	2	3	−4	7	6	**Difference**
array :	0	2	5	1	8	14
prefix :	2	2+3=5	5−4=1	1+7=8	8+6=14	

আমরা difference জানা থাকায় Array restore করতে পেরেছি।

Range update using naive approach

array : 2 5 1 8 14

queries :			
	1	4	2
	2	5	6
	1	5	3

(from, to, value)

* 1-based index

array :	2	5	1	8	14
	+2	+2	+2	+2	
		+6	+6	+6	+6
	+3	+3	+3	+3	+3
updated array :	7	16	12	19	23

* naive

Complexity : $O(n \times q) \rightarrow O(n^2)$

Difference array for efficient range updates

Task$_1$: 1-4 Range এ 2 যোগ করা।

array :	0	2	5	1	8	14	
	0	+2	0	0	0	-2	0
difference :	0	2	2	2	2	0	0
pref :	. . .						

→ একটা cell বেশি নিব (এর effect নাই)

Task$_2$:

1 4 2
2 5 6
2 4 3

→ l, r, value

original array:	0	2	5	1	8	14	
	0	+2	+(6+3)	0	0	(-2-3)	-6
difference array :	0	2	11	0	0	-5	-6
prefix :	0	2	11	11	11	6	0
(arr + pref) :	0	4	16	12	19	20	□ → কাজ নাই

→ একটা cell বেশি নিব

Observation : O(1) এ আমরা সবগুলো value কে update করে দিচ্ছি। সবার শেষে O(n) এ array আপডেট করে দিব।

$O(nr) \longrightarrow O(n)$

//optimized

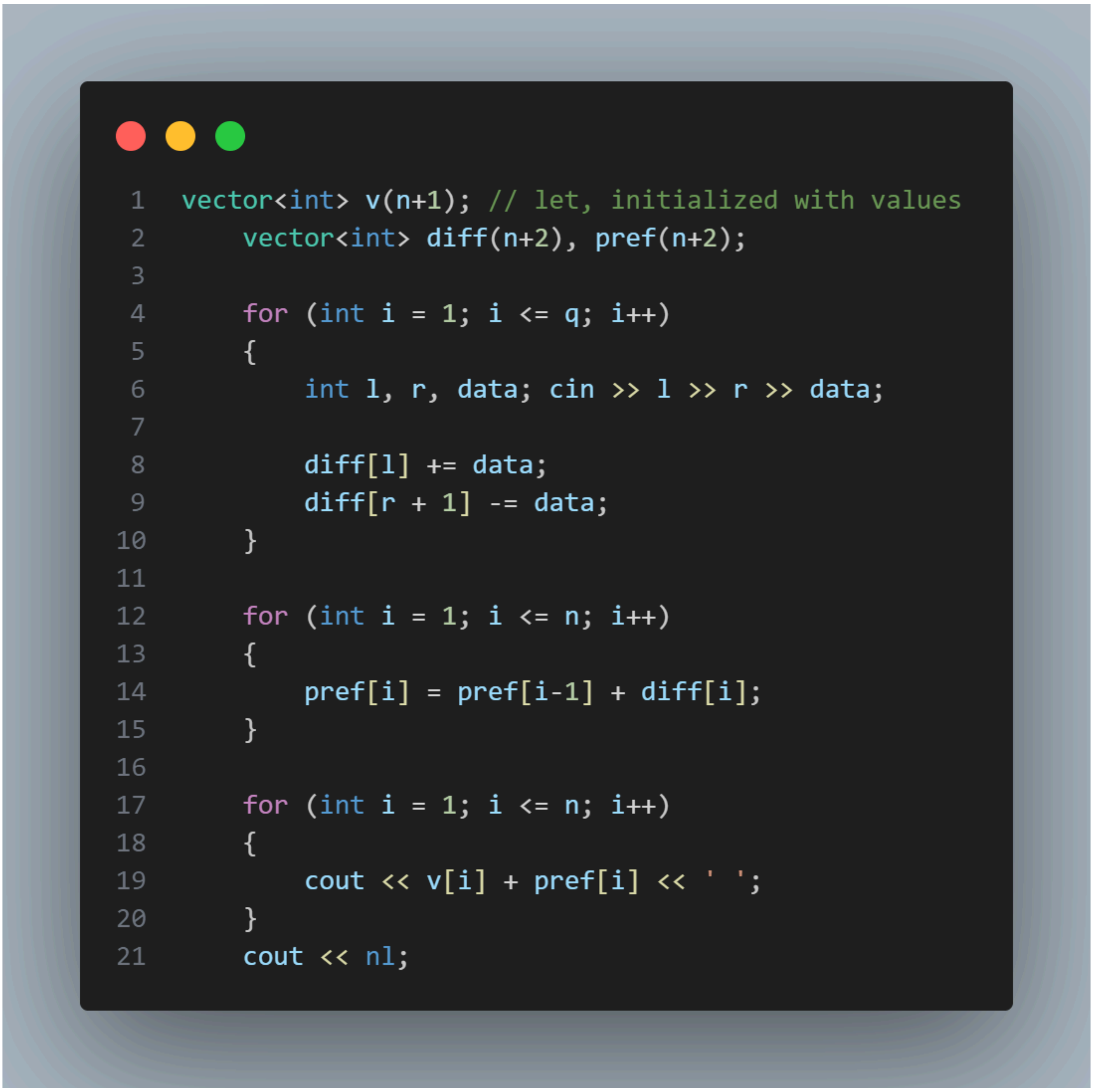
summary

* মূল অ্যারে (1-based index)

* difference array → diff (n+2)

# 2টা index বেশি নিব।

* prefix sum → pref (n+2)



```cpp
vector<int> v(n+1); // let, initialized with values
    vector<int> diff(n+2), pref(n+2);

    for (int i = 1; i <= q; i++)
    {
        int l, r, data; cin >> l >> r >> data;

        diff[l] += data;
        diff[r + 1] -= data;
    }

    for (int i = 1; i <= n; i++)
    {
        pref[i] = pref[i-1] + diff[i];
    }

    for (int i = 1; i <= n; i++)
    {
        cout << v[i] + pref[i] << ' ';
    }
    cout << nl;
```