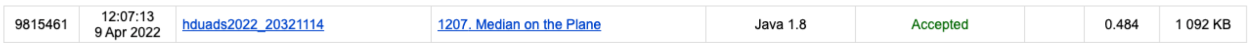
Laboratory work # 3

Student: HU Riqian
Student ID: 20321114
Timus Name: hduads2022_20321114

Mail: jhlxhrq@163.com

Problem # 1207. Median on the plane

Screenshot from Timus:



9815461	12:07:13 9 Apr 2022	hduads2022_20321114	1207. Median on the Plane	Java 1.8	Accepted		0.484	1 092 KB

Explanation of algorithm:

1. Firstly, I want to figure out the far left point P.

2. Through P, I can work out the angles with P and the other points.

3. Then, I select the median in the angles, this point is what I need.

4. I design the `Point` class, which I can use two integers to create the point object. And through `solveArc` I can figure out the angles.

5. Finally, I sort the angels to get my final answer.

Computational complexity of algorithm:

$$O(n^2)$$

Source code:

```java
import java.io.*;

public class MedianInPlane {

    public static void main(String[] args) throws IOException {
        new MedianInPlane().run();
    }

    StreamTokenizer in;
    PrintWriter out;
    static Point[] POINTS;
    static int SIZE;
    static Point INIT_POINT;

    int nextInt() throws IOException {
        in.nextToken();
        return (int)in.nval;
    }
    void run() throws IOException {
        in = new StreamTokenizer(new BufferedReader(new
InputStreamReader(System.in)));
        out = new PrintWriter(System.out);
        inputPoints();
        int initIndex = (sortInitPointIndex() + 1);
        int medianIndex = (sortMedianPointIndex() + 1);
        System.out.println(initIndex + " " + medianIndex);
    }

    void inputPoints() throws IOException {
        SIZE = nextInt();
        POINTS = new Point[SIZE];
        for (int i = 0; i < SIZE; i++) {
            POINTS[i] = new Point(nextInt(), nextInt());
        }
    }

    int sortInitPointIndex() {
        int index = 0;
        for (int i = 0; i < SIZE; i++) {
            if (POINTS[i].getX() <= POINTS[index].getX()) {
                index = i;
            }
        }
        INIT_POINT = POINTS[index];
        return index;
    }

    int sortMedianPointIndex() {
        int index = 0;
        double[] arcs = new double[SIZE];
        for (int i = 0; i < SIZE; i++) {
            arcs[i] = POINTS[i].solveArc(INIT_POINT);
        }
        for (int i = 0; i < SIZE; i++) {
            int count = 0;
            for (int j = 0; j < SIZE; j++) {
                if (arcs[i] > arcs[j]) {count++;}
            }
            if (count == SIZE / 2) {
                index = i;
                break;
            }
        }
        return index;
    }
}

class Point {
    private int x;
    private int y;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    public int getX() {
        return x;
    }
    public int getY() {
        return y;
    }
    public double solveArc(Point initPoint) {
        double opposite = y - initPoint.getY();
        double side = x - initPoint.getX();
        return (opposite == 0 && side == 0) ? -2 : Math.atan2(opposite, side);
    }
}
```