//F(n)*L(n) 1 3 8 21 55 144 377 987 2584 6765

//斐波那契
1)(f(n+1),f(n))=1;
2)f(m+n)=f(m-1)*f(n)+f(m)*f(n+1)
3)(f(m+n),f(n))=(f(n),f(m))
4)f(kx)%f(x)==0
5)(f(m),f(n))=f((m,n))

//斯特林数
  1
  1   1
  1   3   1
  1   7   6   1
  1  15  25  10   1
  1  31  90  65  15   1
  1  63 301 350 140  21   1
  1 127 966 1701 1050 266  28   1
  1 255 3025 7770 6951 2646 462  36   1

//欧拉辗转数
    1
    1     0
    0     1     0
    0     1     1     0
    0     1     2     2     0
    0     2     4     5     5     0
    0     5    10    14    16    16     0
    0    16    32    46    56    61    61     0
    0    61   122   178   224   256   272   272     0
    0   272   544   800  1024  1202  1324  1385  1385     0
    0  1385  2770  4094  5296  6320  7120  7664  7936  7936     0
    0  7936 15872 23536 30656 36976 42272 46366 49136 50521 50521     0
```

4.7 组合数学

4.7.1 Polya

I. $L=\frac{1}{|G|}(m^{c(g1)}+m^{c(g2)}+m^{c(g3)}...)$

II. c(gi) 为置换 gi 的循环的个数 (i=1...s)

```
//题意：用 k 种颜色对 n 个珠子构成的环上色，旋转翻转后相同的只算一种，求不等价的着色方案数。
//
//Burnside 定理的应用：
//
//当 n 为奇数时，有 n 种翻转，每种翻转都是以一个顶点和该顶点对边的中点对称。有 k^(n/2+1)*n 种。
//
//当 n 为偶数时，有 n 种翻转，其中一半是以两个对应顶点，另一半是以两条对边对称。有 k^(n/2+1)*n/2+k^(n/2)*n/2 种。
//
//考虑旋转 ：枚举旋转角度 360/n*i ， (0<i<=n) ，也就是一个置换。经过该置换，颜色仍保持不变的着色方案有     k^GCD(n,i) 种。
//
//一个长度为的环n ，每个上同一种颜色 i ，可以上多少种颜色。
//
//假设起点在 x，则 x,x+i,x+2*i……,,x+k*i……,
//
//假设在第 t 次，第一次回到起点，则 x=(x+t*i)%n => t*i%n=0 => t=LCM(i,n)/i=n*i/GCD(n,i)/i=n/GCD(n,i)。
//
//那么可以上 n/t 种颜色，即 n/(n/GCD(n,i)) 种，所以旋转的着色方案有 k^GCD(n,i) 种。

long long cal(int a,int b)
{
    long long c=1;
    while(b--)c*=a;return c;
}
int gcd(int a,int b)
{
    if(b==0)return a;
    while(a&&b)
    {
        if(a>b)a%=b;
        else b%=a;
    }
    return a+b;
}
int main()
{
    int s;
    while(scanf("%d",&s)&&s!=-1)
    {
        long long ans=0;
        if(s==0){puts("0");continue;}
        for(int i=0;i<s;i++)
            ans+=cal(3,gcd(s,i));
        if(s&1)ans+=s*cal(3,s/2+1);
        else ans+=s/2*(cal(3,s/2+1)+cal(3,s/2));
        printf("%lld\n",ans/s/2);
    }
    return 0;
}
```

4.7.2 小球 _ 盒子

```
//( n 球 k 盒子)
//
//盒子不相同 小球相同不为空  C[n-1][k-1]
//
//盒子不相同 小球相同可为空  C[n+k-1][k-1]
//
//盒子相同 小球不同不为空   S[n][k]
//    for(int i=1;i<30;i++)
//        S[i][1]=1;
//    for(int i=2;i<30;i++)
//        for(int j=2;j<30;j++)
//            S[i][j]=S[i-1][j-1]+j*S[i-1][j];
//
//盒子相同 小球相同可为空 Σ   S[n][i] (k>=i>=1)
//
//盒子不同 小球不同不为空  k!*S[n][k]
//
//盒子不同 小球不同可为空  k^n
//
//盒子相同 小球相同可空盒  dp[n][k]
//    for(int i=0;i<402;i++)
//        dp[0][i]=1;
//    for(int i=1;i<50002;i++)
```

```
//        for(int j=1;j<402;j++)
//        {
//            if(i>=j) dp[i][j]=dp[i-j][j];
//            if(j>0) dp[i][j]=(dp[i][j]+dp[i][j-1])%mod;
//        }
//
//盒子相同 小球不同不为空  dp[n-k][k]
```

4.8 其他

4.8.1 反素数

```
反素数定义对于任何正整数


x其约数的个数记做,g(x)例如.g(1)=1,g(6)如果某个正整数=4.满足x对于任意:i(0<i<x)都有,g(i)<g(x)则称,为反素数x.性质性质——
    个反素数的质因子必然是从开始连续的质数

:2.性质二
:p=2^t1*3^t2*5^t3*7^t4必然.....t1>=t2>=t3>=....
```

4.8.2 高维球体积

```

//(pi^(n/2)*r^n)/gama(n/2);

double V(int n,double r)
{
    return pow(pi,n/2.0)*pow(r,n)/tgamma(n/2.0+1);
}
```

4.8.3 阶乘最后非 0 位

```
//
//...
int mod[20]={1,1,2,6,4,2,2,4,2,8,4,4,8,4,6,8,8,6,8,2};
char num[10005];
int a[10005];

int main()
{
    int len,t,c,i;
    while(scanf("%s",num)==1)
    {
        t=1;
        len=strlen(num);
        for(i=0;i<len;i++)
            a[i]=num[len-i-1]-'0';
        while(len)
        {
            len-=!a[len-1];
            t=t*mod[a[1]%2*10+a[0]]%10;
            for(c=0,i=len-1;i>-1;i--)
            {
                c=c*10+a[i];
                a[i]=c/5;
                c%=5;
            }
        }
        printf("%d\n",t);
    }
    return 0;
}
```

4.8.4 整数拆分积最大不允许重复

```
//整数拆分积最大 不允许重复
int data[100];
int n=8;
int main()
{
    int k = 2;
    for(; n >= k; n-=k,k++)
    data[k] = k;
    for(int i = k-1; i >= 2 && n; i--, n--)
    data[i]++;
    data[k-1] += n;
```

```
12      for(int j = 2; j < k; j++)
13          printf("%d␣",data[j]);
14      putchar(10);
15      return 0;
16  }
```

4.8.5 矩阵中三角形个数统计

```
1   //O(n*m)
2   int gcd(int a,int b)
3   {
4       int c=a%b;
5       while(c)
6       {
7           a=b;b=c;
8           c=a%b;
9       }
10      return b;
11  }
12  long long C(int a)
13  {
14      long long temp=a;
15      return temp*(temp-1)*(temp-2)/6;
16  }
17  int main()
18  {
19      int n,m;
20      while(scanf("%d%d",&n,&m)==2)
21      {
22          m++;n++;
23          long long ans=C(n*m)-C(n)*m-C(m)*n,temp;
24          m--;n--;
25          for(int i=2;i<=n;i++)
26              for(int j=2;j<=m;j++)
27              {
28                  temp=gcd(i,j);
29                  ans-=(temp-1)*(n-i+1)*(m-j+1)*2;
30              }
31          printf("%lld\n",ans);
32      }
33      return 0;
34  }
```

4.8.6 求和

```
1   int n2(int a)
2   {return a*(a+1)*(2*a+1)/6;}
3   int n3(int a)
4   {return a*a*(a+1)*(a+1)/4;}
5   int n4(int a)
6   {return a*(a+1)*(2*a+1)*(3*a*a+3*a-1)/30;}
7
8   Fn=( √ 5/5)*{[(1+ √ 5)/2]^n - [(1- √ 5)/2]^n}
9
10  C[n+1]=2*a*C[n]-(a*a-b)*C[n-1]
11  C[n]=(a+sqrt(b))^n+(a-sqrt(b))^n
12
13  //欧拉常数  0.57721566490153286060651209
14  //
15  //公式：PICKS=I+E/2-1  S面积，,多边形的内坐标点的个数，多边形的边上坐标点的个数。IE
16  //
17  //
18  //欧拉公式
19  //简单多面体的顶点数、面数及棱数间有关系VFE
20  //V+F-E=2
21  //
22  //错排
23  //Dn=(n)-1[D(n)-2+D(n)-1]
24  //Dn=[n!/e+0.5]
25  //
26  //n^k+(n-1)^k+....+2^k+1^k 通项系数
27  //S(n, 1) = (1 + ... + n) = (1/2) * n2 + (1/2) * n
28  //S(n, 2) = (1 + ... + n2) = (1/3) * n3 + (1/2) * n2 + (1/6) * n
29  //S(n, 3) = (1 +...+ n3) = (1/4) * n4 + (1/2) * n3) + (1/4) * n2
30  //S(n, 4) = (1 +...+ n4) = (1/5) * n5 + (1/2) * n4) + (1/3) * n3 - (1/30) * n
31  //
32  //系数
33  //1
34  //1/2  1/2
```

```
//1/6  1/2  1/3
//0  1/4  1/2  1/4
//−1/30  0  1/3  1/2  1/5
//0  −1/12  0  5/12  1/2  1/6
//1/42  0  −1/6  0  1/2  1/2  1/7


struct p
{
    long long x,y;
    bool an;
}s[555][555];
long long pp;
int gcd(long long a,long long b)
{
    pp=a%b;
    while(pp)
    {
        a=b;b=pp;
        pp=a%b;
    }
    return b;
}
p f(int a,int b)
{
    if(s[a][b].an) return s[a][b];
    long long y;
    if(b)
  {
    s[a][b]=f(a−1,b−1);
    s[a][b].x*=a;s[a][b].y*=b+1;
  }
    else
    {
        p temp,c;
        temp.x=0;temp.y=1;
        for(int i=1;i<=a;i++)
        {
            c=f(a,i);
      if(temp.x*c.y+c.x*temp.y!=0)
      {
        y=gcd(temp.y*c.y,temp.x*c.y+c.x*temp.y);
        temp.x=(temp.x*c.y+c.x*temp.y)/y;
        temp.y=temp.y*c.y/y;
      }
        }
        s[a][b].x=temp.y−temp.x;
        s[a][b].y=temp.y;
    }
    y=gcd(s[a][b].x,s[a][b].y);
    s[a][b].x=s[a][b].x/y;
    s[a][b].y=s[a][b].y/y;
    s[a][b].an=1;
    if(s[a][b].y<0) s[a][b].y=−s[a][b].y,s[a][b].x=−s[a][b].x;
    return s[a][b];
}
```

4.8.7 约瑟夫问题

```
//可用递归推导
int n, m, i, s=0;
scanf("%d%d", &n, &m);
for (i=2; i<=n; i++) s=(s+m)%i;
printf ("The␣winner␣is␣%d\n", s+1);
```

5 迷

5.1 DLX Exact

```
VER 2.3
//矩阵标号从开始1
//g.init(COL_SIZE);
//g.insert(x,y);
//g.DLX_DFS(0);

const int MAXN=1005;
const int MAXM=1005;
const int INF=0X3fffffff;
struct DLX_Exact
{
    int D[MAXN*MAXM],U[MAXN*MAXM],L[MAXN*MAXM],R[MAXN*MAXM],COL[MAXN*MAXM],ROW[MAXN*MAXM];
    int CNT,BEG[MAXN],END[MAXN],USE[MAXN],_USE[MAXN];
    int SUM[MAXM];
    bool has;
  int step;
    void init(int n)
    {
        memset(BEG,0xff,sizeof(BEG));
        for(int i=1;i<=n;i++)
            SUM[L[i+1]=R[i-1]=D[i]=U[i]=i]=0;
        L[L[1]=R[n]=0]=n;CNT=n+1;
        has=step=0;
    }
    void insert(int r,int c)
    {
        D[CNT]=D[c];U[CNT]=c;U[D[c]]=CNT;D[c]=CNT;
        COL[CNT]=c;ROW[CNT]=r;SUM[c]++;
        if(BEG[r]==-1)BEG[r]=END[r]=CNT;
        R[END[r]]=CNT;L[CNT]=END[r];R[CNT]=BEG[r];
        L[BEG[r]]=CNT;END[r]=CNT++;
    }
    void DLX_Remove(int c)
    {
        L[R[c]]=L[c];
        R[L[c]]=R[c];
        for(int i=D[c];i!=c;i=D[i])
            for(int j=R[i];j!=i;j=R[j])
                U[D[j]]=U[j],D[U[j]]=D[j],SUM[COL[j]]--;
    }
    void DLX_Resume(int c)
    {
        for(int i=U[c];i!=c;i=U[i])
            for(int j=L[i];j!=i;j=L[j])
                U[D[j]]=j,D[U[j]]=j,SUM[COL[j]]++;
        L[R[c]]=c;
        R[L[c]]=c;
    }
    void DLX_Dfs(int n)
    {
        if(R[0]==0)
        {
            has=1;
            for(int i=0;i<n;i++)
                USE[i]=_USE[i];
            return;
        }
        int i,c=R[0];
        for(i=R[0];i;i=R[i])
            if(SUM[c]>SUM[i])c=i;
        DLX_Remove(c);
        for(i=D[c];i!=c&&!has;i=D[i])
        {
            for(int j=R[i];j!=i;j=R[j])
                DLX_Remove(COL[j]);
            _USE[n]=ROW[i];
            DLX_Dfs(n+1);
            for(int j=L[i];j!=i;j=L[j])
                DLX_Resume(COL[j]);
        }
        DLX_Resume(c);
    }
}g;

```

```
75
76
77
78  VER 1.0
79
80
81  //矩阵标号从开始1
82  //g.n=ROW_SIZE;
83  //g.m=COL_SIZE;
84  //g.init();
85  //g.insert(x,y);
86  //g.initDFS();
87  //g.DFS(0);
88
89
90  const int MaxN=402;
91  const int MaxM=402;
92  struct DLX
93  {
94      int h,n,m,tot;
95      int U[MaxN*MaxM],D[MaxN*MaxM],L[MaxN*MaxM],R[MaxN*MaxM],Row[MaxN*MaxM],Col[MaxN*MaxM];
96      int S[MaxM],O[MaxN];
97      bool hasans;
98      void init()
99      {
100         hasans=h=0;
101         tot=m+n;
102         for(int i=0;i<=m;i++)
103         {
104             D[i]=U[i]=Col[i]=i;
105             Row[i]=S[i]=0;
106             L[i]=(i+m)%(m+1);
107             R[i]=(i+1)%(m+1);
108         }
109         for(int i=1;i<=n;i++)
110         {
111             R[i+m]=L[i+m]=i+m;
112             Row[i+m]=i;
113             Col[i+m]=0;
114         }
115     }
116     void insert(int x,int y)
117     {
118         tot++;
119         Row[tot]=x;
120         Col[tot]=y;
121         S[y]++;
122         int colPos,rowPos;
123         colPos=y;
124         do colPos=D[colPos];
125         while(colPos!=y&&Row[colPos]<=x);
126         colPos=U[colPos];
127         if(Row[colPos]==x)return;
128         D[tot]=D[U[tot]=colPos];
129         U[D[tot]]=D[U[tot]]=tot;
130         rowPos=x+m;
131         do rowPos=R[rowPos];
132         while(rowPos!=x+m&&Col[rowPos]<=y);
133         rowPos=L[rowPos];
134         if(Col[rowPos]==y)return;
135         R[tot]=R[L[tot]=rowPos];
136         L[R[tot]]=R[L[tot]]=tot;
137     }
138     void cover(int col)
139     {
140         L[R[col]]=L[col];
141         R[L[col]]=R[col];
142         for(int i=D[col];i!=col;i=D[i])
143             for(int j=R[i];j!=i;j=R[j])
144                 if(Col[j]!=col)
145                 {
146                     U[D[j]]=U[j];
147                     D[U[j]]=D[j];
148                     S[Col[j]]--;
149                 }
150     }
151     void resume(int col)
152     {
153         for(int i=U[col];i!=col;i=U[i])
154             for(int j=L[i];j!=i;j=L[j])
```

```
155                 if(Col[j]!=col)
156                 {
157                     S[Col[j]]++;
158                     U[D[j]]=j;
159                     D[U[j]]=j;
160                 }
161         L[R[col]]=col;
162         R[L[col]]=col;
163     }
164     void initDFS()
165     {
166         for(int i=1;i<=n;i++)
167         {
168             L[R[i+m]]=L[i+m];
169             R[L[i+m]]=R[i+m];
170         }
171     }
172     void print(int deep)
173     {
174         printf("%d",deep);
175         for(int i=0;i<deep;i++)
176             printf("␣%d",O[i]);
177         putchar(10);
178     }
179     void DFS(int deep)
180     {
181         if(R[0]==0)
182         {
183             hasans=true;
184             print(deep);
185             return;
186         }
187         int tc=R[0];
188         for(int i=R[0];i!=0;i=R[i])
189             if(S[i]<S[tc])tc=i;
190         cover(tc);
191         for(int i=D[tc];i!=tc&&!hasans;i=D[i])
192         {
193             int temp=O[deep];
194             O[deep]=Row[i];
195             for(int j=R[i];j!=i;j=R[j])
196                 cover(Col[j]);
197             DFS(deep+1);
198             for(int j=L[i];j!=i;j=L[j])
199                 resume(Col[j]);
200             O[deep]=temp;
201         }
202         resume(tc);
203     }
204 }g;
```

5.2 DLX Repeat

```
1  //矩阵标号从开始1
2  //g.init(COL_SIZE);
3  //g.insert(x,y);
4  //g.DLX_DFS(0);
5
6  const int MAXN=100;
7  const int MAXM=100;
8  const int INF=0X3fffffff;
9  struct DLX_Repeat
10 {
11     int D[MAXN*MAXM],U[MAXN*MAXM],L[MAXN*MAXM],R[MAXN*MAXM],COL[MAXN*MAXM],ROW[MAXN*MAXM];
12     int CNT,BEG[MAXN],END[MAXN],ANS,USE[MAXN],_USE[MAXN];
13     int SUM[MAXM];
14     bool vis[MAXM];
15     void init(int n)
16     {
17         memset(BEG,0xff,sizeof(BEG));
18         for(int i=1;i<=n;i++)
19             SUM[L[i+1]=R[i-1]=D[i]=U[i]=i]=0;
20         L[L[1]=R[n]=0]=n;CNT=n+1;
21         ANS=n+1;
22     }
23     void insert(int r,int c)
24     {
25         D[CNT]=D[c];U[CNT]=c;U[D[c]]=CNT;D[c]=CNT;
26         COL[CNT]=c;ROW[CNT]=r;SUM[c]++;
27         if(BEG[r]==-1)BEG[r]=END[r]=CNT;
```

```
28          R[END[r]]=CNT;L[CNT]=END[r];R[CNT]=BEG[r];
29          L[BEG[r]]=CNT;END[r]=CNT++;
30      }
31      void DLX_Remove(int c)
32      {
33          for(int i=D[c];i!=c;i=D[i])
34              L[R[i]]=L[i],R[L[i]]=R[i],SUM[COL[i]]--;
35      }
36      void DLX_Resume(int c)
37      {
38          for(int i=U[c];i!=c;i=U[i])
39              L[R[i]]=R[L[i]]=i,SUM[COL[i]]++;
40      }
41      int Heuristics()
42      {
43          memset(vis,1,sizeof(vis));
44          int c,i,j,cnt=0;
45          for(c=R[0];c;c=R[c])
46              if(vis[c])
47                  for(cnt++,vis[c]=0,i=D[c];i!=c;i=D[i])
48                      for(j=R[i];j!=i;j=R[j])
49                          vis[COL[j]]=0;
50          return cnt;
51      }
52      void DLX_Dfs(int n)
53      {
54          if(Heuristics()+n>=ANS)return;
55          if(R[0]==0)
56          {
57              ANS=n;
58              for(int i=0;i<n;i++)
59                  USE[i]=_USE[i];
60              return;
61          }
62          int i,c=R[0];
63          for(i=R[0];i;i=R[i])
64              if(SUM[c]>SUM[i])c=i;
65          for(i=D[c];i!=c;i=D[i])
66          {
67              DLX_Remove(i);
68              for(int j=R[i];j!=i;j=R[j])
69                  DLX_Remove(j);
70              _USE[n]=ROW[i];
71              DLX_Dfs(n+1);
72              for(int j=L[i];j!=i;j=L[j])
73                  DLX_Resume(j);
74              DLX_Resume(i);
75          }
76      }
77  }g;
```

5.3 蔡勒公式

```
1   int get_week(int a,int m,int d,bool flag)
2   {
3       int w,c,y;
4       if(flag)
5       {
6           if(m<3) m+=12,a--;
7           c=a/100,y=a%100;
8           w=c/4-2*c+y+y/4+13*(m+1)/5+d-1;
9       }
10      else
11      {
12          if(m<3) m+=12,a--;
13          c=a/100,y=a%100;
14          w=y+y/4+c/4-2*c+13*(m+1)/5+d+2;
15      }
16      return (w%7+7)%7;
17  }
18  int main()
19  {
20      int a,m,d;
21      scanf("%d%d%d",&a,&m,&d);
22      bool flag=0;
23      if(1582<a) flag=1;
24      else
25      {
26          if(10<m) flag=1;
27          else
```

```
28          {
29              if(4<d) flag=1;
30              else flag=0;
31          }
32      }
33      printf("%d\n",get_week(a,m,d,flag));
34      return 0;
35  }
```

5.4 最优双调路线

```
1   //求解过程：
2   //( 1 )首先将各点按照坐标从小到大排列，时间复杂度为x O(nlgn)
3   //( 2 )寻找子结构：定义从 Pi 到 Pj 的路径为：从 Pi 开始，从右到左一直到 P1 ，然后从左到右一直到 Pj 。在这个路径上，会经过 P1 到 Pmax(i,j) 之间的所有点且只经过一次
4   //在定义 d(i,j) 为满足这一条件的最短路径。我们只考虑 i>=j 的情况
5   //同时，定义 dist(i,j) 为点 Pi 到 Pj 之间的直线距离
6   //( 3 )最优解：我们需要求的是 d(n,n)
7   //关于子问题 d(i,j) 的求解，分三种情况：
8   //A 、当 j < i − 1 时， d(i,j) = d(i−1,j) + dist(i − 1,i)
9   //由定义可知，点 Pi−1 一定在路径 Pi−Pj 上，而且又由于 j<i−1 因此， Pi 的左边的相邻点一定是 Pi−1 因此可以得出上述等式.
10  //B 、当 j = i − 1 时，与 Pi 左相邻的那个点可能是 P1 到 Pi−1 总的任何一个。因此需要递归求出最小的那个路径：
11  //d(i,j) = d(i,i−1) = min{d(i−1,k) + dist(k,i)}, 其中 1 <= k <= j
12  //C 、当 j=i 时，路径上最后相连的两个点可能是 P1−、PiP2−Pi...Pi−1−Pi
13  //因此有：
14  //d(i,i) = min{d(i,1)+dist(1,i),...,d(i,i−1),dist(i−1,i)}.
```

5.5 插头 dp__ 括号匹配

```
1   #include<cstdio>
2   #include<cstring>
3   #include<cmath>
4   #include<algorithm>
5   #include<vector>
6   #include<set>
7   using namespace std;
8
9   char maz[14][14];
10  const int sd=30007;
11  struct Hash_map
12  {
13      int head[sd],st[sd],next[sd],sz;
14      long long val[sd];
15      void init()
16      {
17          memset(head,0xff,sizeof(head));sz=0;
18      }
19      int insert(long long t,int s)
20      {
21          int f=s%sd;
22          for(int i=head[f];~i;i=next[i])
23              if(st[i]==s)
24                  return val[i]+=t;
25          st[sz]=s;val[sz]=t;
26          next[sz]=head[f];head[f]=sz++;
27          return t;
28      }
29  }dp[2];
30  int cg(int s,int l,int f)
31  {
32      int z=(f&1)?2:-2,cnt=0;
33      s^=(f|f<<2)<<l;
34      while(cnt>=0)
35      {
36          l+=z;
37          if((s>>l&3^f)==3)cnt--;
38          if((s>>l&3^f)==0)cnt++;
39      }
40      return s^3<<l;
41  }
42  int main()
43  {
44      freopen("1.txt","r",stdin);
45      int n,m,p1=0,p2=0;
46      scanf("%d%d",&n,&m);
47      for(int i=0;i<n;i++)
48      {
```

```
        scanf("%s",maz[i]);
        for(int j=0;j<m;j++)
            if(maz[i][j]=='.')p1=i,p2=j;
    }
    maz[p1][p2]='?';
    int f=0,lim=1<<(m<<1);
    dp[0].init();
    dp[0].insert(1,0);
    long long ans=0;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            dp[f^=1].init();
            int s,h,l=j<<1;
            int x=3<<l,y=12<<l;
            long long v;
            for(int z=0;z<dp[f^1].sz;z++)
            {
                s=dp[f^1].st[z];v=dp[f^1].val[z];
                h=(x&s)^((y&s)>>2);h|=h<<2;
                if(maz[i][j]=='*')!(x&s|y&s)?dp[f].insert(v,s):0;
                else if(!(x&s|y&s))dp[f].insert(v,s&~(9<<l)|(9<<l));
                else if(!(h>>l&3))dp[f].insert(v,cg(s,l,(x&s)>>l));
                else if((h>>l&3)<3)dp[f].insert(v,s^h),dp[f].insert(v,s);
                else if(((x&s)>>l)==2)dp[f].insert(v,s&~(15<<l));
                else if((s&~(15<<l))==0&&maz[i][j]=='?')ans+=v;
            }
        }
        dp[f^1].init();
        for(int j=0;j<dp[f].sz;j++)
            if(dp[f].st[j]<lim)dp[f^1].insert(dp[f].val[j],dp[f].st[j]<<2);
        f^=1;
    }
    printf("%I64d\n",ans);
    return 0;
}
```

5.6 斯坦纳树

```
//HDU 4085
#include<cstdio>
#include<cstring>
#include<vector>
#include<queue>
#include<algorithm>
#define N 60
#define INF 2000000
using namespace std;
struct edge{
    int v,w;
    edge *nxt;
}E[2009],*Adj[N],*cur;
int n,m,K,nn;
int s[N],in[N][1<<10];
int d[N][1<<10],dp[1<<10];
queue<int> Q;
void addedge(int u,int v,int w){cur->v=v,cur->w=w,cur->nxt=Adj[u],Adj[u]=cur++;}
bool check(int x){
    int r=0;
    for(int i=0;x;i++,x>>=1)
        r+=(x&1)*(i<K?1:-1);
    return r==0;
}
inline bool update(int x,int y,int w){
    if(w<d[x][y]) return d[x][y]=w,true;
    return false;
}
void spfa(){
    while(!Q.empty()){
        int x=Q.front()/10000,y=Q.front()%10000;
        in[x][y]=0;
        Q.pop();
        for(edge *i=Adj[x];i;i=i->nxt)
            if(update(i->v,y|s[i->v],d[x][y]+i->w)&&y==(y|s[i->v])&&!in[i->v][y])
                in[i->v][y]=1,Q.push(i->v*10000+y);

    }
}
void init(){
```

```
    cur=E;
    memset(Adj,0,sizeof(Adj));
    memset(s,0,sizeof(s));
    scanf("%d%d%d",&n,&m,&K);
    nn=1<<(2*K);
    for(int i=1;i<=n;i++)
        for(int j=0;j<nn;j++)
            d[i][j]=INF;
    while(m--){
        int u,v,w;
        scanf("%d%d%d",&u,&v,&w);
        addedge(u,v,w);
        addedge(v,u,w);
    }
    for(int i=1;i<=K;i++){
        s[i]=1<<(i-1),d[i][s[i]]=0;
        s[n-i+1]=1<<(K+i-1),d[n-i+1][s[n-i+1]]=0;
    }
}
int main(){
    int T;
    scanf("%d",&T);
    while(T--){
        init();
        for(int y=0;y<nn;y++){
            for(int x=1;x<=n;x++){
                for(int i=(y-1)&y;i;i=(i-1)&y)
                    d[x][y]=min(d[x][y],d[x][i|s[x]]+d[x][(y-i)|s[x]]);
                if(d[x][y]<INF) Q.push(x*10000+y),in[x][y]=1;
            }
            spfa();
        }
        for(int j=0;j<nn;j++){
            dp[j]=INF;
            for(int i=1;i<=n;i++) dp[j]=min(dp[j],d[i][j]);
        }
        for(int i=1;i<nn;i++)
            if(check(i))
                for(int j=i&(i-1);j;j=(j-1)&i)
                    if(check(j))
                        dp[i]=min(dp[i],dp[j]+dp[i-j]);
        if(dp[nn-1]>=INF) puts("No solution");
        else printf("%d\n",dp[nn-1]);
    }
}
```

5.7 位运算

```
//位反转
inline int reverse_bits(int x)
{
    x = ((x >> 1) & 0x55555555) | ((x << 1) & 0xaaaaaaaa);
    x = ((x >> 2) & 0x33333333) | ((x << 2) & 0xcccccccc);
    x = ((x >> 4) & 0x0f0f0f0f) | ((x << 4) & 0xf0f0f0f0);
    x = ((x >> 8) & 0x00ff00ff) | ((x << 8) & 0xff00ff00);
    x = ((x >>16) & 0x0000ffff) | ((x <<16) & 0xffff0000);
    return x;
}

inline LL reverse_bits(LL x)
{
    x = ((x >> 1) & 0x5555555555555555LL) | ((x << 1) & 0xaaaaaaaaaaaaaaaaLL);
    x = ((x >> 2) & 0x3333333333333333LL) | ((x << 2) & 0xccccccccccccccccLL);
    x = ((x >> 4) & 0x0f0f0f0f0f0f0f0fLL) | ((x << 4) & 0xf0f0f0f0f0f0f0f0LL);
    x = ((x >> 8) & 0x00ff00ff00ff00ffLL) | ((x << 8) & 0xff00ff00ff00ff00LL);
    x = ((x >>16) & 0x0000ffff0000ffffLL) | ((x <<16) & 0xffff0000ffff0000LL);
    x = ((x >>32) & 0x00000000ffffffffLL) | ((x <<32) & 0xffffffff00000000LL);
    return x;
}

//LCIS  = =

memset(dp,0,sizeof(dp));
for(int i=1;i<=n;i++)
    for(int j=1,k=0;j<=m;j++)
    {
        if(a[i]==b[j])dp[j]=dp[k]+1;
        else if(a[i]>b[j]&&dp[j]>dp[k])k=j;
    }
int ans=0;
```

```
for(int i=1;i<=m;i++)
    ans=max(ans,dp[i]);


//回文序列
    memset(dp, 0, sizeof(dp));
    for(int i=1;i<=n;i++) dp[i][i] = 1;
    for(int l=1;l<=n;l++)
        for(int i=1;i+l-1<=n;i++)
        {
            int j=i+l-1;
            dp[i][j]=max(dp[i+1][j-1]+(a[i]==a[j]?2:0),max(dp[i+1][j],dp[i][j-1]));
        }


//求和dp
    a=1<<24;
    for(int i=0;i<24;i++)
        for(int j=0;j<a;j++)
            if((1<<i)&j)dp[j]+=dp[(1<<i)^j];

//枚举子集

for(int i=(s-1)&s;i;i=(i-1)&s)  //真子集

//枚举长为含个的串nk101
int n=5,k=3;
for(int s=(1<<k)-1,u=1<<n;s<u;)
{
  //do it;
  int b=s&-s;
  s=(s+b)|(((s^(s+b))>>2)/b);
}
```

5.8 JAVA 读入

```
import java.util.*;
import java.math.*;
import java.io.InputStreamReader;
import java.io.IOException;
import java.io.BufferedReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.util.StringTokenizer;
import java.io.InputStream;
public class Main
{
    public static BigInteger gcd(BigInteger a, BigInteger b)
    {
        if(b.compareTo(BigInteger.ZERO) == 0)
            return a;
        return gcd(b, a.mod(b));
    }
    public static void main(String args[])
    {
        InputStream inputStream = System.in;
        OutputStream outputStream = System.out;
        InputReader in = new InputReader(inputStream);
        PrintWriter out = new PrintWriter(outputStream);
        long T=in.nextInt();
        int cas = 0;
        while(T-->0)
        {
            long n = in.nextInt();
            long ans = -7 * n + 1;
            BigInteger nn = BigInteger.valueOf(n);
            nn = nn.multiply(nn);
            cas++;
            out.print("Case␣#");
            out.print(cas);
            out.print(":␣");
            out.println(nn.multiply(BigInteger.valueOf(8)).add(BigInteger.valueOf(ans)));
        }
        out.close();
    }
}
class InputReader {
    public BufferedReader reader;
    public StringTokenizer tokenizer;
```

```

    public InputReader(InputStream stream) {
        reader = new BufferedReader(new InputStreamReader(stream), 32768);
        tokenizer = null;
    }

    public String next() {
        while (tokenizer == null || !tokenizer.hasMoreTokens()) {
            try {
                tokenizer = new StringTokenizer(reader.readLine());
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
        return tokenizer.nextToken();
    }

    public long nextLong() {
        return Long.parseLong(next());
    }
    public int nextInt() {
        return Integer.parseInt(next());
    }
}
```

5.9 printf&scanf

```


double c;int b;
scanf("%lf%d",&c,&b);
printf("%.*f\n",b,c);
```

5.10 STL

```
#include<bits/stdc++.h>


#include<tr1/unordered_map>
using namespace tr1;
unordered_map<int,int>mp;


partial_sort(a,a+m,a+n,cmp);

nth_element(a,a+m,a+n,cmp);
//第大元素m从开始数(0)，然后访问a[m]


bitset支持位运算()
b.any()  //中是否存在置为的二进制位b1
b.none()  //中不存在置为的二进制位吗？b1
b.count()  //中置为的二进制位的个数b1
b.size()  //中二进制位的个数b
b.set()  //把中所有二进制位都置为b1
b.set(pos)  //把中在处的二进制位置为bpos1
b.reset()  //把中所有二进制位都置为b0
b.reset(pos)  //把中在处的二进制位置为bpos0
b.flip()  //把中所有二进制位逐位取反b
b.flip(pos)  //把中在处的二进制位取反bpos
b.to_ulong()  //用中同样的二进制位返回一个bunsigned 值long
os << b  //把中的位集输出到流bos
```

5.11 读入外挂

```
C++:

//注意有负数要改
//可以改成引用试试
inline int Int()
{
  int a=0;char c=getchar();
  while(c<'0')c=getchar();
  while(c>='0'&&c<='9')a=(a<<3)+(a<<1)+c-'0',c=getchar();
  return a;
```

```
}

//另一种 可处理负数
inline int Int()
{
  int a=0;char c=getc(stdin),s;
  while(c<'0')s=c,c=getc(stdin);
  while(c>='0'&&c<='9')a=(a<<3)+(a<<1)+c-'0',c=getc(stdin);
  return s!='-'?a:-a;
}

//输出外挂
inline void out(int c)
{
    if(c>9)out(c/10);
    putchar(c-c/10*10+'0');
}
```

5.12 栈外挂

```
//原理不明

#pragma comment(linker,"/STACK:102400000,102400000")

int size = 256 << 20; // 256MB
char *p = (char*)malloc(size) + size;
__asm__("movl␣%0,␣%%esp\n" :: "r"(p));
```