757. Set Intersection Size At Least Two

Solved

Hard | Topics | Companies

You are given a 2D integer array `intervals` where `intervals[i] = [start_i, end_i]` represents all the integers from `start_i` to `end_i` inclusively.

A **containing set** is an array `nums` where each interval from `intervals` has **at least two** integers in `nums`.

- For example, if `intervals = [[1,3], [3,7], [8,9]]`, then `[1,2,4,7,8,9]` and `[2,3,4,8,9]` are **containing sets**.

Return *the minimum possible size of a containing set.*

Example 1:

```
Input: intervals = [[1,3],[3,7],[8,9]]
Output: 5
Explanation: let nums = [2, 3, 4, 8, 9].
It can be shown that there cannot be any containing array
of size 4.
```

Example 2:

```
Input: intervals = [[1,3],[1,4],[2,5],[3,5]]
Output: 3
Explanation: let nums = [2, 3, 4].
It can be shown that there cannot be any containing array
of size 2.
```

Example 3:

```
Input: intervals = [[1,2],[2,3],[2,4],[4,5]]
Output: 5
Explanation: let nums = [1, 2, 3, 4, 5].
It can be shown that there cannot be any containing array
of size 4.
```

Constraints:

- `1 <= intervals.length <= 3000`
- `intervals[i].length == 2`
- `0 <= start_i < end_i <= 10^8`

Python:

```
from typing import List

class Solution:
    def intersectionSizeTwo(self, intervals: List[List[int]]) -> int:
        n = len(intervals)
        if n == 0:
            return 0
        intervals.sort(key=lambda x: (x[1], x[0]))
        res = []
        res.append(intervals[0][1] - 1)
        res.append(intervals[0][1])
        for i in range(1, n):
            start = intervals[i][0]
```

```
            end = intervals[i][1]
            last = res[-1]
            second_last = res[-2]
            if start > last:
                res.append(end - 1)
                res.append(end)
            elif start == last:
                res.append(end)
            elif start > second_last:
                res.append(end)
        return len(res)
```

JavaScript:

```
/**
 * @param {number[][]} intervals
 * @return {number}
 */
var intersectionSizeTwo = function(intervals) {
    if (!intervals || intervals.length === 0) return 0;
    intervals.sort((a, b) => {
        if (a[1] !== b[1]) return a[1] - b[1];
        return b[0] - a[0];
    });
    let res = 0;
    let a = -Infinity, b = -Infinity;
    for (const [l, r] of intervals) {
        if (l > b) {
            res += 2;
            a = r - 1;
            b = r;
        } else if (l > a) {
            res += 1;
            a = b;
            b = r;
        }
    }
    return res;
};
```

Java:

```
class Solution {
        public int intersectionSizeTwo(int[][] intervals) {
        int n = intervals.length;
        Arrays.sort(intervals, (a, b) -> a[1] == b[1] ? a[0] - b[0] : a[1] - b[1]); // Sort intervals: 1- end
2- start- O(nlogn)
```

```
        List<Integer> res = new ArrayList<>();
        res.add(intervals[0][1] - 1); // Add one before end
        res.add(intervals[0][1]); // Add end
        for (int i = 1; i < n; i++) { // O(n)
            int start = intervals[i][0], end = intervals[i][1], size = res.size(), last = res.get(size - 1),
secondLast = res.get(size - 2);
            if (start > last) { // We need to add two fresh points
                res.add(end - 1);
                res.add(end);
            } else if (start == last) res.add(end); // We already added one. We need to add the end of
this interval
            else if (start > secondLast) res.add(end); // We already added last. We need one more
        }
        return res.size();
    }
}
```