> D-计算几何3

题目描述

- 给定 5 个点，其中任意两点不重合，任意三点不共线。
- 记其中三个点的外接圆为 $\odot P$，另外两点形成的直线为 l，求直线和圆的最小距离。

解题思路

- 由于点的总数只有 5，因此我们可以枚举哪三个点组成 $\odot P$，剩下两个点即形成 l。
- 题目难点在于，给定坐标的三点，如何求其外接圆。
- 由公式知，设圆心的坐标为 (x,y)，三点坐标分别为 $(x_1,y_1),(x_2,y_2),(x_3,y_3)$ 则有：
  - 令
  - $a=x_1-x_2$
  - $b=y_1-y_2$
  - $c=x_1-x_3$
  - $d=y_1-y_3$
  - $e=\frac{(x_1^2-x_2^2)-(y_2^2-y_1^2)}{2}$
  - $f=\frac{(x_1^2-x_3^2)-(y_3^2-y_1^2)}{2}$
  - 则 $x=\frac{e\times d-b\times f}{a\times d-b\times c},y=\frac{a\times f-e\times c}{a\times d-b\times c}$
- 求出圆心后，半径可以用圆心到某一点的距离来求：
  - $r=\sqrt{(x-x_1)^2+(y-y_1)^2}$
- 求出圆心和半径后，圆到直线的距离则变为判断圆心到直线的距离
  - 设直线为 $Ax+By+C=0$
  - $dis_{Pl}=\frac{|Ax+By+C|}{\sqrt{A^2+B^2}}$
  - 接着分类讨论：
    - 当 $dis>r$ 时，最小距离即为 $dis-r$
    - 当 $dis\le r$ 时，最小距离为 0

代码实现

```cpp
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <algorithm>
#include <cmath>

const double eps = 1e-16;

int read(void) {
    int x, f = 1;
    char ch;
    while(!isdigit(ch = getchar()))
        (ch == '-') && (f = -1);
```

```
    for(x = ch ^ 48; isdigit(ch = getchar()); x = (x << 1) + (x << 3) + (ch ^
48));
    return x * f;
}

void write(int x)
{
    if(x < 0)
        putchar('-'), x = -x;
    if(x > 9)
        write(x / 10);
    putchar(x % 10 + '0');
    return;
}

struct Node {
    double x, y;
};

Node nodes[6];

double distBetweenTwoPoints(Node A, Node B) {
    return std :: sqrt((A.x - B.x) * (A.x - B.x) + (A.y - B.y) * (A.y - B.y));
}

double distBetweenCircleandLine(Node A, Node B, Node C, Node D, Node E) {
    // Node A, B, C for Circle, Node D, E for line
    double a = (A.x - B.x);
    double b = (A.y - B.y);
    double c = (A.x - C.x);
    double d = (A.y - C.y);
    double e = ((A.x * A.x - B.x * B.x) - (B.y * B.y - A.y * A.y)) / 2.0;
    double f = ((A.x * A.x - C.x * C.x) - (C.y * C.y - A.y * A.y)) / 2.0;
    double xx = (e * d - b * f) / (a * d - b * c);
    double yy = (a * f - e * c) / (a * d - b * c);
    Node P = Node{xx, yy};
    double r = distBetweenTwoPoints(A, P);
    double dist = 0.0;
    if(D.x == E.x)
        dist = std :: abs(xx - D.x);
    else {
        double a = (D.y - E.y) / (D.x - E.x);
        double b = D.y - a * D.x;
        dist = std :: abs(a * xx + b - yy) / std :: sqrt(a * a + 1);
    }
    if(dist - r < eps)
        return 0;
    else
        return dist - r;
}
int main() {
    int T = read();
    while(T--) {
        //printf("%.3lf\n", (double) 0);
        //continue;
        for(int i = 1; i <= 5; i++)
```

```
            nodes[i].x = read(), nodes[i].y = read();
        double ans = 1 << 30;
        for(int i = 1; i <= 5; ++i) {
            for(int j = i + 1; j <= 5; ++j) {
                for(int k = j + 1; k <= 5; ++k) {
                    int d = 0, e;
                    for(int l = 1; l <= 5; ++l) {
                        if(l == i || l == j || l == k)
                            continue;
                        else {
                            if(d == 0)
                                d = l;
                            else {
                                e = l;
                                break;
                            }
                        }
                    }
                    ans = std :: min(ans, distBetweenCircleandLine(nodes[i],
nodes[j], nodes[k], nodes[d], nodes[e]));
                }
            }
        }
        printf("%.3lf\n", ans);
    }
}
```