算法分析与设计E5-B

22373340 詹佳博

题目描述

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。

小 A 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 A 只能在 C 上移动。

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

小 A 想最小化两人之间的最短距离，使得寄蒜所支付的邮费最少。形式化地说，你需要求出 $\min\limits_{C,\ell}\min\limits_{P\in C,Q\in\ell}|PQ|$，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。

输入格式

本题测试点包含多组数据。

第一行，一个正整数 T（$1\le T\le 10^4$），表示数据组数。

对于每组数据：

一行，10 个整数 $x_A,y_A,x_B,y_B,x_C,y_C,x_D,y_D,x_E,y_E$（$-10^3\le x,y\le 10^3$），表示点 A,B,C,D,E 的坐标。

输出格式

对于每组数据：

一行，一个实数，表示 $\min\limits_{C,\ell}\min\limits_{P\in C,Q\in\ell}|PQ|$。答案保留三位小数。

题目分析

题目也就是画好图，明确两个目标：三角形外接圆求法+垂线段最短的垂足求法。

对于三角形外接圆，可以通过暴力列出点到三个顶点距离相等的等式进行求解，通过参数方程轮换式可解出圆心，再通过两点间距离求半径。

对于垂线段求法，先通过投影算出垂足距离固定直线的两点的相对位置，再通过向量法则求出垂足位置即可。

题目求解

- 先准备好初始条件。

```
#include <bits/stdc++.h>
#define LL long long
const int inf_int = 0x3f3f3f3f;
const LL inf_ll = 0x3f3f3f3f3f3f3f3f;
```

```cpp
using namespace std;

struct Point {
    double x, y;    //坐标
};

struct LineSegment {
    Point p1, p2;   //两点确定一条直线
};

struct Circle {
    double r;       //半径长度
    Point centre;   //圆心
};
```

- 将所需要的函数一一实现出来。

```cpp
double direction(Point pi, Point pj, Point pk) {    //pipk,pipj向量叉积
    return (pk.x - pi.x) * (pj.y - pi.y) - (pj.x - pi.x) * (pk.y - pi.y);
}

double PointMul(Point pi, Point pj, Point pk) {     //pipk,pipj向量点积
    return (pk.x - pi.x) * (pj.x - pi.x) + (pk.y - pi.y) * (pj.y - pi.y);
}

double distance(Point a, Point b) {                 //两点之间的距离
    return sqrt(pow(a.x - b.x, 2) + pow(a.y - b.y, 2));
}

Circle TriCircle(Point a, Point b, Point c) {
    //暴力推导,设圆心x
    Point x;
    double r;
    double fenmu=2*direction(a,b,c);
    x.x=((b.y-a.y)*(c.y*c.y-a.y*a.y+c.x*c.x-a.x*a.x)-(c.y-a.y)*(b.y*b.y-
a.y*a.y+b.x*b.x-a.x*a.x))/fenmu;
    x.y=-((b.x-a.x)*(c.x*c.x-a.x*a.x+c.y*c.y-a.y*a.y)-(c.x-a.x)*(b.x*b.x-
a.x*a.x+b.y*b.y-a.y*a.y))/fenmu;
    r=distance(x,a);
    return Circle{r,x};
}

double PointToLine(Point p, LineSegment l) {
    double len = distance(l.p1, l.p2);//线段距离

    if (len == 0) { // 线段长度为0
        return distance(p, l.p1);
    }

    double r = PointMul(l.p1, l.p2, p) / pow(len, 2);
    Point foot = { l.p1.x + r * (l.p2.x - l.p1.x), l.p1.y + r * (l.p2.y -
l.p1.y) };
    return distance(p, foot);
}
```

- 通过循环找所有排列组合，并计算最小值。

```
Point p[5];
int main() {
    int t;
    scanf("%d", &t);
    while (t--) {
        for (int i = 0; i < 5; i++) {
            scanf("%lf%lf", &p[i].x, &p[i].y);
        }
        double mini=100000000;
        for (int i = 0; i < 3; i++) {
            for (int j = i + 1; j < 4; j++) {
                for (int k = j + 1; k < 5; k++) {
                    double ans;
                    Point L[2];
                    int top = 0;
                    for (int l = 0; l < 5; l++) {
                        if (l != i && l != j && l != k) {
                            L[top++] = p[l];
                        }
                    }
                    LineSegment line=LineSegment{L[0],L[1]};
                    Circle cc=TriCircle(p[i],p[j],p[k]);
                    Point center=cc.centre;

                    double len=PointToLine(center,line);
                    if(len<cc.r){
                        ans=0;
                    }else{
                        ans=len-cc.r;
                    }
                    if(ans<mini){
                        mini=ans;
                    }
                }
            }
        }
        printf("%.3f\n",mini);
    }
}
```

时间复杂度

经分析，仅有常数级时间复杂度，故$O(1)$。

总代码

```
#include <bits/stdc++.h>
#define LL long long
const int inf_int = 0x3f3f3f3f;
const LL inf_ll = 0x3f3f3f3f3f3f3f3f;

using namespace std;
```

```
struct Point {
    double x, y;    //坐标
};

struct LineSegment {
    Point p1, p2;   //两点确定一条直线
};

struct Circle {
    double r;       //半径长度
    Point centre;   //圆心
};

double direction(Point pi, Point pj, Point pk) {    //pipk,pipj向量叉积
    return (pk.x - pi.x) * (pj.y - pi.y) - (pj.x - pi.x) * (pk.y - pi.y);
}

double PointMul(Point pi, Point pj, Point pk) {     //pipk,pipj向量点积
    return (pk.x - pi.x) * (pj.x - pi.x) + (pk.y - pi.y) * (pj.y - pi.y);
}

double distance(Point a, Point b) {                 //两点之间的距离
    return sqrt(pow(a.x - b.x, 2) + pow(a.y - b.y, 2));
}

Circle TriCircle(Point a, Point b, Point c) {
    //暴力推导,设圆心x
    Point x;
    double r;
    double fenmu=2*direction(a,b,c);
    x.x=((b.y-a.y)*(c.y*c.y-a.y*a.y+c.x*c.x-a.x*a.x)-(c.y-a.y)*(b.y*b.y-
a.y*a.y+b.x*b.x-a.x*a.x))/fenmu;
    x.y=-((b.x-a.x)*(c.x*c.x-a.x*a.x+c.y*c.y-a.y*a.y)-(c.x-a.x)*(b.x*b.x-
a.x*a.x+b.y*b.y-a.y*a.y))/fenmu;
    r=distance(x,a);
    return Circle{r,x};
}

double PointToLine(Point p, LineSegment l) {
    double len = distance(l.p1, l.p2);//线段距离

    if (len == 0) { // 线段长度为0
        return distance(p, l.p1);
    }

    double r = PointMul(l.p1, l.p2, p) / pow(len, 2);
    Point foot = { l.p1.x + r * (l.p2.x - l.p1.x), l.p1.y + r * (l.p2.y - l.p1.y)
};
    return distance(p, foot);
}

Point p[5];
int main() {
    int t;
    scanf("%d", &t);
    while (t--) {
        for (int i = 0; i < 5; i++) {
            scanf("%lf%lf", &p[i].x, &p[i].y);
        }
        double mini=100000000;
        for (int i = 0; i < 3; i++) {
            for (int j = i + 1; j < 4; j++) {
                for (int k = j + 1; k < 5; k++) {
                    double ans;
                    Point L[2];
                    int top = 0;
                    for (int l = 0; l < 5; l++) {
                        if (l != i && l != j && l != k) {
                            L[top++] = p[l];
                        }
                    }
                    LineSegment line=LineSegment{L[0],L[1]};
                    Circle cc=TriCircle(p[i],p[j],p[k]);
                    Point center=cc.centre;

                    double len=PointToLine(center,line);
                    if(len<cc.r){
                        ans=0;
                    }else{
                        ans=len-cc.r;
                    }
                    if(ans<mini){
                        mini=ans;
                    }
                }
            }
        }
        printf("%.3f\n",mini);
    }
}
```