Problem 2256: Minimum Average Difference

Problem Information

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

Problem Description

You are given a

0-indexed

integer array

nums

of length

n

.

The

average difference

of the index

i

is the

absolute

difference

between the average of the

first

i + 1

elements of

nums

and the average of the

last

n - i - 1

elements. Both averages should be

rounded down

to the nearest integer.

Return

the index with the

minimum average difference

. If there are multiple such indices, return the

smallest

one.

Note:

The

absolute difference

of two numbers is the absolute value of their difference.

The

average

of

n

elements is the

sum

of the

n

elements divided (

integer division

) by

n

.

The average of

0

elements is considered to be

0

.

Example 1:

Input:

nums = [2,5,3,9,5,3]

Output:

3

Explanation:

- The average difference of index 0 is: |2 / 1 - (5 + 3 + 9 + 5 + 3) / 5| = |2 / 1 - 25 / 5| = |2 - 5| = 3. - The average difference of index 1 is: |(2 + 5) / 2 - (3 + 9 + 5 + 3) / 4| = |7 / 2 - 20 / 4| = |3 - 5| = 2. - The average difference of index 2 is: |(2 + 5 + 3) / 3 - (9 + 5 + 3) / 3| = |10 / 3 - 17 / 3| = |3 - 5| = 2. - The average difference of index 3 is: |(2 + 5 + 3 + 9) / 4 - (5 + 3) / 2| = |19 / 4 - 8 / 2| = |4 - 4| = 0. - The average difference of index 4 is: |(2 + 5 + 3 + 9 + 5) / 5 - 3 / 1| = |24 / 5 - 3 / 1| = |4 - 3| = 1. - The average difference of index 5 is: |(2 + 5 + 3 + 9 + 5 + 3) / 6 - 0| = |27 / 6 - 0| = |4 - 0| = 4. The average difference of index 3 is the minimum average difference so return 3.

Example 2:

Input:

nums = [0]

Output:

0

Explanation:

The only index is 0 so return 0. The average difference of index 0 is: |0 / 1 - 0| = |0 - 0| = 0.

Constraints:

1 <= nums.length <= 10

5

0 <= nums[i] <= 10

5

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int minimumAverageDifference(int* nums, int numsSize) {

}
```

Go:

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

}
```

Kotlin:

```
class Solution {
fun minimumAverageDifference(nums: IntArray): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int minimumAverageDifference(List<int> nums) {

}
```

```
}
```

Scala:

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

}
}
```

Elixir:

```
defmodule Solution do
@spec minimum_average_difference(nums :: [integer]) :: integer
def minimum_average_difference(nums) do

end
end
```

Erlang:

```
-spec minimum_average_difference(Nums :: [integer()]) -> integer().
minimum_average_difference(Nums) ->
.
```

Racket:

```
(define/contract (minimum-average-difference nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Average Difference
* Difficulty: Medium
* Tags: array
*
```

```
* 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 minimumAverageDifference(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Average Difference
* Difficulty: Medium
* Tags: array
*
* 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 minimumAverageDifference(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Average Difference
Difficulty: Medium
Tags: array

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 minimumAverageDifference(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Average Difference
* Difficulty: Medium
* Tags: array
*
* 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 minimumAverageDifference = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Average Difference
* Difficulty: Medium
* Tags: array
```

```
*
* 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 minimumAverageDifference(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Average Difference
* Difficulty: Medium
* Tags: array
*
* 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 MinimumAverageDifference(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Average Difference
* Difficulty: Medium
* Tags: array
*
* 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
*/

int minimumAverageDifference(int* nums, int numsSize) {
```

```
}
```

Go Solution:

```
// Problem: Minimum Average Difference
// Difficulty: Medium
// Tags: array
//
// 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 minimumAverageDifference(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minimumAverageDifference(nums: IntArray): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Average Difference
// Difficulty: Medium
// Tags: array
//
// 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 minimum_average_difference(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int minimumAverageDifference(List<int> nums) {

}
}
```

Scala Solution:

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

```
}
}
```

Elixir Solution:

```
defmodule Solution do
@spec minimum_average_difference(nums :: [integer]) :: integer
def minimum_average_difference(nums) do

end
end
```

Erlang Solution:

```
-spec minimum_average_difference(Nums :: [integer()]) -> integer().
minimum_average_difference(Nums) ->
.
```

Racket Solution:

```
(define/contract (minimum-average-difference nums)
(-> (listof exact-integer?) exact-integer?)
)
```