### At your university, an open-air get-together party, arranged just after Eid, brought together students from all batches. Thousands of past and present students attended, showing their love for their alma mater. Among them was a section where alumni sat, and you had one mission—collect salami from the alumni seniors.

### You knew that the probability of getting salami from any alumnus was the same, but there was a trick. If one alumnus saw you receiving salami from another, they might be more inclined to give as well. So, the best strategy was to find the two alumni sitting closest to each other and start from them.

### You had encountered a similar situation before with relatives, but that group was small. Back then, finding the closest pair was simple—just measure the distance between every possible pair and pick the smallest one. But now, with thousands of alumni present, brute force would take too long, and the party would end before you got any salami. You needed a smarter, more efficient approach. Can you write a program that finds the closest pair of alumni quickly so you can maximize your salami collection before the gathering ends?

### **Input Format:**

### The first line contains an integer **N** (5 ≤ N ≤ 100,000), representing the number of people in the alumni section. The next **N** lines each contain two space-separated integers, **x** and **y**, representing the coordinates of a particular alumnus’s position.

### **Output Format:**

### In one line, print the **IDs** of the two alumni who are sitting closest to each other (1-based index, corresponding to their order in input) and the **Euclidean distance** between them, rounded to six decimal places, separated by a space.

| Sample Input | Sample Output |
| --- | --- |
| 6  2 3  4 7  5 5  5 1  3 2  0 3 | 1 5 1.414214 |

### You have to submit the .py or .java or .cpp file in the submission form.

Hint: [Closest Pair of Points (Divide and Conquer) Explained](https://www.youtube.com/watch?v=6u_hWxbOc7E)