Range Addition

Assume you have an array of length ***n*** initialized with all **0**'s and are given ***k*** update operations.

Each operation is represented as a triplet: **[startIndex, endIndex, inc]** which increments each element of subarray **A[startIndex ... endIndex]** (startIndex and endIndex inclusive) with **inc**.

Return the modified array after all ***k*** operations were executed.

Example:

```
Given:

    length = 5,
    updates = [
        [1,  3,  2],
        [2,  4,  3],
        [0,  2, -2]
    ]

Output:

    [-2, 0, 3, 5, 3]
```

Explanation:

```
Initial state:
[ 0, 0, 0, 0, 0 ]

After applying operation [1, 3, 2]:
[ 0, 2, 2, 2, 0 ]

After applying operation [2, 4, 3]:
[ 0, 2, 5, 5, 3 ]

After applying operation [0, 2, -2]:
[-2, 0, 3, 5, 3 ]
```

1. Thinking of using advanced data structures? You are thinking it too complicated.
2. For each update operation, do you really need to update all elements between i and j?
3. Update only the first and end element is sufficient.
4. The optimal time complexity is O(***k*** + ***n***) and uses O(1) extra space.

Credits:

Special thanks to @vinod23 for adding this problem and creating all test cases.

Solution 1

From the hint, we only need to update first and end element, so we update the startIndex with inc, then update endIndex + 1 with -inc. Using the example in the problem, We get vector nums = {-2, 2, 3, 2, -2, -3}, then we compute range sum (Range Sum Query - Immutable), that is the final result = {-2, 0, 3, 5, 3}.

```
class Solution {
public:
    vector<int> getModifiedArray(int length, vector<vector<int>>& updates) {
        vector<int> res, nums(length + 1, 0);
        for (int i = 0; i < updates.size(); ++i) {
            nums[updates[i][0]] += updates[i][2];
            nums[updates[i][1] + 1] -= updates[i][2];
        }
        int sum = 0;
        for (int i = 0; i < length; ++i) {
            sum += nums[i];
            res.push_back(sum);
        }
        return res;
    }
};
```

written by grandyang original link here

Solution 2

```
public class Solution {
    public int[] getModifiedArray(int length, int[][] updates) {
        int[] nums = new int[length];
        for (int[] update : updates) {
            nums[update[1]] += update[2];
            if (update[0] > 0) nums[update[0] - 1] -= update[2];
        }
        for (int i = length - 2; i >= 0; i--) {
            nums[i] = nums[i + 1] + nums[i];
        }
        return nums;
    }
}
```

written by xsunfeng original link here

Solution 3

```java
public int[] getModifiedArray(int length, int[][] updates) {
    int[] ans = new int[length];
    // Iterate updates and use answer array to store two-end info.
    for (int[] update : updates) {
        int l_pos = update[0];
        int r_pos = update[1];
        ans[l_pos] += update[2];
        // Keep update in the range by removing the increment after.
        if (r_pos < length - 1) {
            // Ignore the rightmost update.
            ans[r_pos + 1] -= update[2];
        }
    }
    int sum = 0;
    for (int i = 0; i < length; i++) {
        sum += ans[i];
        ans[i] = sum;
    }
    return ans;
}
```

written by McLovin_Feng original link here