Range Sum Query - Immutable

Given an integer array *nums*, find the sum of the elements between indices *i* and *j* (*i* ≤ *j*), inclusive.

Example:

```
Given nums = [-2, 0, 3, -5, 2, -1]

sumRange(0, 2) -> 1
sumRange(2, 5) -> -1
sumRange(0, 5) -> -3
```

Note:

1. You may assume that the array does not change.
2. There are many calls to *sumRange* function.

Solution 1

public class NumArray {

```
int[] nums;

public NumArray(int[] nums) {
    for(int i = 1; i < nums.length; i++)
        nums[i] += nums[i - 1];

    this.nums = nums;
}

public int sumRange(int i, int j) {
    if(i == 0)
        return nums[j];

    return nums[j] - nums[i - 1];
}
```

}

written by arthur13 original link here

Solution 2

```
class NumArray {
public:
    NumArray(vector<int> &nums) : psum(nums.size()+1, 0) {
        partial_sum( nums.begin(), nums.end(), psum.begin()+1);
    }

    int sumRange(int i, int j) {
        return psum[j+1] - psum[i];
    }
private:
    vector<int> psum;
};
```

written by rantos22 original link here

Solution 3

The idea is fairly straightforward: create an array `accu` that stores the accumulated sum for `nums` such that `accu[i] = nums[0] + ... + nums[i - 1]` in the initializer of `NumArray`. Then just return `accu[j + 1] - accu[i]` in `sumRange`. You may try the example in the problem statement to convince yourself of this idea.

The code is as follows.

C++

```
class NumArray {
public:
    NumArray(vector<int> &nums) {
        accu.push_back(0);
        for (int num : nums)
            accu.push_back(accu.back() + num);
    }

    int sumRange(int i, int j) {
        return accu[j + 1] - accu[i];
    }
private:
    vector<int> accu;
};


// Your NumArray object will be instantiated and called as such:
// NumArray numArray(nums);
// numArray.sumRange(0, 1);
// numArray.sumRange(1, 2);
```

Python

```python
class NumArray(object):
    def __init__(self, nums):
        """
        initialize your data structure here.
        :type nums: List[int]
        """
        self.accu = [0]
        for num in nums:
            self.accu += self.accu[-1] + num,

    def sumRange(self, i, j):
        """
        sum of elements nums[i..j], inclusive.
        :type i: int
        :type j: int
        :rtype: int
        """
        return self.accu[j + 1] - self.accu[i]


# Your NumArray object will be instantiated and called as such:
# numArray = NumArray(nums)
# numArray.sumRange(0, 1)
# numArray.sumRange(1, 2)
```

written by jianchao.li.fighter original link here

From Leetcoder.