307. Range Sum Query - Mutable

Description

Given an integer array `nums`, handle multiple queries of the following types:

1. **Update** the value of an element in `nums`.
2. Calculate the **sum** of the elements of `nums` between indices `left` and `right` **inclusive** where `left <= right`.

Implement the `NumArray` class:

- `NumArray(int[] nums)` Initializes the object with the integer array `nums`.
- `void update(int index, int val)` **Updates** the value of `nums[index]` to be `val`.
- `int sumRange(int left, int right)` Returns the **sum** of the elements of `nums` between indices `left` and `right` **inclusive** (i.e. `nums[left] + nums[left + 1] + ... + nums[right]`).

Example 1:

```
Input
["NumArray", "sumRange", "update", "sumRange"]
[[[1, 3, 5]], [0, 2], [1, 2], [0, 2]]
Output
[null, 9, null, 8]

Explanation
NumArray numArray = new NumArray([1, 3, 5]);
numArray.sumRange(0, 2); // return 1 + 3 + 5 = 9
numArray.update(1, 2);   // nums = [1, 2, 5]
numArray.sumRange(0, 2); // return 1 + 2 + 5 = 8
```

Constraints:

- $1 <= \text{nums.length} <= 3 * 10^4$
- `-100 <= nums[i] <= 100`
- `0 <= index < nums.length`
- `-100 <= val <= 100`
- `0 <= left <= right < nums.length`
- At most $3 * 10^4$ calls will be made to `update` and `sumRange`.