Julia Paola Orduño Ahumada
A01630895

Project 1: *Lazy Snail*

Context

Here, in Romania, all snails are lazy. Take Wally the Snail, for example. He has to visit **N** friends which are located at distinct coordinates in the plane. But since he is so lazy, he doesn't want to leave his house. He said that he will go visit his friends if someone can show him the right path to follow.

He wants to leave his house, visit all of his friends exactly once and then return to his house. Between 2 friends' houses or between his house and a friend's house, he walks on the straight line which connects them. 'Is that all ?', someone asked. Wally realized that this would be too easy, so he added that, during his trip, no two line segments along which he travels should cross (except for every 2 consecutive segments, which cross at one end). You must find a path for Wally, so he can go visit all of his friends (although he doesn't want to).

Input

On the 1st line of input, there will be 2 real numbers: X and Y, separated by a blank, representing the coordinates of Wally's house. On the 2nd line, there will be an integer number: N (2 ≤ N ≤ 1000), the number of friends Wally has to visit. On the next N lines, there will be 3 numbers, separated by blanks: X , Y and ID. ID will be an integer number, representing the ID of one of Wally's friends. X and Y will be 2 real numbers, representing the coordinates of Wally's friend's house (they will be given with at most 3 decimal digits and will be in the range -100000 .. 100000).

All IDs are unique, between 1 and N. No 3 friends (including Wally) have their houses on the same straight line.

Output

You should output N+2 lines: the IDs of the friends whose houses Wally is about to visit, in the order he visits them. Start with

Wally's ID, continue with the ID of the friend he visits first and so on. Finish with Wally's ID. Wally has ID 0.

If there is no solution, then print a single line, containing the number -1.

Breakdown

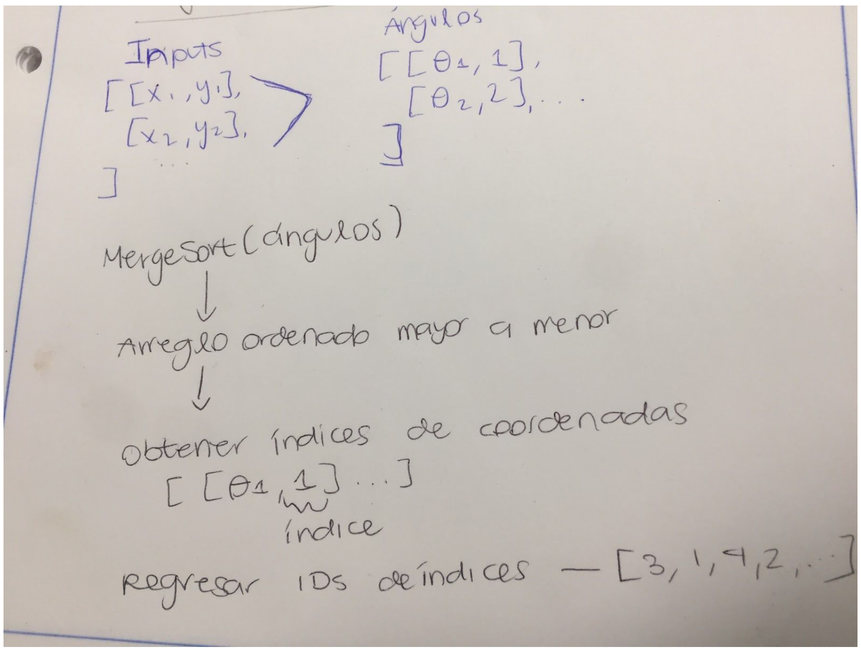
- Coordinates are real numbers (doubles).
- The restrictions for coordinates are: maximum 3 decimals and a range from -100000 to 100000.
- How to fix to 3 decimals?
- Reads N (integer) number of friends, each friend has an ID (integer).
- Segments mustn't cross.

First case:

- Wally's coordinates are (0,0) and his friends' x values are: x > 0.

  Solution

  - Get friends' coordinates, calculate their angles to the origin (0,0).
  - Sort angles from large to small using merge sort.


Inputs
[[x1,y1],
[x2,y2],
...
]
Angulos
[[θ1, 1],
[θ2, 2], ...
]
MergeSort(angulos)
Arreglo ordenado mayor a menor
obtener indices de coordenadas
[[θ1, 1] ...]
indice
Regresar IDs de indices — [3, 1, 4, 2, ...]

Second case:

- Wally's coordinates are (0,0) and his friends' x values are:

-100000 < x < 100000.

Solution

- Added a conditional to previous procedure. After calculating the coordinates angle:

  ```
  if( xi < XWally )
        angleXi -= 180.
  ```

  This way the criteria to get the correct path will be sorting the angles from large to small and from 90º to -270º.

Solution

First, I initialize the required variables:

- x, y for Wally’s coordinates.
- n for Wally’s number of friends.
- coordinates, bidimensional double array to store Wally’s friends’ coordinates and their IDs.
- angles, bidimensional double array to store the angles from Wally’s friends’ coordinates to the origin, and to have a reference to their indices in coordinates array.
- path, integer array to store friends’ IDs in the correct order.

Then, I get the required data from the user, rounding each coordinate to 3 decimals and calculating their angles with the next formula:

```
arctan(x/y)*180.0/PI
```

Also, I make the validation to get the correct angles when the x coordinate is smaller than Wally’s x, so I get a range from 90º to -270º.

Using merge sort algorithm, I sort the angles array from largest to smallest.

Finally, I fill the path array, using the index references from the angles array to the coordinates and get the friends’ IDs. Then, I just print the output as is requested,starting with 0 (Wally’s ID), the path, and finalizing with 0 again, because it’s a cycle.