Problem 846: Hand of Straights

Problem Information

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

Problem Description

Alice has some number of cards and she wants to rearrange the cards into groups so that each group is of size

groupSize

, and consists of

groupSize

consecutive cards.

Given an integer array

hand

where

hand[i]

is the value written on the

i

th

card and an integer

groupSize

, return

true

if she can rearrange the cards, or

false

otherwise.

Example 1:

Input:

hand = [1,2,3,6,2,3,4,7,8], groupSize = 3

Output:

true

Explanation:

Alice's hand can be rearranged as [1,2,3],[2,3,4],[6,7,8]

Example 2:

Input:

hand = [1,2,3,4,5], groupSize = 4

Output:

false

Explanation:

Alice's hand can not be rearranged into groups of 4.

Constraints:

1 <= hand.length <= 10

4

0 <= hand[i] <= 10

9

1 <= groupSize <= hand.length

Note:

This question is the same as 1296:

https://leetcode.com/problems/divide-array-in-sets-of-k-consecutive-numbers/

Code Snippets

C++:

```
class Solution {
public:
bool isNStraightHand(vector<int>& hand, int groupSize) {

}
};
```

Java:

```
class Solution {
public boolean isNStraightHand(int[] hand, int groupSize) {

}
}
```

Python3:

```
class Solution:
def isNStraightHand(self, hand: List[int], groupSize: int) -> bool:
```

Python:

```
class Solution(object):
def isNStraightHand(self, hand, groupSize):
"""
:type hand: List[int]
:type groupSize: int
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {number[]} hand
* @param {number} groupSize
* @return {boolean}
*/
var isNStraightHand = function(hand, groupSize) {

};
```

TypeScript:

```
function isNStraightHand(hand: number[], groupSize: number): boolean {

};
```

C#:

```
public class Solution {
public bool IsNStraightHand(int[] hand, int groupSize) {

}
}
```

C:

```
bool isNStraightHand(int* hand, int handSize, int groupSize) {

}
```

Go:

```
func isNStraightHand(hand []int, groupSize int) bool {

}
```

Kotlin:

```
class Solution {
fun isNStraightHand(hand: IntArray, groupSize: Int): Boolean {

}
}
```

Swift:

```
class Solution {
func isNStraightHand(_ hand: [Int], _ groupSize: Int) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn is_n_straight_hand(hand: Vec<i32>, group_size: i32) -> bool {

}
}
```

Ruby:

```
# @param {Integer[]} hand
# @param {Integer} group_size
# @return {Boolean}
def is_n_straight_hand(hand, group_size)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $hand
* @param Integer $groupSize
* @return Boolean
*/
function isNStraightHand($hand, $groupSize) {

}
}
```

Dart:

```
class Solution {
bool isNStraightHand(List<int> hand, int groupSize) {

}
}
```

Scala:

```
object Solution {
def isNStraightHand(hand: Array[Int], groupSize: Int): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec is_n_straight_hand(hand :: [integer], group_size :: integer) :: boolean
def is_n_straight_hand(hand, group_size) do

end
end
```

Erlang:

```
-spec is_n_straight_hand(Hand :: [integer()], GroupSize :: integer()) ->
boolean().
```

```
is_n_straight_hand(Hand, GroupSize) ->
.
```

Racket:

```
(define/contract (is-n-straight-hand hand groupSize)
(-> (listof exact-integer?) exact-integer? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Hand of Straights
* 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:
bool isNStraightHand(vector<int>& hand, int groupSize) {

}
};
```

Java Solution:

```
/**
* Problem: Hand of Straights
* 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 boolean isNStraightHand(int[] hand, int groupSize) {

}
}
```

Python3 Solution:

```
"""
Problem: Hand of Straights
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 isNStraightHand(self, hand: List[int], groupSize: int) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def isNStraightHand(self, hand, groupSize):
"""
:type hand: List[int]
:type groupSize: int
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Hand of Straights
* 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[]} hand
* @param {number} groupSize
* @return {boolean}
*/
var isNStraightHand = function(hand, groupSize) {

};
```

TypeScript Solution:

```
/**
* Problem: Hand of Straights
* 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 isNStraightHand(hand: number[], groupSize: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Hand of Straights
* 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 bool IsNStraightHand(int[] hand, int groupSize) {

}
}
```

C Solution:

```
/*
* Problem: Hand of Straights
* 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
*/

bool isNStraightHand(int* hand, int handSize, int groupSize) {

}
```

Go Solution:

```
// Problem: Hand of Straights
// 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 isNStraightHand(hand []int, groupSize int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isNStraightHand(hand: IntArray, groupSize: Int): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func isNStraightHand(_ hand: [Int], _ groupSize: Int) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Hand of Straights
// 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 is_n_straight_hand(hand: Vec<i32>, group_size: i32) -> bool {

}
}
```

Ruby Solution:

```
# @param {Integer[]} hand
# @param {Integer} group_size
# @return {Boolean}
def is_n_straight_hand(hand, group_size)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $hand
* @param Integer $groupSize
* @return Boolean
*/
function isNStraightHand($hand, $groupSize) {

}
}
```

Dart Solution:

```
class Solution {
bool isNStraightHand(List<int> hand, int groupSize) {

}
}
```

Scala Solution:

```
object Solution {
def isNStraightHand(hand: Array[Int], groupSize: Int): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec is_n_straight_hand(hand :: [integer], group_size :: integer) :: boolean
def is_n_straight_hand(hand, group_size) do

end
end
```

Erlang Solution:

```
-spec is_n_straight_hand(Hand :: [integer()], GroupSize :: integer()) ->
boolean().
is_n_straight_hand(Hand, GroupSize) ->
```

```
.
```

Racket Solution:

```
(define/contract (is-n-straight-hand hand groupSize)
(-> (listof exact-integer?) exact-integer? boolean?)
)
```