Soma de Prefixos
Professor Wladimir

1480. Running Sum of 1d Array

url : https://leetcode.com/problems/running-sum-of-1d-array/description/
A ideia central do algoritmo baseia-se nas seguintes observações:

$$\text{runningSum}[0] = \text{nums}[0] \quad \text{runningSum}[i] = \text{runningSum}[i-1] + \text{nums}[i] \quad \text{para } i > 0, \tag{1}$$

onde:

- nums é o vetor de entrada.
- runningSum é o vetor de saída, que armazena as somas acumuladas.

Utilizando essa relação de recorrência, o vetor runningSum pode ser construído de forma eficiente com uma complexidade linear, $\mathcal{O}(n)$, onde n é o número de elementos em nums. Isso é alcançado calculando cada elemento de runningSum em uma única passagem pelo vetor de entrada.

```cpp
vector<int> runningSum(vector<int>& nums) {
        vector <int> res;
        res.resize(nums.size());
        res[0] = nums[0];
        for(int i = 1; i < (int)nums.size(); i++){
            res[i] = res[i-1] + nums[i];
        }
        return res;
    }
```

303. Range Sum Query - Immutable

url: https://leetcode.com/problems/range-sum-query-immutable/description/
Seja $nums$ um vetor de entrada. Consider que

$$\text{acc}[j] = \sum_{i=0}^{j} nums[i] \tag{2}$$

$$\tag{3}$$

Com o vetor de soma acumulada, podemos calcular o rangeSum da seguinte maneira:

$$\text{rangeSum}(0, j) = \text{acc}[j] \tag{4}$$
$$\text{rangeSum}(i, j) = \text{acc}[j] - \text{acc}[i-1] \tag{5}$$
$$\tag{6}$$

```cpp
class NumArray {
public:
    vector <int> acc;
    NumArray(vector<int>& nums) {
        for(int i = 0; i < nums.size(); i++){
            if(i==0) acc.push_back(nums[i]);
            else acc.push_back( acc.back() + nums[i] );
        }


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

724. Find Pivot Index

url: https://leetcode.com/problems/find-pivot-index/description/
Note que

$$\text{left}[i] = \sum_{j=0}^{i-1} nums[i]$$
$$\text{right}[i] = \sum_{j=i+1}^{n} nums[i]$$

Para $i = 0$, temos que:

$$left[0] = 0$$
$$right[0] = total - nums[0]$$

Além disso, temos que

$$left[i+1] = left[i] + nums[i]$$
$$right[i+1] = right[i] - nums[i+1]$$

Essa idéias podem ser combinadas no seguinte programa:

```cpp
int pivotIndex(vector<int>& nums) {
        int right = 0;
        for(int x : nums) right += x;
        int left = 0;

        for(int i = 0; i < nums.size(); i++){
            if( left == right - nums[i]){
                return i;
            }
            left += nums[i];
            right -= nums[i];
        }
        return -1;

    }
```

Sugestões de Leitura

- USACO Guide https://usaco.guide/silver/prefix-sums?lang=cpp
- Trilha Prefiz Sum Codechef https://www.codechef.com/pre/acticprefix-sums

Exercícios Sugeridos

- 238. Product of Array Except Self
  https://leetcode.com/problems/product-of-array-except-self/description/
- 304. Range Sum Query 2D - Immutable
  https://leetcode.com/problems/range-sum-query-2d-immutable/description/