Problem 2358: Maximum Number of Groups Entering a Competition

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given a positive integer array

grades

which represents the grades of students in a university. You would like to enter

all

these students into a competition in

ordered

non-empty groups, such that the ordering meets the following conditions:

The sum of the grades of students in the

i

th

group is

less than

the sum of the grades of students in the

(i + 1)

th

group, for all groups (except the last).

The total number of students in the

i

th

group is

less than

the total number of students in the

(i + 1)

th

group, for all groups (except the last).

Return

the

maximum

number of groups that can be formed

.

Example 1:

Input:

grades = [10,6,12,7,3,5]

Output:

3

Explanation:

The following is a possible way to form 3 groups of students: - 1

st

group has the students with grades = [12]. Sum of grades: 12. Student count: 1 - 2

nd

group has the students with grades = [6,7]. Sum of grades: 6 + 7 = 13. Student count: 2 - 3

rd

group has the students with grades = [10,3,5]. Sum of grades: 10 + 3 + 5 = 18. Student count: 3 It can be shown that it is not possible to form more than 3 groups.

Example 2:

Input:

grades = [8,8]

Output:

1

Explanation:

We can only form 1 group, since forming 2 groups would lead to an equal number of students in both groups.

Constraints:

1 <= grades.length <= 10

5

1 <= grades[i] <= 10

5

Code Snippets

C++:

```
class Solution {
public:
int maximumGroups(vector<int>& grades) {

}
};
```

Java:

```
class Solution {
public int maximumGroups(int[] grades) {

}
}
```

Python3:

```
class Solution:
def maximumGroups(self, grades: List[int]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function maximumGroups(grades: number[]): number {

};
```

C#:

```
public class Solution {
public int MaximumGroups(int[] grades) {

}
}
```

C:

```
int maximumGroups(int* grades, int gradesSize) {

}
```

Go:

```
func maximumGroups(grades []int) int {

}
```

Kotlin:

```
class Solution {
fun maximumGroups(grades: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func maximumGroups(_ grades: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn maximum_groups(grades: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} grades
# @return {Integer}
def maximum_groups(grades)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $grades
* @return Integer
*/
function maximumGroups($grades) {

}
}
```

Dart:

```
class Solution {
int maximumGroups(List<int> grades) {

}
}
```

Scala:

```
object Solution {
def maximumGroups(grades: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec maximum_groups(grades :: [integer]) :: integer
def maximum_groups(grades) do

end
end
```

Erlang:

```
-spec maximum_groups(Grades :: [integer()]) -> integer().
maximum_groups(Grades) ->
.
```

Racket:

```
(define/contract (maximum-groups grades)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Number of Groups Entering a Competition
* Difficulty: Medium
* Tags: array, greedy, math, search
*
```

```
* 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 maximumGroups(vector<int>& grades) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Number of Groups Entering a Competition
* Difficulty: Medium
* Tags: array, greedy, math, search
*
* 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 maximumGroups(int[] grades) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Number of Groups Entering a Competition
Difficulty: Medium
Tags: array, greedy, math, search

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Number of Groups Entering a Competition
* Difficulty: Medium
* Tags: array, greedy, math, search
*
* 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[]} grades
* @return {number}
*/
var maximumGroups = function(grades) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Number of Groups Entering a Competition
* Difficulty: Medium
* Tags: array, greedy, math, search
```

```
*
* 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 maximumGroups(grades: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Number of Groups Entering a Competition
* Difficulty: Medium
* Tags: array, greedy, math, search
*
* 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 MaximumGroups(int[] grades) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Number of Groups Entering a Competition
* Difficulty: Medium
* Tags: array, greedy, math, search
*
* 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 maximumGroups(int* grades, int gradesSize) {
}
```

Go Solution:

```
// Problem: Maximum Number of Groups Entering a Competition
// Difficulty: Medium
// Tags: array, greedy, math, search
//
// 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 maximumGroups(grades []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maximumGroups(grades: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func maximumGroups(_ grades: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Number of Groups Entering a Competition
// Difficulty: Medium
// Tags: array, greedy, math, search
//
// 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 maximum_groups(grades: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} grades
# @return {Integer}
def maximum_groups(grades)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $grades
* @return Integer
*/
function maximumGroups($grades) {

}
}
```

Dart Solution:

```
class Solution {
int maximumGroups(List<int> grades) {

}
}
```

Scala Solution:

```
object Solution {
def maximumGroups(grades: Array[Int]): Int = {


}
}
```

Elixir Solution:

```
defmodule Solution do
@spec maximum_groups(grades :: [integer]) :: integer
def maximum_groups(grades) do

end
end
```

Erlang Solution:

```
-spec maximum_groups(Grades :: [integer()]) -> integer().
maximum_groups(Grades) ->
.
```

Racket Solution:

```
(define/contract (maximum-groups grades)
(-> (listof exact-integer?) exact-integer?)
)
```