Galactic Trade Routes

The time for an interstellar trade route has come! The plaque sent out on the Pioneer probes was decoded by an alien civilization, and they have imparted to us the secrets of interstellar travel. Your job is to find the two stars nearest to each other in our galaxy, and begin your trade route between these two star systems. We'll assume that you can pick any two stars to begin your trade route; you don't have to start at Earth. Being as our galaxy has a lot of stars (up to 400 billion), your algorithm will need to be efficient.

For this assignment, we'll assume that our galaxy is 2-dimensional, so you'll need to handle star locations in 2D space. The coordinates will range from -50,000 to +50,000 (the Milky Way is 100,000 light years in diameter). Your trading ships cannot travel more than 10,000 light years, and any input case that has no stars closer than this will need to indicate such.

Input

The input will consist of a number of test cases (more than one per file!). The first line of each test case will contain n, the number of stars for that case: $2 \leq n \leq 100,000$. The following n lines will contain the (x,y) coordinates of successive stars. Each x and y coordinate will be a real number with a minimum value of -50,000 and a maximum value of 50,000. The coordinates will be given to two decimal places of accuracy. The input will terminate when n=0

Output

For each test case, your output should contain a single line that is the minimum distance between two points, to four digits of accuracy after the decimal point. You should not print the stars that are closest; only the distance. If there are no points closer than a distance of 10,000, you should print out 'infinity'.

Activities and a Report

Submit a PDF file with a brief but clear report that addresses the following:

1. A brute-force solution to this problem is very straight-forward (and very little code): compare all possible pairs of stars, and remember the smallest distance. Like you did in the sorting HW, use a brute-force algorithm on an input of some reasonably large size X, then something that's size 2X, and something that's size 3X. Write down the time it took to find the closest-pair for each run, and in a short sentence explain whether or not the run-times match what you expect, based on the time-complexity of the brute-force approach. (Important: don't include the cost of reading data from the input file in your run-times. You may also cut the brute-force short if it takes too long and report the time as greater than threshold, e.g., > 60 seconds)
2. When programming the divide and conquer solution to this problem, it's fairly easy to fail to properly handle the "strip" part of the solution, and instead end up with a quadratic solution. Repeat what

you did in the previous step for your divide and conquer solution and report the run-times, and argue that these results show that your algorithm is $o(n^2)$ and not $\Theta(n^2)$. (Note: a typo here was updated on 9/30/2021. It incorrectly said $o(n)$ before and not $o(n^2)$.)

3. *NOTE: This report can be very short. Just report a few times executing your code and write a paragraph or so discussing the times that you saw and what they imply. You do not need to use LaTeX for this report, but the submission must be a PDF file.*

Grading

You will pass this assignment if you pass enough test cases AND submit an acceptable report. You must have the following:

1. Submitted code that uses divide-and-conquer and passes at least 8 out of 10 of the test cases.
2. Submitted a PDF file that lists run-times for your code and contains at least one paragraph describing the results you found.

Sample Input

```
2
1.25 6.11
1.64 1.50
2
0 0
0 10001
3
-6.47 2.24
8.90 6.53
-1.45 5.05
4
2.77 -9.81
-0.19 4.85
1.38 9.05
-4.95 3.93
10
8.15 6.21
-5.64 -4.31
-0.03 7.05
5.98 -4.64
1.49 -1.59
4.34 0.67
-2.79 -0.93
-7.41 2.89
-1.79 -6.22
-0.98 1.74
0
```

Sample Output

```
4.6265
infinity
5.7530
4.4838
3.2257
```