Premium

Description | Editorial | Solutions (1.3K) | Submissions

2574. Left and Right Sum Differences

Hint

Easy 678 50

Companies

Given a **0-indexed** integer array `nums`, find a **0-indexed** integer array `answer` where:

- `answer.length == nums.length`.
- `answer[i] = |leftSum[i] - rightSum[i]|`.

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 *the array* `answer`.

Example 1:

```
Input: nums = [10,4,8,3]
Output: [15,1,11,22]
Explanation: The array leftSum is [0,10,14,22] and the array rightSum is
[15,11,3,0].
                                                    0|] = [15,1,11,22].
```

Java | Auto

```java
class Solution {
    public int[] leftRightDifference(int[] nums) {
        int leftSum = 0, rightSum = 0;
        for(int n : nums) rightSum += n;
        for(int i = 0; i < nums.length; i ++) {
            leftSum += nums[i];
            rightSum -= nums[i];
            nums[i] = Math.abs((leftSum - nums[i]) - rightSum);
        }
        return nums;
    }
}
```

Console

Run Submit

https://leetcode.com/problems/left-and-right-sum-differences/description

2640. Find the Score of All Prefixes of an Array

Hint

Medium 194 11

Companies

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]`.

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]
```


Java • Auto

```java
class Solution {
    public long[] findPrefixScore(int[] nums) {
        int n = nums.length;
        long[] score = new long[n + 1];
        for (int i = 0, max = 0; i < n; ++i) {
            max = Math.max(max, nums[i]);
            score[i + 1] += score[i] + max + nums[i];
        }
        return Arrays.copyOfRange(score, 1, n + 1);
    }
}
```

Console

Run Submit

Description Editorial Solutions (574) Submissions

1685. Sum of Absolute Differences in a Sorted Array

Hint

Medium 1.1K 32

Companies

You are given an integer array `nums` sorted in **non-decreasing** order.

Build and return *an integer array* `result` *with the same length as* `nums` *such that* `result[i]` *is equal to the* ***summation of absolute differences*** *between* `nums[i]` *and all the other elements in the array.*

In other words, `result[i]` is equal to `sum(|nums[i]-nums[j]|)` where `0 <= j < nums.length` and `j != i` (**0-indexed**).

Example 1:

```
Input: nums = [2,3,5]
Output: [4,3,5]
Explanation: Assuming the arrays are 0-indexed, then
result[0] = |2-2| + |2-3| + |2-5| = 0 + 1 + 3 = 4,
result[1] = |3-2| + |3-3| + |3-5| = 1 + 0 + 2 = 3,
result[2] = |5-2| + |5-3| + |5-5| = 3 + 2 + 0 = 5.
```

Example 2:

```
Input: nums = [1,4,6,8,10]
```

https://leetcode.com/problems/sum-of-absolute-differences-in-a-sorted-array/description

Java Auto

```java
class Solution {
    public int[] getSumAbsoluteDifferences(int[] nums) {
        int n = nums.length;
        int[] res = new int[n];
        int sumBelow = 0;
        int sumTotal = Arrays.stream(nums).sum();

        for (int i = 0; i < n; i++) {
            int num = nums[i];
            sumTotal -= num;
            res[i] = sumTotal - (n - i - 1) * num + i * num - sumBelow;
            sumBelow += num;
        }
        return res;
    }
}
```

Console Run Submit

Description Editorial Solutions (938) Submissions

2485. Find the Pivot Integer

Hint

Easy 509 8

Companies

Given a positive integer `n`, find the **pivot integer** `x` such that:

- The sum of all elements between `1` and `x` inclusively equals the sum of all elements between `x` and `n` inclusively.

Return *the pivot integer* `x`. If no such integer exists, return `-1`. It is guaranteed that there will be at most one pivot index for the given input.

Example 1:

```
Input: n = 8
Output: 6
Explanation: 6 is the pivot integer since: 1 + 2 + 3 + 4 + 5 + 6 = 6 + 7 + 8 = 21.
```

Example 2:

```
Input: n = 1
Output: 1
Explanation: 1 is the pivot integer since: 1 = 1.
```

https://leetcode.com/problems/find-the-pivot-integer/description

Java Auto

```java
class Solution {
    public int pivotInteger(int n) {
        int a = (int)Math.sqrt((n*n + n)/2);
        if(2*a*a==n*n+n)
        return a;

        return -1;
    }
}
```

Console Run Submit

Problem List

Premium

Description | Editorial | Solutions (1.1K) | Submissions

1991. Find the Middle Index in Array

Hint

Easy 1.2K 49

Companies

Given a **0-indexed** integer array `nums`, find the **leftmost** `middleIndex` (i.e., the smallest amongst all the possible ones).

A `middleIndex` is an index where `nums[0] + nums[1] + ... + nums[middleIndex-1] == nums[middleIndex+1] + nums[middleIndex+2] + ... + nums[nums.length-1]`.

If `middleIndex == 0`, the left side sum is considered to be `0`. Similarly, if `middleIndex == nums.length - 1`, the right side sum is considered to be `0`.

Return *the **leftmost** `middleIndex` that satisfies the condition, or `-1` if there is no such index.*

Example 1:

```
Input: nums = [2,3,-1,8,4]
Output: 3
Explanation: The sum of the numbers before index 3 is: 2 + 3 + -1 = 4
The sum of the numbers after index 3 is: 4 = 4
```

Example 2:

Java | Auto

```java
class Solution {
    public int findMiddleIndex(int[] nums) {
        int sum = 0;
        int temp = 0;
        for(int i = 0 ; i < nums.length ; i++){
            sum += nums[i];
        }
        for(int j=0 ; j < nums.length ; j++){
                sum -= nums[j];
            if(sum == temp)return j;
                temp += nums[j];
        }
        return -1;
    }
}
```

Console

Run | Submit

https://leetcode.com/problems/find-the-middle-index-in-array/description

Problem List

Premium

Description | Editorial | Solutions (3.8K) | Submissions

1732. Find the Highest Altitude

Hint

Easy 2.4K 216

Companies

There is a biker going on a road trip. The road trip consists of `n + 1` points at different altitudes. The biker starts his trip on point `0` with altitude equal `0`.

You are given an integer array `gain` of length `n` where `gain[i]` is the **net gain in altitude** between points `i` and `i + 1` for all (`0 <= i < n`). Return *the* ***highest altitude*** *of a point.*

Example 1:

```
Input: gain = [-5,1,5,0,-7]
Output: 1
Explanation: The altitudes are [0,-5,-4,1,1,-6]. The highest is 1.
```

Example 2:

```
Input: gain = [-4,-3,-2,-1,4,3,2]
Output: 0
Explanation: The altitudes are [0,-4,-7,-9,-10,-6,-3,-1]. The highest is 0.
```

https://leetcode.com/problems/find-the-highest-altitude/description

Java | Auto

```java
class Solution {
    public int largestAltitude(int[] gain) {
        int maxAltitude = 0;
        int currentAltitude = 0;
        for (int i = 0; i < gain.length; i++) {
            currentAltitude += gain[i];
            maxAltitude = Math.max(maxAltitude, currentAltitude);
        }
        return maxAltitude;
    }
}
```

Console

Run Submit

Problem List

Premium

Description | Editorial | Solutions (2K) | Submissions

1588. Sum of All Odd Length Subarrays

Hint

Easy 3.4K 256

Companies

Given an array of positive integers `arr`, return *the sum of all possible* ***odd-length subarrays*** *of* `arr`.

A **subarray** is a contiguous subsequence of the array.

Example 1:

```
Input: arr = [1,4,2,5,3]
Output: 58
Explanation: The odd-length subarrays of arr and their sums are:
[1] = 1
[4] = 4
[2] = 2
[5] = 5
[3] = 3
[1,4,2] = 7
[4,2,5] = 11
[2,5,3] = 10
[1,4,2,5,3] = 15
If we add all these together we get 1 + 4 + 2 + 5 + 3 + 7 + 11 + 10 + 15 = 58
```

Example 2:

Java | Auto

```java
class Solution {
    public int sumOddLengthSubarrays(int[] arr) {
        int result = 0;
        int n = arr.length;

        for (int i = 0; i < n; i++) {
            int end = i + 1;
            int start = n - i;
            int totalSubarrays = end * start;
            int oddSubarrays = totalSubarrays / 2;
            if (totalSubarrays % 2 == 1) {
                oddSubarrays++;
            }
            result += oddSubarrays * arr[i];
        }

        return result;
    }
}
```

Console

Run | Submit

Problem List

Premium

Description | Editorial | Solutions (9.3K) | Submissions

1480. Running Sum of 1d Array

Hint

Easy 7K 315

Companies

Given an array `nums`. We define a running sum of an array as `runningSum[i] = sum(nums[0]…nums[i])`.

Return the running sum of `nums`.

Example 1:

```
Input: nums = [1,2,3,4]
Output: [1,3,6,10]
Explanation: Running sum is obtained as follows: [1, 1+2, 1+2+3, 1+2+3+4].
```

Example 2:

```
Input: nums = [1,1,1,1,1]
Output: [1,2,3,4,5]
Explanation: Running sum is obtained as follows: [1, 1+1, 1+1+1, 1+1+1+1,
1+1+1+1+1].
```

Example 3:

```
Input: nums = [3,1,2,10,1]
```

Java | Auto

```java
class Solution {
    public int[] runningSum(int[] nums) {
        for(int i=1;i<nums.length;i++)  nums[i]+=nums[i-1];
        return nums;
    }
}
```

Testcase | Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Console

Run Submit