Obstacle Courses

The Association of Obstacle Course Racing wants to create a park for their members. The association has divided the park into n different *zones*. Each zone i is described by a unique point given by integer coordinates (x_i, y_i) on the park. The park has two special zones; the *start-zone* is the only zone where members can enter the park and the *end-zone* is the only zone where members can leave the park. To advance from a zone to the next one must complete an *obstacle*. The association has decided on 4 different types of obstacles for their park with the following restrictions:

The NEWS[1] Climbing Wall (O1). Can be built from a zone A to any other zone on the same vertical or horizontal line as A.
The Bungee Slingshot (O2). Can be built from a zone A to the zone furthest away from A (euclidean distance). If multiple zones are equally far away, it can only be built to the *first* of these (ie. the first zone in the order they come in the input)
The Salmon Ladder Slide (O3). Can be built from a zone A to any other zone B if there are at least two other zones in between A and B (ie. exactly on the line segment between A and B there must be at least 2 other zones).
The Heavenly Rope Climb (O4). Can be built from any zone directly to the end-zone.

The park is divided into different *courses*. A course is a sequence of obstacles starting from the start-zone and ending at the end-zone. To avoid members colliding two courses cannot share an obstacle, but they may share zones.

The association has already picked the location of the zones. Your task is to decide which obstacles should be built between a pair of zones. Multiple obstacles may be built between the same two zones - even obstacles of the same type. However, there is a limit on how many of each kind of obstacle can at most be built from each of the zones. Your target is to design the Obstacle course with as many courses as possible.

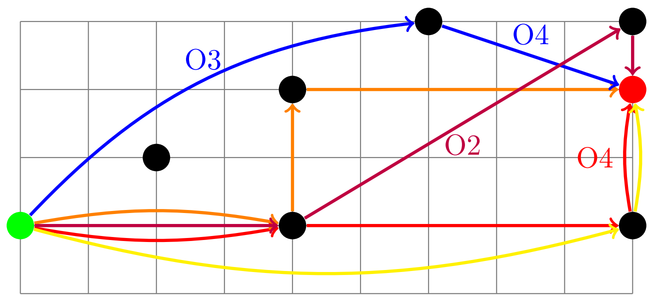

Figure 1: Illustration of the first sample test with an optimal solution. Zones are marked by circles. The green circle is the start-zone. The red circle is the end-zone. An arrow corresponds to an activity between two zones. The label on an arrow tells the type of activity in this solution (arrows without labels are all O1). This solution has 5 courses (colored in different colors). This solution is optimal, but other optimal solutions may exist.

[1] North, East, West, South

Input The input has the following format. All numbers are integers:

```
Line 1: "n". n is the number of zones.
        (2 <= n <= 100 for the hand-in and 2 <= n <= 100.000 for the competition)
Next n lines: "X Y O1 O2 O3 O4". (X, Y) is the position of the zones.
              O1, O2, O3, and O4 are the restrictions on the number of
              each kind of obstacle starting from the zone.
              (0 <= X, Y <= 100.000 and 0 <= O1, O2, O3, O4 <= 100.000)
```

The first zone in the input is the start-zone and the last zone is the end-zone. You can assume no two zones have the same position. Your program should read the input from standard input.

Output Your program should output the maximum possible number of courses the can have (to standard output).

Sample test This is an example of the input. The answer for this test is 5 (see Figure 1):

```
8
0 1 4 0 2 0
2 2 1 2 3 4
4 1 3 1 1 0
4 3 1 0 0 1
6 4 0 0 1 1
9 1 1 0 0 2
9 4 2 0 0 1
9 3 0 0 0 0
```

More sample tests can be found on CodeJudge.

Hand-in The hand-in is individual. You must hand-in your solution before **November 27, 23.59** on CodeJudge (`https://dtu.codejudge.net/02110-e22/exercise/16075`). Hand-in under the exercise "Assignment 2".

Competition You can submit your solution to a competition among the students in the course. The rank in the competition will be determined based on (1) the number of solved test cases (2) the total sum of your programs running time on all test cases it solved. (2) will only apply, if two people solve the same number of test cases. The competition will be based on different test data than the tests we provide before the deadline. To join the competition, you must **also** hand-in your solution on CodeJudge under the exercise "Competition".