Assignment Solutions | Prefix sum | Week 12

1. Given an integer array `nums`, handle multiple queries of the following type:

   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`.
- `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]`). [Leetcode 303]

  Example 1:

  Input

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

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

  Output

  [null, 1, -1, -3]

  Explanation

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

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

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

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

Solution :

```
class NumArray {
public:
    vector<int>pre;
    NumArray(vector<int>& nums) {
        pre = vector<int>(nums.size());
        pre[0] = nums[0];
        int n = nums.size();
        for(int i=1;i<n;i++)pre[i] = pre[i-1] + nums[i];
    }

    int sumRange(int left, int right) {
        if(left == 0)return pre[right];
        return pre[right] - pre[left - 1];
    }
};
```

2. Given an array of integers `nums`, calculate the **pivot index** of this array.

   The **pivot index** is the index where the sum of all the numbers **strictly** to the left of the index is equal to the sum of all the numbers **strictly** to the index's right.

   If the index is on the left edge of the array, then the left sum is `0` because there are no elements to the left. This also applies to the right edge of the array.

   Return *the **leftmost pivot index***. If no such index exists, return `-1`.
   [Leetcode 724]

   Example 1:

   Input: nums = [1,7,3,6,5,6]

   Output: 3

   Explanation:

   The pivot index is 3.

   Left sum = nums[0] + nums[1] + nums[2] = 1 + 7 + 3 = 11

   Right sum = nums[4] + nums[5] = 5 + 6 = 11

   Example 2:

   Input: nums = [1,2,3]

   Output: -1

   Explanation:

   There is no index that satisfies the conditions in the problem statement.

   Example 3:

   Input: nums = [2,1,-1]

   Output: 0

   Explanation:

   The pivot index is 0.

   Left sum = 0 (no elements to the left of index 0)

   Right sum = nums[1] + nums[2] = 1 + -1 = 0

Solution :

```
class Solution {
public:
    int pivotIndex(vector<int>& a) {
        int n = a.size();
        int leftsum = 0 , rightsum  = 0;
        for(auto x:a)rightsum += x;
        for(int i=0;i<n;i++){
            rightsum = rightsum - a[i];
            if(leftsum == rightsum)return i;
            leftsum += a[i];
        }
        return -1;
    }
};
```

3. We define the **conversion array** `conver` of an array `arr` as follows:

- `conver[i] = arr[i] + max(arr[0..i])` where `max(arr[0..i])` is the maximum value of `arr[j]` over `0 <= j <= i`.

  We also define the **score** of an array `arr` as the sum of the values of the conversion array of `arr`.

  Given a **0-indexed** integer array `nums` of length `n`, return *an array* `ans` *of length* `n` *where* `ans[i]` *is the score of the prefix* `nums[0..i]`. [Leetcode 2640]

  Example 1:

  Input: nums = [2,3,7,5,10]

  Output: [4,10,24,36,56]

  Explanation:

  For the prefix [2], the conversion array is [4] hence the score is 4

  For the prefix [2, 3], the conversion array is [4, 6] hence the score is 10

  For the prefix [2, 3, 7], the conversion array is [4, 6, 14] hence the score is 24

  For the prefix [2, 3, 7, 5], the conversion array is [4, 6, 14, 12] hence the score is 36

  For the prefix [2, 3, 7, 5, 10], the conversion array is [4, 6, 14, 12, 20] hence the score is 56

  Example 2:

  Input: nums = [1,1,2,4,8,16]

  Output: [2,4,8,16,32,64]

  Explanation:

  For the prefix [1], the conversion array is [2] hence the score is 2

  For the prefix [1, 1], the conversion array is [2, 2] hence the score is 4

  For the prefix [1, 1, 2], the conversion array is [2, 2, 4] hence the score is 8

  For the prefix [1, 1, 2, 4], the conversion array is [2, 2, 4, 8] hence the score is 16

  For the prefix [1, 1, 2, 4, 8], the conversion array is [2, 2, 4, 8, 16] hence the score is 32

  For the prefix [1, 1, 2, 4, 8, 16], the conversion array is [2, 2, 4, 8, 16, 32] hence the score is 64

Solution :

```
class Solution {
public:
    vector<long long> findPrefixScore(vector<int>& a) {
        int n = a.size();
        vector<long long int>res(n,0);
        res[0] = 2*a[0];
        int maxi = a[0];
        // maxi = max(maxi , a[0]);

        for(int i=1;i<n;i++){
            maxi = max(maxi , a[i]);

            res[i] = a[i] + maxi + res[i-1];
        }
        return res;
    }
};
```

4. There are `n` flights that are labeled from `1` to `n`.

   You are given an array of flight bookings `bookings`, where `bookings[i] = [firsti, lasti, seatsi]` represents a booking for flights `firsti` through `lasti` (**inclusive**) with `seatsi` seats reserved for **each flight** in the range.

   Return *an array* `answer` *of length* `n`*, where* `answer[i]` *is the total number of seats reserved for flight* `i`. [Leetcode 1109]

   Example 1:

   Input: bookings = [[1,2,10],[2,3,20],[2,5,25]], n = 5

   Output: [10,55,45,25,25]

   Explanation:

   Flight labels: 1   2   3   4   5

   Booking 1 reserved: 10 10

   Booking 2 reserved: 20 20

   Booking 3 reserved: 25 25 25 25

   Total seats: 10 55 45 25 25

   Hence, answer = [10,55,45,25,25]

   Example 2:

   Input: bookings = [[1,2,10],[2,2,15]], n = 2

   Output: [10,25]

   Explanation:

   Flight labels: 1   2

   Booking 1 reserved: 10 10

   Booking 2 reserved: 15

Total seats: 10 25

Hence, answer = [10,25]

Solution :

```
class Solution {
public:
    vector<int> corpFlightBookings(vector<vector<int>>& a, int n) {
        vector<int>res(n,0);

        for(int i=0;i<a.size();i++){
            res[a[i][0] - 1] += a[i][2];
            if(a[i][1] < n)res[a[i][1]] -= a[i][2];
        }

        for(int i=1;i<n;i++){
            res[i] += res[i-1];
        }
        return res;
    }
};
```

©