Problem 2910: Minimum Number of Groups to Create a Valid Assignment

Problem Information

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

Problem Description

You are given a collection of numbered

balls

and instructed to sort them into boxes for a nearly balanced distribution. There are two rules you must follow:

Balls with the same box must have the same value. But, if you have more than one ball with the same number, you can put them in different boxes.

The biggest box can only have one more ball than the smallest box.

Return the

fewest number of boxes

to sort these balls following these rules.

Example 1:

Input:

balls = [3,2,3,2,3]

Output:

2

Explanation:

We can sort

balls

into boxes as follows:

[3,3,3]

[2,2]

The size difference between the two boxes doesn't exceed one.

Example 2:

Input:

balls = [10,10,10,3,1,1]

Output:

4

Explanation:

We can sort

balls

into boxes as follows:

[10]

[10,10]

[3]

[1,1]

You can't use fewer than four boxes while still following the rules. For example, putting all three balls numbered 10 in one box would break the rule about the maximum size difference between boxes.

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int minGroupsForValidAssignment(vector<int>& balls) {

}
};
```

Java:

```
class Solution {
public int minGroupsForValidAssignment(int[] balls) {

}
}
```

Python3:

```
class Solution:
def minGroupsForValidAssignment(self, balls: List[int]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function minGroupsForValidAssignment(balls: number[]): number {

};
```

C#:

```
public class Solution {
public int MinGroupsForValidAssignment(int[] balls) {

}
}
```

C:

```
int minGroupsForValidAssignment(int* balls, int ballsSize) {

}
```

Go:

```
func minGroupsForValidAssignment(balls []int) int {

}
```

Kotlin:

```
class Solution {
fun minGroupsForValidAssignment(balls: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func minGroupsForValidAssignment(_ balls: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_groups_for_valid_assignment(balls: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} balls
# @return {Integer}
def min_groups_for_valid_assignment(balls)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $balls
* @return Integer
*/
function minGroupsForValidAssignment($balls) {

}
}
```

Dart:

```
class Solution {
int minGroupsForValidAssignment(List<int> balls) {

}
}
```

Scala:

```
object Solution {
def minGroupsForValidAssignment(balls: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_groups_for_valid_assignment(balls :: [integer]) :: integer
def min_groups_for_valid_assignment(balls) do

end
end
```

Erlang:

```
-spec min_groups_for_valid_assignment(Balls :: [integer()]) -> integer().
min_groups_for_valid_assignment(Balls) ->
.
```

Racket:

```
(define/contract (min-groups-for-valid-assignment balls)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Number of Groups to Create a Valid Assignment
* Difficulty: Medium
* Tags: array, greedy, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int minGroupsForValidAssignment(vector<int>& balls) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Number of Groups to Create a Valid Assignment
* Difficulty: Medium
* Tags: array, greedy, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int minGroupsForValidAssignment(int[] balls) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Number of Groups to Create a Valid Assignment
Difficulty: Medium
Tags: array, greedy, hash, sort

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Number of Groups to Create a Valid Assignment
* Difficulty: Medium
* Tags: array, greedy, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {number[]} balls
* @return {number}
*/
var minGroupsForValidAssignment = function(balls) {
};
```

TypeScript Solution:

```
/**
* Problem: Minimum Number of Groups to Create a Valid Assignment
* Difficulty: Medium
* Tags: array, greedy, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function minGroupsForValidAssignment(balls: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Number of Groups to Create a Valid Assignment
* Difficulty: Medium
* Tags: array, greedy, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int MinGroupsForValidAssignment(int[] balls) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Number of Groups to Create a Valid Assignment
* Difficulty: Medium
* Tags: array, greedy, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int minGroupsForValidAssignment(int* balls, int ballsSize) {

}
```

Go Solution:

```
// Problem: Minimum Number of Groups to Create a Valid Assignment
// Difficulty: Medium
// Tags: array, greedy, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func minGroupsForValidAssignment(balls []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minGroupsForValidAssignment(balls: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func minGroupsForValidAssignment(_ balls: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Number of Groups to Create a Valid Assignment
// Difficulty: Medium
// Tags: array, greedy, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn min_groups_for_valid_assignment(balls: Vec<i32>) -> i32 {


}
}
```

Ruby Solution:

```
# @param {Integer[]} balls
# @return {Integer}
def min_groups_for_valid_assignment(balls)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $balls
* @return Integer
*/
function minGroupsForValidAssignment($balls) {

}
}
```

Dart Solution:

```
class Solution {
int minGroupsForValidAssignment(List<int> balls) {
}
}
```

Scala Solution:

```
object Solution {
def minGroupsForValidAssignment(balls: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_groups_for_valid_assignment(balls :: [integer]) :: integer
def min_groups_for_valid_assignment(balls) do

end
end
```

Erlang Solution:

```
-spec min_groups_for_valid_assignment(Balls :: [integer()]) -> integer().
min_groups_for_valid_assignment(Balls) ->
.
```

Racket Solution:

```
(define/contract (min-groups-for-valid-assignment balls)
(-> (listof exact-integer?) exact-integer?)
)
```