757. Set Intersection Size At Least Two

DescriptionHintsSubmissionsDiscussSolution

An integer interval `[a, b]` (for integers `a < b`) is a set of all consecutive integers from `a` to `b`, including `a` and `b`.

Find the minimum size of a set S such that for every integer interval A in `intervals`, the intersection of S with A has size at least 2.

Example 1:

```
Input: intervals = [[1, 3], [1, 4], [2, 5], [3, 5]]
Output: 3
Explanation:
Consider the set S = {2, 3, 4}.  For each interval, there are at least 2
elements from S in the interval.
Also, there isn't a smaller size set that fulfills the above condition.
Thus, we output the size of this set, which is 3.
```

Example 2:

```
Input: intervals = [[1, 2], [2, 3], [2, 4], [4, 5]]
Output: 5
Explanation:
An example of a minimum sized set is {1, 2, 3, 4, 5}.
```

Note:

1. `intervals` will have length in range `[1, 3000]`.
2. `intervals[i]` will have length `2`, representing some integer interval.
3. `intervals[i][j]` will be an integer in `[0, 10^8]`.

- Difficulty:Hard
- Total Accepted:866
- Total Submissions:2.6K
- Contributor:zhuofuch@usc.edu
-
- Subscribe to see which companies asked this question.

Related Topics Greedy

A few of points:

1. Sort the array according to their end point in ascending order, AND if two intervals have same end, sort them according to their start point in descending order. e.g [[1,5],[4,5],[5,9], [7,9],[9,10]] => [[4,5], [1,5], [7,9], [5,9] , [9,10]]

2. Greedy to get the rightmost two point

```
class Solution {
    public int intersectionSizeTwo(int[][] intervals) {
        int res = 0;
        if (intervals == null || intervals.length == 0) {
            return res;
        }
        Arrays.sort(intervals, (a, b)-> a[1] != b[1] ? a[1] - b[1] : b[0] -
a[0]);
        // known two rightmost point in the set/res
        int left = intervals[0][1] - 1;
        int right = intervals[0][1];
        res += 2;
        for (int i = 1; i < intervals.length; i++) {
            int[] curr = intervals[i];
            // 1. one element of the set is in the interval
            // 2. no elemnet of the set is in the interval
            if (left < curr[0] && curr[0] <= right) {
                res++;
                left = right;
                right = curr[1];
            } else if (curr[0] > right) {
                res += 2;
                left = curr[1] - 1;
                right = curr[1];
            }
        }
        return res;
    }
}
```