Problem 1685: Sum of Absolute Differences in a Sorted Array

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

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]

Output:

[24,15,13,15,21]

Constraints:

2 <= nums.length <= 10

5

1 <= nums[i] <= nums[i + 1] <= 10

4

Code Snippets

C++:

```
class Solution {
public:
vector<int> getSumAbsoluteDifferences(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int[] getSumAbsoluteDifferences(int[] nums) {

}
}
```

Python3:

```
class Solution:
def getSumAbsoluteDifferences(self, nums: List[int]) -> List[int]:
```

Python:

```
class Solution(object):
def getSumAbsoluteDifferences(self, nums):
"""
:type nums: List[int]
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {number[]} nums
* @return {number[]}
*/
var getSumAbsoluteDifferences = function(nums) {

};
```

TypeScript:

```
function getSumAbsoluteDifferences(nums: number[]): number[] {

};
```

C#:

```
public class Solution {
public int[] GetSumAbsoluteDifferences(int[] nums) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* getSumAbsoluteDifferences(int* nums, int numsSize, int* returnSize){

}
```

Go:

```
func getSumAbsoluteDifferences(nums []int) []int {

}
```

Kotlin:

```
class Solution {
fun getSumAbsoluteDifferences(nums: IntArray): IntArray {

}
}
```

Swift:

```
class Solution {
func getSumAbsoluteDifferences(_ nums: [Int]) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn get_sum_absolute_differences(nums: Vec<i32>) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @return {Integer[]}
def get_sum_absolute_differences(nums)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer[]
*/
function getSumAbsoluteDifferences($nums) {

}
}
```

Scala:

```
object Solution {
def getSumAbsoluteDifferences(nums: Array[Int]): Array[Int] = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Sum of Absolute Differences in a Sorted Array
* Difficulty: Medium
* Tags: array, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
class Solution {
public:
vector<int> getSumAbsoluteDifferences(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Sum of Absolute Differences in a Sorted Array
* Difficulty: Medium
* Tags: array, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int[] getSumAbsoluteDifferences(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Sum of Absolute Differences in a Sorted Array
Difficulty: Medium
Tags: array, math, sort

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def getSumAbsoluteDifferences(self, nums: List[int]) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def getSumAbsoluteDifferences(self, nums):
"""
:type nums: List[int]
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Sum of Absolute Differences in a Sorted Array
* Difficulty: Medium
* Tags: array, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[]} nums
* @return {number[]}
*/
var getSumAbsoluteDifferences = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Sum of Absolute Differences in a Sorted Array
* Difficulty: Medium
* Tags: array, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function getSumAbsoluteDifferences(nums: number[]): number[] {
```

```
};
```

C# Solution:

```
/*
* Problem: Sum of Absolute Differences in a Sorted Array
* Difficulty: Medium
* Tags: array, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int[] GetSumAbsoluteDifferences(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Sum of Absolute Differences in a Sorted Array
* Difficulty: Medium
* Tags: array, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/




/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* getSumAbsoluteDifferences(int* nums, int numsSize, int* returnSize){

}
```

Go Solution:

```
// Problem: Sum of Absolute Differences in a Sorted Array
// Difficulty: Medium
// Tags: array, math, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func getSumAbsoluteDifferences(nums []int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun getSumAbsoluteDifferences(nums: IntArray): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func getSumAbsoluteDifferences(_ nums: [Int]) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Sum of Absolute Differences in a Sorted Array
// Difficulty: Medium
// Tags: array, math, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn get_sum_absolute_differences(nums: Vec<i32>) -> Vec<i32> {
```

```
}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @return {Integer[]}
def get_sum_absolute_differences(nums)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer[]
*/
function getSumAbsoluteDifferences($nums) {

}
}
```

Scala Solution:

```
object Solution {
def getSumAbsoluteDifferences(nums: Array[Int]): Array[Int] = {

}
}
```