<https://leetcode.com/problems/corporate-flight-bookings/description/>

There are n flights that are labeled from 1 to n.

You are given an array of flight bookings bookings, where bookings[i] = [firsti, lasti, seatsi] represents a booking for flights firsti through lasti (**inclusive**) with seatsi seats reserved for **each flight** in the range.

Return **an array**answer**of length**n**, where**answer[i]**is the total number of seats reserved for flight**i.

**Example 1:**

**Input:** bookings = [[1,2,10],[2,3,20],[2,5,25]], n = 5

**Output:** [10,55,45,25,25]

**Explanation:**

Flight labels: 1 2 3 4 5

Booking 1 reserved: 10 10

Booking 2 reserved: 20 20

Booking 3 reserved: 25 25 25 25

Total seats: 10 55 45 25 25

Hence, answer = [10,55,45,25,25]

**Example 2:**

**Input:** bookings = [[1,2,10],[2,2,15]], n = 2

**Output:** [10,25]

**Explanation:**

Flight labels: 1 2

Booking 1 reserved: 10 10

Booking 2 reserved: 15

Total seats: 10 25

Hence, answer = [10,25]

**Constraints:**

* 1 <= n <= 2 \* 10^4
* 1 <= bookings.length <= 2 \* 10^4
* bookings[i].length == 3
* 1 <= firsti <= lasti <= n
* 1 <= seatsi <= 10^4

**Attempt 1: 2023-12-16**

**Solution 1: Line Sweep (10 min)**

class Solution {

    public int[] corpFlightBookings(int[][] bookings, int n) {

        int[] delta = new int[n];

        Arrays.sort(bookings, (a, b) -> a[0] - b[0]);

        for(int i = 0; i < bookings.length; i++) {

            // '-1' to mapping 1-indexed 'start' flight number to 0-indexed array

            int start = bookings[i][0] - 1;

            // '+1' to make sure decrease after 'end' flight number only

            // '-1' to mapping 1-indexed 'end' flight number to 0-indexed array

            int end = bookings[i][1] + 1 - 1;

            delta[start] += bookings[i][2];

            // 'end' flight number edge case, when handling the last concatenate

            // flight we don't need to decrease

            // e.g flights = [[1,2,10],[2,3,20],[2,5,25]], n = 5, check out [2,5,25]

            if(end < n) {

                delta[end] -= bookings[i][2];

            }

        }

        int[] result = new int[n];

        int count = 0;

        for(int i = 0; i < delta.length; i++) {

            count += delta[i];

            result[i] = count;

        }

        return result;

    }

}

Time Complexity: O(N)

Space Complexity: O(N)