986. Interval List Intersections

Medium 1939 60 Add to List Share

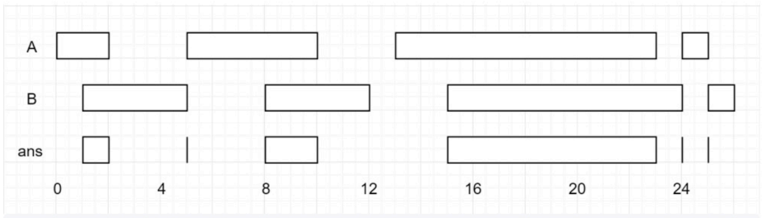
You are given two lists of closed intervals, `firstList` and `secondList`, where `firstList[i] = [start_i, end_i]` and `secondList[j] = [start_j, end_j]`. Each list of intervals is pairwise **disjoint** and in **sorted order**.

Return *the intersection of these two interval lists*.

A **closed interval** `[a, b]` (with `a < b`) denotes the set of real numbers `x` with `a <= x <= b`.

The **intersection** of two closed intervals is a set of real numbers that are either empty or represented as a closed interval. For example, the intersection of `[1, 3]` and `[2, 4]` is `[2, 3]`.

Example 1:



```
Input: firstList = [[0,2],[5,10],[13,23],[24,25]], secondList = [[1,5],[8,12],[15,24],[25,26]]
Output: [[1,2],[5,5],[8,10],[15,23],[24,24],[25,25]]
```

Example 2:

```
Input: firstList = [[1,3],[5,9]], secondList = []
Output: []
```

Example 3:

```
Input: firstList = [], secondList = [[4,8],[10,12]]
Output: []
```

Example 4:

```
Input: firstList = [[1,7]], secondList = [[3,10]]
Output: [[3,7]]
```

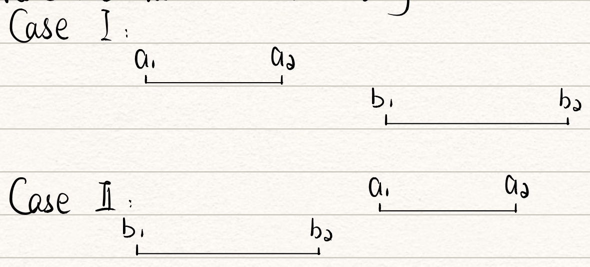
Constraints:

- `0 <= firstList.length, secondList.length <= 1000`
- `firstList.length + secondList.length >= 1`
- $0 \le start_i < end_i \le 10^9$
- $end_i < start_{i+1}$
- $0 \le start_j < end_j \le 10^9$

1. Sort

2. Use two pointer to find intersection in A & B.

There are two case when they don't have intersection



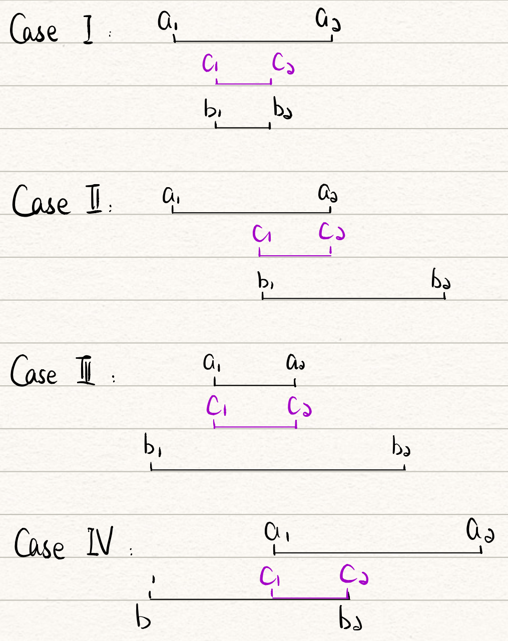
if $b_2 < a_1$ or $a_2 < b_1$:
$[a_1, a_2]$ and $[b_1, b_2]$ dependent

$\Downarrow$

if $b_2 \ge a_1$ and $a_2 \ge b_1$:
$[a_1, a_2]$ and $[b_1, b_2]$ has intersaction

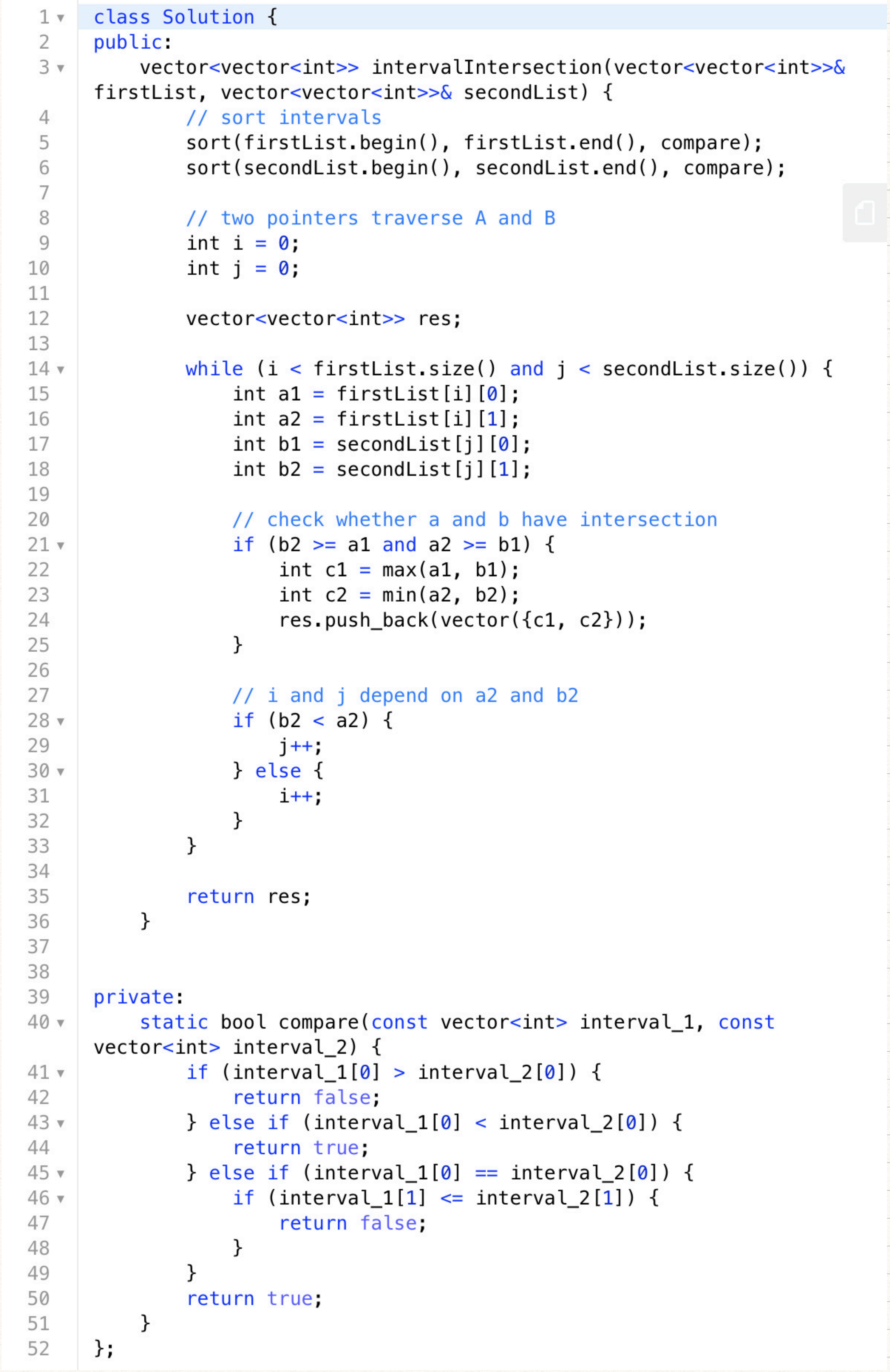
There are four cases with intersaction.

$c_1 = \max(a_1, b_1)$
$c_2 = \min(a_2, b_2)$



i and j is depended on a_2 and b_2:

```
if b2 < a2 :
    j++
else :
    i++
```



```cpp
class Solution {
public:
    vector<vector<int>> intervalIntersection(vector<vector<int>>& firstList, vector<vector<int>>& secondList) {
        // sort intervals
        sort(firstList.begin(), firstList.end(), compare);
        sort(secondList.begin(), secondList.end(), compare);

        // two pointers traverse A and B
        int i = 0;
        int j = 0;

        vector<vector<int>> res;

        while (i < firstList.size() and j < secondList.size()) {
            int a1 = firstList[i][0];
            int a2 = firstList[i][1];
            int b1 = secondList[j][0];
            int b2 = secondList[j][1];

            // check whether a and b have intersection
            if (b2 >= a1 and a2 >= b1) {
                int c1 = max(a1, b1);
                int c2 = min(a2, b2);
                res.push_back(vector({c1, c2}));
            }

            // i and j depend on a2 and b2
            if (b2 < a2) {
                j++;
            } else {
                i++;
            }
        }

        return res;
    }


private:
    static bool compare(const vector<int> interval_1, const vector<int> interval_2) {
        if (interval_1[0] > interval_2[0]) {
            return false;
        } else if (interval_1[0] < interval_2[0]) {
            return true;
        } else if (interval_1[0] == interval_2[0]) {
            if (interval_1[1] <= interval_2[1]) {
                return false;
            }
        }
        return true;
    }
};
```