370. Range Addition

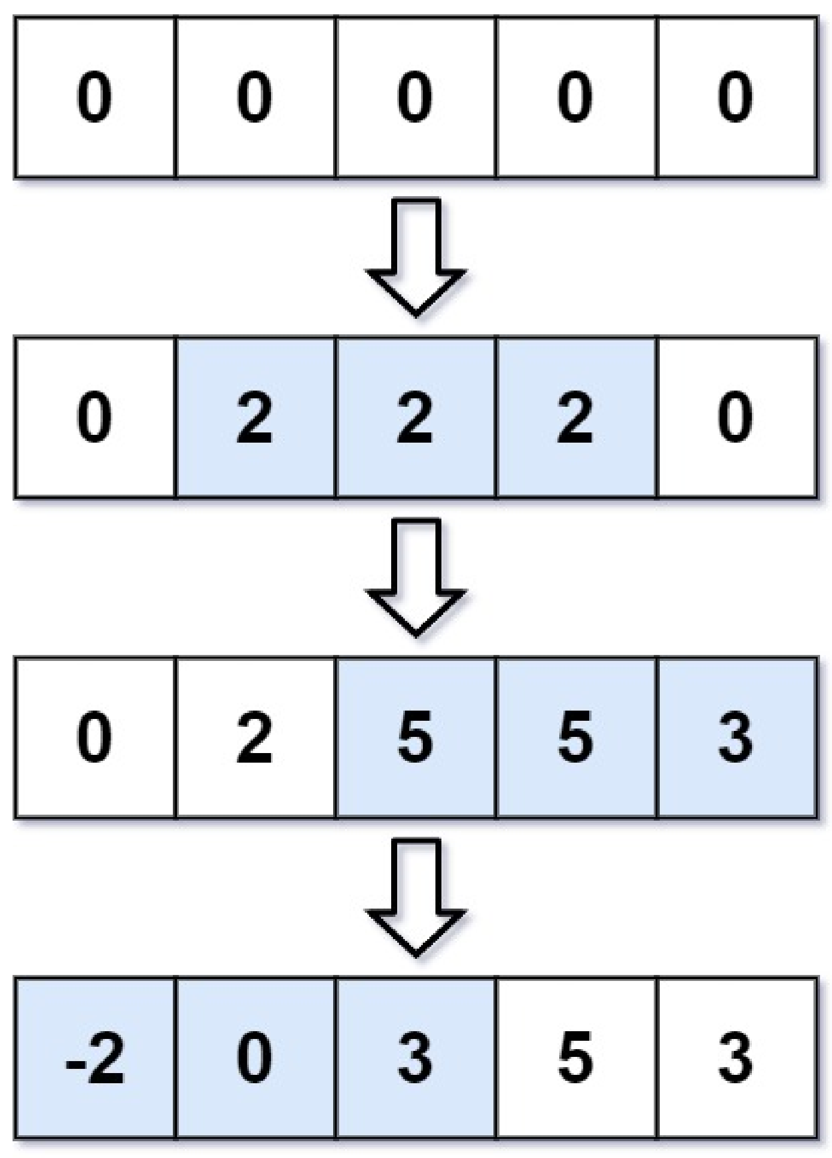
Medium 953 42 Add to List Share

You are given an integer `length` and an array `updates` where `updates[i] = [startIdx_i, endIdx_i, inc_i]`.

You have an array `arr` of length `length` with all zeros, and you have some operation to apply on `arr`. In the i^{th} operation, you should increment all the elements `arr[startIdx_i], arr[startIdx_i + 1], ..., arr[endIdx_i]` by `inc_i`.

Return `arr` *after applying all the* `updates`.

Example 1:



```
Input: length = 5, updates = [[1,3,2],[2,4,3],[0,2,-2]]
Output: [-2,0,3,5,3]
```

Example 2:

```
Input: length = 10, updates = [[2,4,6],[5,6,8],[1,9,-4]]
Output: [0,-4,2,2,2,4,4,-4,-4,-4]
```

Constraints:

- $1 <= length <= 10^5$
- $0 <= updates.length <= 10^4$
- $0 <= startIdx_i <= endIdx_i < length$
- $-1000 <= inc_i <= 1000$

Accepted 47,884 | Submissions 71,600

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months   6 months ~ 1 year   1 year ~ 2 years

Amazon | 22   Citadel | 13   VMware | 2

Related Topics

Array   Prefix Sum

Similar Questions

Range Addition II   Easy

Hide Hint 1

Thinking of using advanced data structures? You are thinking it too complicated.

Hide Hint 2

For each update operation, do you really need to update all elements between i and j?

Hide Hint 3

Update only the first and end element is sufficient.

Hide Hint 4

The optimal time complexity is O(*k* + *n*) and uses O(1) extra space.

C++   Autocomplete

```cpp
vector<int> getModifiedArray(int length, vector<vector<int> > updates)
{
    vector<int> result(length, 0);

    for (auto& tuple : updates) {
        int start = tuple[0], end = tuple[1], val = tuple[2];

        result[start] += val;
        if (end < length - 1)
            result[end + 1] -= val;
    }

    // partial_sum applies the following operation (by default) for the parameters {x[0], x[n], y[0]}:
    // y[0] = x[0]
    // y[1] = y[0] + x[1]
    // y[2] = y[1] + x[2]
    // ...  ...  ...
    // y[n] = y[n-1] + x[n]

    partial_sum(result.begin(), result.end(), result.begin());

    return result;
}


vector<int> getModifiedArray(int length, vector<vector<int> > updates)
{
    vector<int> result(length, 0);

    for (auto& tuple : updates) {
        int start = tuple[0], end = tuple[1], val = tuple[2];

        for (int i = start; i <= end; i++) {
            result[i] += val;
        }
    }

    return result;
}
```