

Subtask 1:

For around 16 points you can do a binary search on each row to try and find the missing element. Overall $n \cdot \lceil \log n \rceil$ queries.

For 30 points you need to do something smarter. Notice that in the binary search solution, in each row you are searching for the greatest j such that $a_{i,j} \leq x$. For 30 points, you still need to compute that but in a smart way. Say L_i is the highest integer j such that $a_{i,j} \leq x$, or -1 if such a number does not exist. Notice that $L_i \geq L_{i+1}$, since if $a_{i+1,j} \leq x$ then $a_{i,j} \leq x$.

This means that if we know L_i , and want to find L_{i+1} we only need to consider j smaller than L_i . This leads us to the following algorithm. To compute L_0 we start from the corner, $(0, n-1)$ and search left to find L_0 . Then to find L_1 , we search from the point $(1, L_0)$ leftwards and then for L_2 we start from the point $(2, L_1)$ and so on. See the code for a precise implementation of this algorithm.

In the worst case, we will use $2n-1$ queries. Since in each step we either go down or left so overall we make at most $(n-1) \cdot 2$ steps, and we also ask one query for the initial query so overall $2n-1$.

Subtask 2:

Some contestants got around 15 points for this subtask by randomizing the position of the cell where x is located. Because of the nature of the grader, this algorithm can technically get any number of points between 0 and 70, but most likely it will get something in the range of 0 and 15.

The way one can think about this question is the following. With each query Faruk gains some information and reduces the total number of positions that x can be at. We know that there exists an algorithm which always finds x in $2n-1$ queries, so we should try and start with at least $2n-1$ possible positions that x could be at then make him eliminate at most one position from the possible set of positions in each query.

WLOG, say $x = 1$.

The initial set of positions that x could be at is of course, every possible position. However we only consider the set which contains $(0, n-1), (1, n-2), \dots, (n-1, 0)$ and $(1, n-1), (2, n-2), \dots, (n-1, 1)$. These cells are marked with # below for $n = 4$.

```

0 ...#
1 ..##
2 .##.
3 ##..
  0123

```

For cells not in the initial set we always answer 2 if $i + j > n$ (the upper diagonal of #), otherwise we answer with 0 (the lower diagonal of #). Replace any of the marked cells with x , and all the other marked cells with 0 or 2 (according to the rule above) and the grid will always be valid. This leads us to the following algorithm.

Once Faruk asks for a query in something in the set, we answer with 1 if it is the last cell remaining, otherwise we answer 2 if $i + j = n$ and 0 otherwise.

With this strategy, we guarantee at least $2n - 1$ queries and consistency among queries regardless of Faruk's strategy.

Variations of this strategy which consider only one diagonal or only a few of cells still work, but score fewer points.