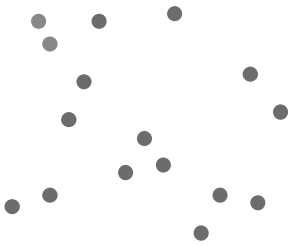
6.6 Closest Points

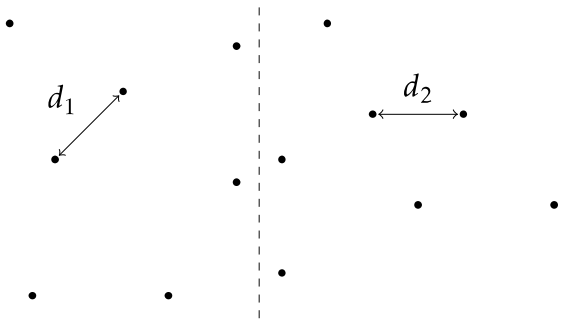
Closest Points Problem
Find the closest pair of points in a set of points on a plane.

Input: A set of points on a plane.
Output: The minimum distance between a pair of these points.

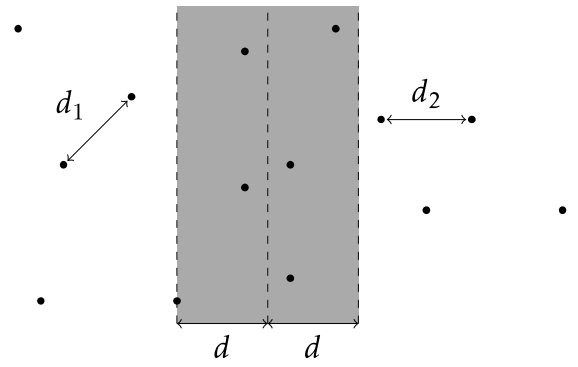


This computational geometry problem has many applications in computer graphics and vision. A naive algorithm with quadratic running time iterates through all pairs of points to find the closest pair. Your goal is to design an $O(n \log n)$ time divide and conquer algorithm.

To solve this problem in time $O(n \log n)$, let's first split the given n points by an appropriately chosen vertical line into two halves S_1 and S_2 of size $\frac{n}{2}$ (assume for simplicity that all x-coordinates of the input points are different). By making two recursive calls for the sets S_1 and S_2, we find the minimum distances d_1 and d_2 in these subsets. Let $d = \min\{d_1, d_2\}$.



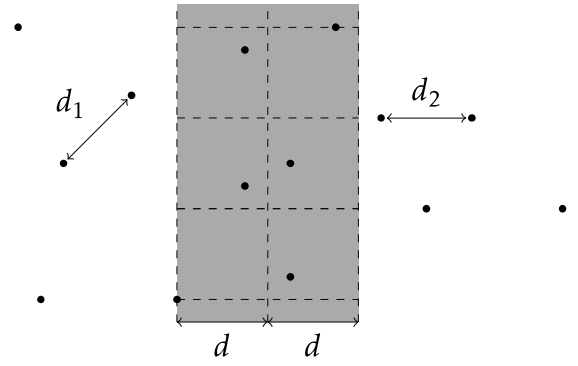
It remains to check whether there exist points $p_1 \in S_1$ and $p_2 \in S_2$ such that the distance between them is smaller than d. We cannot afford to check all possible such pairs since there are $\frac{n}{2} \cdot \frac{n}{2} = \Theta(n^2)$ of them. To check this faster, we first discard all points from S_1 and S_2 whose x-distance to the middle line is greater than d. That is, we focus on the following strip:



Stop and Think. Why can we narrow the search to this strip?

Now, let's sort the points of the strip by their y-coordinates and denote the resulting sorted list by $P = [p_1, \ldots, p_k]$. It turns out that if $|i-j| > 7$, then the distance between points p_i and p_j is greater than d for sure. This follows from the Exercise Break below.

Exercise Break. Partition the strip into $d \times d$ squares as shown below and show that each such square contains at most four input points.



This results in the following algorithm. We first sort the given n points by their x-coordinates and then split the resulting sorted list into two halves S_1 and S_2 of size $\frac{n}{2}$. By making a recursive call for each of the sets S_1 and S_2, we find the minimum distances d_1 and d_2 in them. Let $d = \min\{d_1, d_2\}$. However, we are not done yet as we also need to find the minimum distance between points from different sets (i.e, a point from S_1 and a point from S_2) and check whether it is smaller than d. To perform

such a check, we filter the initial point set and keep only those points whose x-distance to the middle line does not exceed d. Afterwards, we sort the set of points in the resulting strip by their y-coordinates and scan the resulting list of points. For each point, we compute its distance to the seven subsequent points in this list and compute d', the minimum distance that we encountered during this scan. Afterwards, we return $\min\{d, d'\}$.

The running time of the algorithm satisfies the recurrence relation

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + O(n \log n).$$

The $O(n \log n)$ term comes from sorting the points in the strip by their y-coordinates at every iteration.

Exercise Break. Prove that $T(n) = O(n \log^2 n)$ by analyzing the recursion tree of the algorithm.

Exercise Break. Show how to bring the running time down to $O(n \log n)$ by avoiding sorting at each recursive call.

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

Output format. The minimum distance. Recall that the distance between points (x_1, y_1) and (x_2, y_2) is equal to $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Thus, while the Input contains only integers, the Output is not necessarily integer and you have to pay attention to precision when you report it. 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 even correctly computed answer may fail to pass our grader because of the rounding errors).

Constraints. $2 \le n \le 10^5$; $-10^9 \le x_i, y_i \le 10^9$ are integers.

Sample 1.

Input:

```
2
0 0
3 4
```

Output:

```
5.0
```

There are only two points at distance 5.

Sample 2.

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
```

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