303 Range Sum Query - Immutable

2018年4月9日 13:40

Question:

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.

来自 <https://leetcode.com/problems/range-sum-query-immutable/description/>

给定一个数组，求出数组从索引 *i* 到 *j* （*i* ≤ *j*）范围内元素的总和，包含 *i*, *j* 两点。

例如：

给定nums = [-2, 0, 3, -5, 2, -1]，求和函数为sumRange()
sumRange(0, 2) -> 1
sumRange(2, 5) -> -1
sumRange(0, 5) -> -3

注意：

1. 你可以假设数组不可变。
2. 会多次调用 *sumRange* 方法。

Solution for Python3:

```python
class NumArray:

    def __init__(self, nums):
        """
        :type nums: List[int]
        """
        self.sum = [0] * (len(nums) + 1)
        for i in range(len(nums)):
            self.sum[i + 1] = self.sum[i] + nums[i]


    def sumRange(self, i, j):
        """
        :type i: int
        :type j: int
        :rtype: int
```

```
        """
        return self.sum[j + 1] - self.sum[i]


# Your NumArray object will be instantiated and called as such:
# obj = NumArray(nums)
# param_1 = obj.sumRange(i,j)
```

Solution for C++:

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

    int sumRange(int i, int j) {
        return sum[j + 1] - sum[i];
    }
};

/**
 * Your NumArray object will be instantiated and called as such:
 * NumArray obj = new NumArray(nums);
 * int param_1 = obj.sumRange(i,j);
 */
```