**Problem Name:** Interval list Intersection

**Topics:** Array, Two-Pointers

**Companies:** Facebook, Microsoft, Amazon, Google, Apple, Bloomberg, Yandex, Adobe,

**Level:** Medium

**Language:** C++

**Problem Statement:** You are given two lists of closed intervals, firstList and secondList, where firstList[i] = [starti, endi] and secondList[j] = [startj, endj]. 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].

**Input Format:**

First line of the input contains integer n (length of first 2D Vector)

Second line contain 2n space separated integer values of first vector.

Third line contain integer value m (length of second 2D Vector)

Fourth line contain 2m space separated integer values of second vector.

Ex:

4

0 2 5 10 13 23 24 25

4

1 5 8 12 15 24 25 26

**Output Format:** Print 2D vector representing intersection of two closed interval.

**Constraints:**

* 0 <= firstList.length, secondList.length <= 1000
* firstList.length + secondList.length >= 1
* 0 <= starti < endi <= 109
* endi < starti+1
* 0 <= startj < endj <= 109
* endj < startj+1

**Examples:**

**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]]

**Brute force Solution:**

**Explanation:** we're just moving through all element one by one and checking if it is valid then push it into answer vector 'v'.

**Code:**

#include <bits/stdc++.h>

using namespace std;

vector<vector<int>> intervalIntersection(vector<vector<int>>&l1, vector<vector<int>>&l2) {

    vector<vector<int>>v;

    for(int i=0;i<l1.size();i++){

        for(int j=0;j<l2.size();j++){

            if(l1[i][1]<l2[j][0])

                break;

            int x=max(l1[i][0],l2[j][0]);

            int y=min(l1[i][1],l2[j][1]);

            if(x<=y)

                v.push\_back({x,y});

        }

    }

    return v;

}

int main() {

    int m, n;

    cin>>n;

    vector<vector<int>> first;

    for(int i=0; i<n; i++){

        vector<int> temp(2);

        for(int j=0; j<2; j++){

            cin>>temp[j];

        }

        first.push\_back(temp);

    }

    cin>>m;

    vector<vector<int>> second;

    for(int i=0; i<m; i++){

        vector<int> temp(2);

        for(int j=0; j<2; j++){

            cin>>temp[j];

        }

        second.push\_back(temp);

    }

    vector<vector<int>> result;

    result = intervalIntersection(first, second);

    for(int i=0; i< result.size(); i++){

        for(int j=0; j<2; j++){

            cout<<result[i][j]<<" ";

        }

    }

    return 0;

}

**Time Complexity**: O(m\*n)

**Space Complexity:** O(1)

**Optimized Solution:**

**Explanation:**

Idea issimple, we initialise 2 ponters, one for each of the lists

1. Limits of intersection of an interval = [max(left limit of lists), min(right Limit of lists)] = [mx, mn]  
   **here we can see if mn > mx then interval will not be valid so ingnore it**
2. Now check if mn==right limit of first lists then current interval of first list is finished so move forward in first list  
   similarly check for the second list.

ex. firstList = [1, 5]  
secondList = [2, 4]  
intersection = [max(1,2), min(4,5)] = [2, 4]  
we can see that in the intersection, right limit = secondList's right limit so move forward in secondList

**Code:**

#include <bits/stdc++.h>

using namespace std;

vector<vector<int>> intervalIntersection(vector<vector<int>>& A, vector<vector<int>>& B) {

    vector<vector<int>> v;

    int i=0,j=0;

    while(i<A.size() && j<B.size()){

        int l=max(A[i][0], B[j][0]);

        int u=min(A[i][1], B[j][1]);

        if(l<=u)

            v.push\_back({l,u});

        if(A[i][1] < B[j][1])

            i++;

        else

            j++;

    }

    return v;

}

int main() {

    int m, n;

    cin>>n;

    vector<vector<int>> first;

    for(int i=0; i<n; i++){

        vector<int> temp(2);

        for(int j=0; j<2; j++){

            cin>>temp[j];

        }

        first.push\_back(temp);

    }

    cin>>m;

    vector<vector<int>> second;

    for(int i=0; i<m; i++){

        vector<int> temp(2);

        for(int j=0; j<2; j++){

            cin>>temp[j];

        }

        second.push\_back(temp);

    }

    vector<vector<int>> result;

    result = intervalIntersection(first, second);

    for(int i=0; i< result.size(); i++){

        for(int j=0; j<2; j++){

            cout<<result[i][j]<<" ";

        }

    }

    return 0;

}

**Time Complexity**: O(N)

**Space Complexity:** O(1)