Introduction

This is a programming challenge and we would like you to use Python 3 to solve it. We expect the challenge to take you around 2 hours, but please indicate the amount of time you spend on the assignment when you return your solution.

Problem

You are given a set of two-dimensional rectangular boxes on a two-dimensional Cartesian plane with the following assumptions and constraints:

- All boxes are axis-aligned, i.e. each box can be defined in terms of the Cartesian coordinates of its minimum corner $\{a, b\}$ and its maximum corner $\{c, d\}$ such that $a < c$ and $b < d$.
- Boxes may be contained within other boxes or may be disjoint from one another, i.e. boxes do not overlap or touch one another.
- The boxes therefore partition the plane into a number of regions. The unbounded region, which lies outside of all the boxes, is classified as sea.
- All other regions are classified either as sea or as land, subject to the constraint that adjacent regions must not share the same classification.

The task is to read the definition of the rectangles from standard input formatted as defined below, and output the number of regions classified as land to standard output.

If you make any other assumptions as part of your solution, please make comments in the code. If there are further considerations that might affect the memory use or performance of your solution then do make a note of them.

Submission

When submitting your solution please return all of your code, including any tests that you might have written and any documentation that you feel is appropriate. Please archive your solution either as a .tar or .zip file.

Input Format

Some sample data is provided below for you to use when testing your solution. The first line contains an integer N denoting the number of boxes. The next N lines contain the four floating point values a,b,c,d that define Cartesian coordinates of the minimum corner and maximum corner of each box separated by a single space.

```
<N>
a1 b1 c1 d1
...
aN bN cN dN
```

Some sample data is provided on the next page for you to use when testing your solution.

Example

Sample input:

```
14
1.0 1.0 10.0 6.0
1.5 1.5 6.0 5.0
2.0 2.0 3.0 3.0
2.0 3.5 3.0 4.5
3.5 2.0 5.5 4.5
4.0 3.5 5.0 4.0
4.0 2.5 5.0 3.0
7.0 3.0 9.5 5.5
7.5 4.0 8.0 5.0
8.5 3.5 9.0 4.5
3.0 7.0 8.0 10.0
5.0 7.5 7.5 9.5
5.5 8.0 6.0 9.0
6.5 8.0 7.0 9.0
```

Expected output:

```
9
```