B. Game On Ranges

There are a lot of solutions to this problem. The solution I described below might be the simplest in my opinion.

Tutorial

1623B - Game on Ranges

If the length of a range $[l, r]$ is 1 (that is, $l = r$), then $d = l = r$. Otherwise, if Bob picks a number d, then Alice **has to put** the sets $[l, d - 1]$ and $[d + 1, r]$ (if existed) back to the set. Thus, there will be a moment that Alice picks the range $[l, d - 1]$ (if existed), and another moment to pick the range $[d + 1, r]$ (if existed) as well.

Using the above observation, for each range $[l, r]$, we can iterate the number d from l to r, check if both range $[l, d - 1]$ (if $d > l$) and $[d + 1, r]$ (if $d < r$) existed in the Alice's picked ranges. Or in other words, check if these ranges are given in the input.

For checking, we can either use `set` data structures supported in most programming languages or simply use a 2-dimensional array for marking the picked ranges. The time complexity is, therefore, $O(n^2)$.

This problem can be solved in $O(n \log n)$ as well, and even $O(n)$ with some black magic like counting sort, but that is not required during the contest.

Problem note

- The game process is actually inspired by *Quick sort*: the range, picked by Alice, is the sorting range, and the number, picked by Bob, is the pivot.
- Testers really like sorting. Some of the testers demand **order** for the input, so they need to do the sorting. But Nah, that is totally not required :)

Pascal solution: 140968967. C++ solution: 140968942