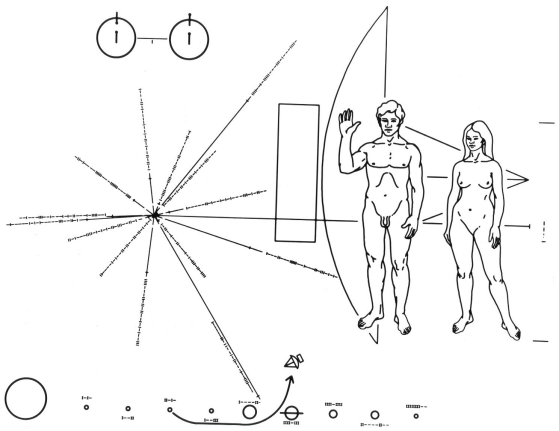
Galactic Trade Routes

The time for an interstellar trade route has come! The plaque sent out on the Pioneer probes (see image at right) 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 starts 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.



We will assume that the galaxy is 2-dimensional. This is not an unreasonable assumption for this problem, as it's diameter is 1,000 times more than it's thickness. The coordinates are on a 2-D plane, and in light-years from the galactic center; thus they 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.

Your task is to determine the distance between the two closest of the provided star coordinates.

Input

The input will consist of a number of test cases. The first line of each test case will contain n, the number of points in that case: $2 \leq n \leq 10,000$. The following n lines will contain the (x, y) coordinates of successive points. 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 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'.

Requirements

This homework **MUST** be a divide-and-conquer algorithm. A solution that is not divide-and-conquer will receive zero credit.

An $\Theta(n^2)$ algorithm will not work for this assignment, as there will specifically not be enough running time for such an algorithm to complete.

You cannot write this assignment in Scheme, PHP, or Ruby – those languages are just too slow for this assignment. You can use C, C++, and Java. Our reference solution was written in Java (the slowest of the three allowed languages), and the execution time is based on that solution. The test case we are using to judge the problems took about 10 seconds with our $\Theta(n \log n)$ solution, and about 30 seconds with our $\Theta(n^2)$ solution.

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
```

How to start

You should first generate some sample input cases – just generate lots of random points, and make sure the input file format is correct. You will want test cases of a large size and a small size. You can write a $\Theta(n^2)$ algorithm to find the minimum distance, and print that out. This will (hopefully!) ensure that you have the correct answer. You can then use this solution to test your $\Theta(n \log n)$ solution to check for accuracy. Once you think you have it working, try generating many different test cases (different sizes, etc.), and seeing if your two solutions – the $\Theta(n^2)$ one and the $\Theta(n \log n)$ one – yield the same results. A good debugger, and print statements tracing your program's execution will be your friend.

The algorithm is as described in lecture, and is also on pages 1039–1044 of the textbook; although the lecture explanation used slightly different notation than the text, the idea is still the same.