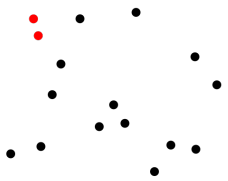
6 Advanced Problem: Finding the Closest Pair of Points

We strongly recommend you start solving advanced problems only when you are done with the basic problems (for some advanced problems, algorithms are not covered in the video lectures and require additional ideas to be solved; for some other advanced problems, algorithms are covered in the lectures, but implementing them is a more challenging task than for other problems).

Problem Introduction

In this problem, your goal is to find the closest pair of points among the given n points. This is a basic primitive in computational geometry having applications in, for example, graphics, computer vision, traffic-control systems.



Problem Description

Task. Given n points on a plane, find the smallest distance between a pair of two (different) points. Recall that the distance between points (x_1, y_1) and (x_2, y_2) is equal to $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Input Format. The first line contains the number n of points. Each of the following n lines defines a point (x_i, y_i).

Constraints. $1 \le n \le 10^5$; $-10^9 \le x_i, y_i \le 10^9$ are integers.

Output Format. Output the minimum distance. The absolute value of the difference between the answer of your program and the optimal value should be at most 10^{-3}. To ensure this, output your answer with at least four digits after the decimal point (otherwise your answer, while being computed correctly, can turn out to be wrong because of rounding issues).

Time Limits.

language	C	C++	Java	Python	C#	Haskell	JavaScript	Ruby	Scala
time in seconds	2	2	3	10	3	4	10	10	6

Memory Limit. 512MB.

Sample 1.

Input:

```
2
0 0
3 4
```

Output:

```
5.0
```

Explanation:

There are only two points here. The distance between them is 5.

Sample 2.

Input:

```
4
7 7
1 100
4 8
7 7
```

Output:

```
0.0
```

Explanation:
There are two coinciding points among the four given points. Thus, the minimum distance is zero.

Sample 3.

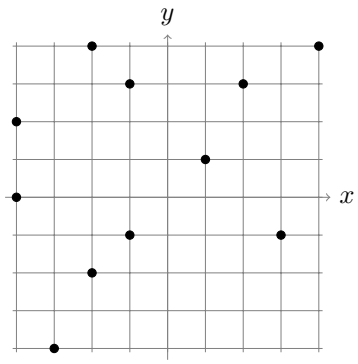
Input:

```
11
4 4
-2 -2
-3 -4
-1 3
2 3
-4 0
1 1
-1 -1
3 -1
-4 2
-2 4
```

Output:

```
1.414213
```

Explanation:
The smallest distance is $\sqrt{2}$. There are two pairs of points at this distance: $(-1, -1)$ and $(-2, -2)$; $(-2, 4)$ and $(-1, 3)$.



Starter Files

The starter solutions for this problem read the input data from the standard input, pass it to a blank procedure, and then write the result to the standard output. You are supposed to implement your algorithm in this blank procedure if you are using `C++`, `Java`, or `Python3`. For other programming languages, you need to implement a solution from scratch.

What To Do

A naive algorithm iterates through all pairs of points to find the minimal distance. Its running time is $\Theta(n^2)$ and it will not fit into the given time limit. Your goal therefore is to use the divide-and-conquer strategy to design a $O(n \log n)$ time algorithm. As usual, you might want to first partition the given set S of n points into two subsets S_1, S_2 of size $n/2$. You then make two recursive calls for the sets S_1 and S_2. After this, you know the minimal distances d_1 and d_2 for S_1 and S_2, respectively. But what if the minimal distance for the initial set S is realized for a pair of points p_1, p_2 such that $p_1 \in S_1$ and $p_2 \in S_2$? That is, you need to check whether there is a point $p_1 \in S_1$ and $p_2 \in S_2$ such that the distance between p_1 and p_2 is smaller than $\min\{d_1, d_2\}$. The corresponding algorithm is not easy to discover, but we still strongly encourage you to try to design it before reading a solution on the Internet or in a textbook (see, for example, [**?**, Exercise 2.32], or [**?**, Section 33.4], or [**?**, Section 5.4]).

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.