题目回顾

```
void vis(int flag[], int n, int yuan[]){
    if(n==3){
        int xtop=0;
        int xian[5];
        for(int i=0;i<5;i++){
            if(flag[i]==0){
                xian[xtop++]=i;
            }
        }
        suan(yuan, xian);
        return ;
    }
    for(int i=0;i<5;i++){
        if(flag[i]==0){
            flag[i]=1;
            yuan[n]=i;
            vis(flag, n+1, yuan);
            flag[i]=0;
        }
    }
}
```

小 A 想给小 B 寄几盒蒜。

$\mathbb{R}^2$ 平面上有五个特殊点 A, B, C, D, E，任意两点不重合，任意三点不共线。

C_5^3 十种情况

小 B 可以任意选择 $\{A, B, C, D, E\}$ 中不同的三个点 P_1, P_2, P_3，并记 $\triangle P_1P_2P_3$ 的外接圆为 C。小 B 只能在 C 上移动。

记以 $\{A, B, C, D, E\}\backslash\{P_1, P_2, P_3\}$ 中剩余的两个点 P_4, P_5 所在的直线为 ℓ。小 A 只能在 ℓ 上移动。

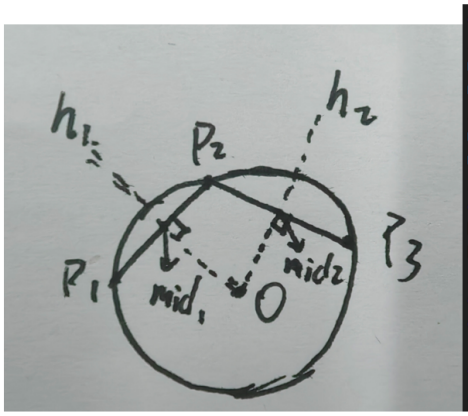
可是小 B 不喜欢吃蒜。

小 B 想最大化两人之间的最短距离。形式化地说，你需要求出 $\max\limits_{C,\ell}\min\limits_{P\in\ell,Q\in C}|PQ|$，其中 $|PQ|=\sqrt{(x_P-x_Q)^2+(y_P-y_Q)^2}$ 为线段 PQ 的欧几里得距离。

与上一个题一模一样

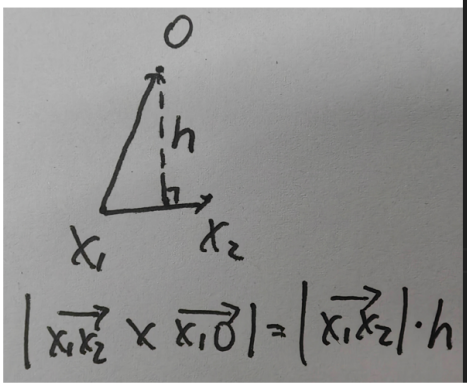
. . .

已知圆上三点求圆心及半径




```
double midx1, midy1, midx2, midy2;
//定义两个点，分别表示两个中点
midx1=(a[yuan[0]].x+a[yuan[1]].x)/2;
midy1=(a[yuan[0]].y+a[yuan[1]].y)/2;
//求出点2和点1的中点
midx2=(a[yuan[2]].x+a[yuan[1]].x)/2;
midy2=(a[yuan[2]].y+a[yuan[1]].y)/2;
//求出点2和点3的中点
double h1, h2;
h1=-(a[yuan[1]].x-a[yuan[0]].x)/(a[yuan[1]].y-a[yuan[0]].y);
h2=-(a[yuan[2]].x-a[yuan[1]].x)/(a[yuan[2]].y-a[yuan[1]].y);
//求出分别与直线p1p2，p2p3垂直的直线的斜率
double x = (midy2-midy1+midx1*h1-midx2*h2)/(h1-h2);
double y = h1*(x-midx1)+midy1;
//连立两条中垂线方程求解交点得到圆心坐标
double r = sqrt((a[yuan[0]].x-x)*(a[yuan[0]].x-x)+(a[yuan[0]].y-y)*(a[yuan[0]].y-y));
//用圆心和其中一个点求距离得到半径
```

圆到直线距离




```
double l = sqrt((a[xian[0]].x-a[xian[1]].x)*(a[xian[0]].x-a[xian
//求|x1x2|
point tmp;
tmp.x=x;
tmp.y=y;
double result = cha(a[xian[0]], a[xian[1]], a[xian[0]], tmp);
if(result<0){
    result=-result;
}
//求|x1x2 x x1o|
result=result/l-r;
//圆到直线距离
if(result>max){
    max=result;
}
return ;
```

详细代码：

```
#include <iostream>
#include <math.h>
#define eps 1e-6

struct point {
    double x;
    double y;
};
struct point a[10];
double max=0.000;

double cha(point start1, point end1, point start2, point end2){
    return (end1.x-start1.x)*(end2.y-start2.y)-(end1.y-start1.y)*(end2.x-
start2.x);
}

void suan(int yuan[], int xian[]){
    double midx1, midy1, midx2, midy2;
    //定义两个点，分别表示两个中点
    midx1=(a[yuan[0]].x+a[yuan[1]].x)/2;
    midy1=(a[yuan[0]].y+a[yuan[1]].y)/2;
    //求出点2和点1的中点
    midx2=(a[yuan[2]].x+a[yuan[1]].x)/2;
    midy2=(a[yuan[2]].y+a[yuan[1]].y)/2;
    //求出点2和点3的中点
    double h1, h2;
    h1=-(a[yuan[1]].x-a[yuan[0]].x)/(a[yuan[1]].y-a[yuan[0]].y);
    h2=-(a[yuan[2]].x-a[yuan[1]].x)/(a[yuan[2]].y-a[yuan[1]].y);
    //求出分别与直线p1p2，p2p3垂直的直线的斜率
```

```
    double x = (midy2-midy1+midx1*h1-midx2*h2)/(h1-h2);
    double y = h1*(x-midx1)+midy1;
    //连立两条中垂线方程求解交点得到圆心坐标
    double r = sqrt((a[yuan[0]].x-x)*(a[yuan[0]].x-x)+(a[yuan[0]].y-y)*
(a[yuan[0]].y-y));
    //用圆心和其中一个点求距离得到半径

    double l = sqrt((a[xian[0]].x-a[xian[1]].x)*(a[xian[0]].x-a[xian[1]].x)+
(a[xian[0]].y-a[xian[1]].y)*(a[xian[0]].y-a[xian[1]].y));
    //求|x1x2|
    point tmp;
    tmp.x=x;
    tmp.y=y;
    double result = cha(a[xian[0]], a[xian[1]], a[xian[0]], tmp);
    if(result<0){
        result=-result;
    }
    //求|x1x2 x x1o|
    result=result/l-r;
    //圆到直线距离
    if(result>max){
        max=result;
    }
    return ;
}

void vis(int flag[], int n, int yuan[]){
    if(n==3){
        int xtop=0;
        int xian[5];
        for(int i=0;i<5;i++){
            if(flag[i]==0){
                xian[xtop++]=i;
            }
        }
        suan(yuan, xian);
        return ;
    }
    for(int i=0;i<5;i++){
        if(flag[i]==0){
            flag[i]=1;
            yuan[n]=i;
            vis(flag, n+1, yuan);
            flag[i]=0;
        }
    }
}

int flag[10];
int yuan[10];

int main(){
    int T;
    scanf("%d", &T);
```

```
    while(T--){
        max=0.000;
        for(int i=0;i<5;i++){
            scanf("%lf%lf", &a[i].x, &a[i].y);
        }
        vis(flag, 0, yuan);
        if(max<eps){
            max=0.000;
        }
        printf("%.3lf\n", max);
    }

    return 0;
}
```