**Optimum Location**

**Difficulty: HARD**

A picture containing text, clipart

Description automatically generated

**Avg. time to solve**

**45 min**

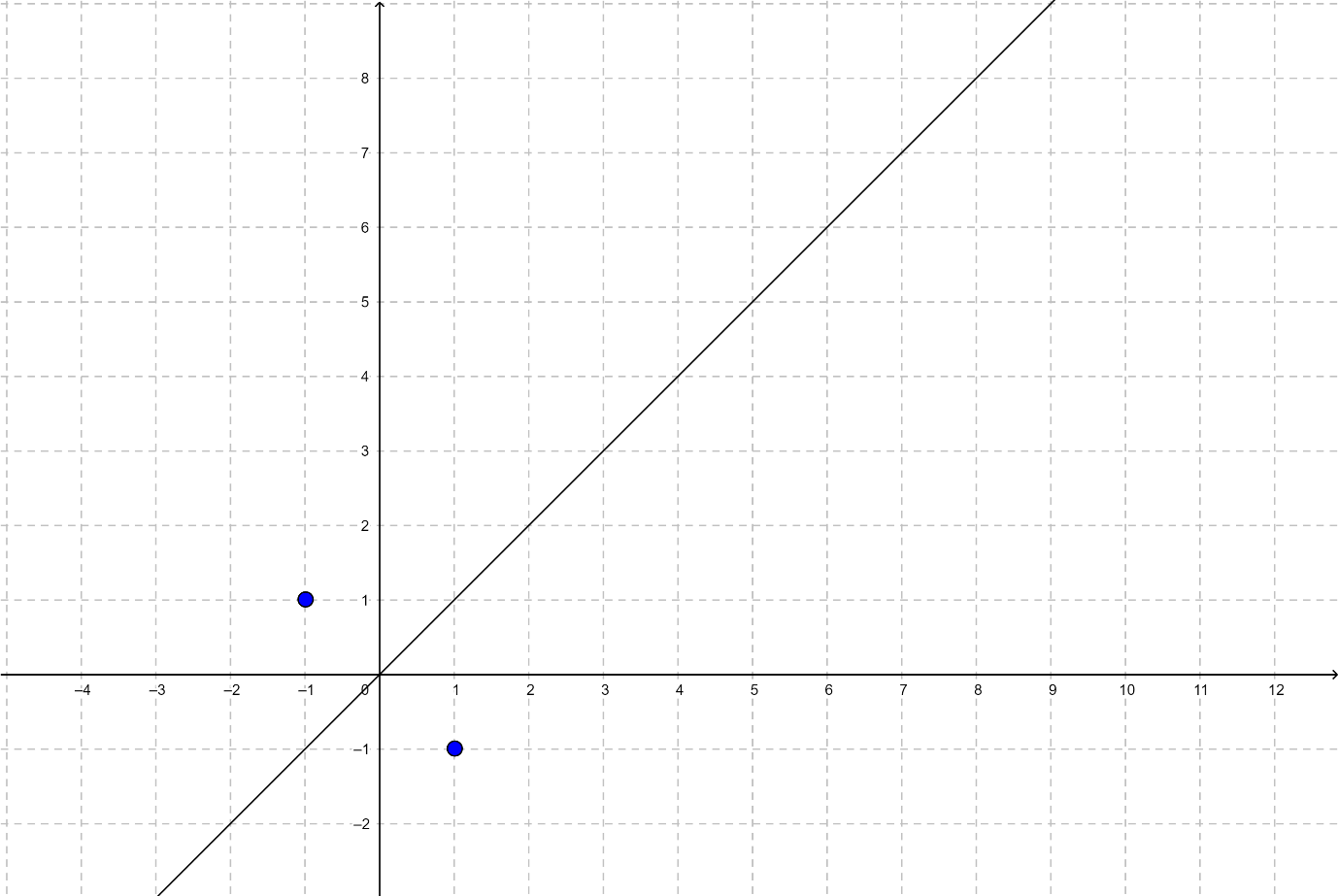
**Success Rate**

**55%**

**Problem Statement**

**You are given a straight line on a 2D plane in the form of (ax + by + c = 0) and an array of points of the form (Xi, Yi). Your task is to find a point on the given line for which the sum of distances from this point to the given array/list of points is minimum.**

**For Example :**



Here the equation for the given line is x - y = 0, i.e. a = 1, b = -1 and c = 0.

There are two points, i.e. (-1, 1) and (1, -1).

So, the point on the line that has minimum distances from the points is (0, 0). The sum of distances is sqrt(1 + 1) + sqrt(1 + 1) = sqrt(2) + sqrt(2) = 2.83 (rounding to 2 decimal digits).

**Input Format :**

The first line contains an integer 'T', which denotes the number of test cases or queries to be run. Then, the T test cases follow.

The first line of each test case contains three space-separated integers denoting a, b and c, respectively, representing the equation of the given line as mentioned above.

The second line of each test case contains a single integer N, denoting the number of coordinates.

Then N lines follow. Each line contains two space-separated integers denoting the x-coordinate and the y-coordinate of the point, respectively.

**Output Format :**

For each test case, print the minimum possible distance.

Your answer will be considered correct if its absolute or relative error doesn’t exceed 10^-6.

Output for each test case will be printed in a separate line.

**Note:**

You do not need to print anything. It has already been taken care of. Just implement the given function.

**Constraints :**

1 <= T <= 100

1 <= N <= 3000

-100 <= a, b, c, points[i].x, points[i].y <= 100 and (a != 0, b != 0)

Time limit: 1 second

**Sample Input 1:**

2

3 2 5

5

1 -1

2 3

4 4

5 -1

3 2

3 1 -4

4

1 2

4 -2

5 -3

7 -6

**Sample Output 1:**

24.94

15.32

**Explanation For Sample 1:**

For the first test case, the optimum point is (-0.72, -1.42).

For the second test case, the optimum point is (2.29, -2.87).

**Sample Input 2:**

2

-1 2 -4

4

-2 1

2 3

0 2

-4 0

2 -5 4

5

2 1

7 -9

4 -3

-5 2

8 -6

**Sample Output 2:**

8.94

33.83

class Point {

public:

    double x, y;

    Point(double a=0, double b=0) : x(a),y(b) {}

};

double distance(Point P1, Point P2) {

    // Calculating distance

    double x1=P1.x, y1=P1.y;

    double x2=P2.x, y2=P2.y;

    return sqrt(pow(x2 - x1, 2) +

                pow(y2 - y1, 2) \* 1.0);

}

Point mirrorImage(int a, int b, int c, Point X) {

    double temp = -2 \* (a \* X.x + b \* X.y + c) /

                              (a \* a + b \* b);

    double p = temp \* a + X.x;

    double q = temp \* b + X.y;

    Point P(p, q);

    return P;

}

double f(vector<Point> V, Point P) {

    double sum=0;

    for (int i=0; i<V.size(); i++) {

        sum+=distance(V[i], P);

    }

    return sum;

}

double optimumDistance(int a, int b, int c, vector<vector<int>> &points, int n){

    // Write your code here

    a=(double)a, b=(double)b, c=(double)c;

    double eps=1e-7;

    vector<Point> Po;

    for (int i=0; i<points.size(); i++) {

        Point A((double)points[i][0], (double)points[i][1]);

        Po.push\_back(A);

    }

    double x0=-101, x1=101;

    while ((x1-x0)>eps) {

        double m1=x0+(x1-x0)/3;

        double m2=x1-(x1-x0)/3;

        Point P1(m1, (-1)\*((c+(a\*m1))/b)), P2(m2, (-1)\*((c+(a\*m2))/b));

        double f1=f(Po, P1);

        double f2=f(Po, P2);

        if (f1<f2) x1=m2;

        else x0=m1;

    }

    double p=(x0+x1)/2;

    Point X(p, (-1)\*((c+(a\*p))/b));

    double min\_distance=f(Po, X);

    return min\_distance;

}