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.

From Leetcoder.