区域和检索 - 数组不可变

给定一个整数数组 nums，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。

实现 NumArray 类：

- NumArray(int[] nums) 使用数组 nums 初始化对象
- int sumRange(int i, int j) 返回数组 nums 从索引 i 到 j（i ≤ j）范围内元素的总和，包含 i、j 两点（也就是 sum(nums[i], nums[i + 1], ... , nums[j])）

示例：

输入：

```
["NumArray", "sumRange", "sumRange", "sumRange"]

[[[-2, 0, 3, -5, 2, -1]], [0, 2], [2, 5], [0, 5]]
```

输出：

```
[null, 1, -1, -3]
```

解释：

```
NumArray numArray = new NumArray([-2, 0, 3, -5, 2, -1]);

numArray.sumRange(0, 2); // return 1 ((-2) + 0 + 3)

numArray.sumRange(2, 5); // return -1 (3 + (-5) + 2 + (-1))

numArray.sumRange(0, 5); // return -3 ((-2) + 0 + 3 + (-5) + 2 + (-1))
```

提示：

- 0 <= nums.length <= 10^4
- -10^5 <= nums[i] <= 10^5
- 0 <= i <= j < nums.length
- 最多调用 10^4 次 sumRange 方法

```
class NumArray {
public:
    vector<int> sums;
    NumArray(vector<int>& nums) {
        //sums[i]表示前 i-1 个元素的和
        sums.resize(nums.size()+1);
        for(int i=0;i<nums.size();i++)
        {
            sums[i+1]=sums[i]+nums[i];//前 i 个元素的和等于前 i-1 个元素和加上当前第 i 个元素
        }
    }

    int sumRange(int left, int right) {
    //sumRange(i,j)，则需要计算数组 nums 在下标 j 和下标 i-1 的前缀和
        return sums[right+1]-sums[left];
    }
};

/**
 * Your NumArray object will be instantiated and called as such:
 * NumArray* obj = new NumArray(nums);
 * int param_1 = obj->sumRange(left,right);
 */
```