DSA SERIES

- Learn Coding

Topic to be Covered today

Prefix Sum

LETS START TODAY'S LECTURE

Prefix Sum

A **prefix sum** is an array (or sometimes a formula) that stores the cumulative sum of a given array up to each index.

- **Each element of the prefix sum array tells you the sum of all elements before (and including) that index.**

Given an array: arr = [2, 4, 1, 3, 6]

We want to compute the sum from index i to j (inclusive) multiple times quickly.

```
vector<int> prefix(arr.size());
prefix[0] = arr[0];

for (int i = 1; i < arr.size(); i++) {
    prefix[i] = prefix[i - 1] + arr[i];
}
```

Problem : 303

Range Sum Query -Immutable

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

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

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

```cpp
class NumArray {
public:
    vector<int> prefix;
    NumArray(vector<int>& nums) {
        prefix.push_back(nums[0]);

        int n = nums.size();

        for(int i =1;i<n;i++){
            prefix.push_back(nums[i]+prefix[i-1]);
        }

    }

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

Problem : 2574

Left and Right Sum Differences

You are given a **0-indexed** integer array nums of size n.

Define two arrays leftSum and rightSum where:

- leftSum[i] is the sum of elements to the left of the index i in the array nums. If there is no such element, leftSum[i] = 0.
- rightSum[i] is the sum of elements to the right of the index i in the array nums. If there is no such element, rightSum[i] = 0.

Return an integer array answer of size n where answer[i] = |leftSum[i] - rightSum[i]|.

```
class Solution {
public:
vector<int> left;
vector<int> right;
    void leftSum(vector<int> &nums){
        int n = nums.size();
        left = vector<int> (n,0);
        for(int i=1;i<n;i++){
            left[i]=left[i-1]+nums[i-1];
        }
    }
      void rightSum(vector<int> &nums){
        int n = nums.size();
        right = vector<int> (n,0);


        for(int i=n-2;i>=0;i--){
           right[i]=right[i+1]+nums[i+1];
        }

    }
vector<int> leftRightDifference(vector<int>&
nums) {
        leftSum(nums);
        rightSum(nums);

        vector<int> ans;

        for(int i = 0;i<nums.size();i++){
            ans.push_back(abs(left[i]-
right[i]));
        }

        return ans;
    }
};
```

Problem : 2559

Count Vowel Strings In Ranges

You are given a **0-indexed** array of strings words and a 2D array of integers queries.

Each query queries[i] = [l_i, r_i] asks us to find the number of strings present at the indices ranging from l_i to r_i (both **inclusive**) of words that start and end with a vowel.

Return *an array* ans *of size* queries.length*, where* ans[i] *is the answer to the* ith *query*.

Note that the vowel letters are 'a', 'e', 'i', 'o', and 'u'.

```
class Solution {
public:
    bool isVowel(char &ch){
        if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'){
            return true;
        }
        return false;
    }
    vector<int> vowelStrings(vector<string>& words,
                             vector<vector<int>>& queries) {
        int Q = queries.size();
        int N = words.size();

        vector<int> cumSum(N);
        vector<int> ans(Q);
        int sum = 0;
        for (int i = 0; i < N; i++) {
            if (isVowel(words[i][0]) && isVowel(words[i].back())) {
                sum++;
            }
            cumSum[i] = sum;
        }
```

```cpp
        for (int i = 0; i < Q; i++) {
            int l = queries[i][0];
            int r = queries[i][1];

            if (l == 0) {
                ans[i] = cumSum[r];
            } else {
                ans[i] = cumSum[r] - cumSum[l - 1];
            }
        }

        return ans;
    }
};
```

Learn coding

THANK YOU