Problem 2860: Happy Students

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

where

n

is the total number of students in the class. The class teacher tries to select a group of students so that all the students remain happy.

The

i

th

student will become happy if one of these two conditions is met:

The student is selected and the total number of selected students is

strictly greater than

nums[i]

.

The student is not selected and the total number of selected students is

strictly

less than

nums[i]

.

Return

the number of ways to select a group of students so that everyone remains happy.

Example 1:

Input:

nums = [1,1]

Output:

2

Explanation:

The two possible ways are: The class teacher selects no student. The class teacher selects both students to form the group. If the class teacher selects just one student to form a group then the both students will not be happy. Therefore, there are only two possible ways.

Example 2:

Input:

nums = [6,0,3,3,6,7,2,7]

Output:

3

Explanation:

The three possible ways are: The class teacher selects the student with index = 1 to form the group. The class teacher selects the students with index = 1, 2, 3, 6 to form the group. The class teacher selects all the students to form the group.

Constraints:

1 <= nums.length <= 10

5

0 <= nums[i] < nums.length

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public int countWays(List<Integer> nums) {

}
```

```
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public int CountWays(IList<int> nums) {

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

```
class Solution {
fun countWays(nums: List<Int>): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

```
object Solution {
def countWays(nums: List[Int]): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (count-ways nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Happy Students
* Difficulty: Medium
* Tags: array, 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 countWays(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Happy Students
* Difficulty: Medium
* Tags: array, 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 countWays(List<Integer> nums) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Happy Students
Difficulty: Medium
Tags: array, 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 countWays(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Happy Students
* Difficulty: Medium
* Tags: array, 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 countWays = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Happy Students
* Difficulty: Medium
* Tags: array, 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 countWays(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Happy Students
* Difficulty: Medium
* Tags: array, 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 CountWays(IList<int> nums) {

}
}
```

C Solution:

```
/*
* Problem: Happy Students
* Difficulty: Medium
* Tags: array, 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
*/

int countWays(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Happy Students
// Difficulty: Medium
// Tags: array, 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 countWays(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countWays(nums: List<Int>): Int {

}
}
```

Swift Solution:

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

```
}
}
```

Rust Solution:

```
// Problem: Happy Students
// Difficulty: Medium
// Tags: array, 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 count_ways(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

```
object Solution {
def countWays(nums: List[Int]): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (count-ways nums)
(-> (listof exact-integer?) exact-integer?)
)
```