Problem 954: Array of Doubled Pairs

Problem Information

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

Problem Description

Given an integer array of even length

arr

, return

true

if it is possible to reorder

arr

such that

arr[2 * i + 1] = 2 * arr[2 * i]

for every

0 <= i < len(arr) / 2

, or

false

otherwise

.

Example 1:

Input:

arr = [3,1,3,6]

Output:

false

Example 2:

Input:

arr = [2,1,2,6]

Output:

false

Example 3:

Input:

arr = [4,-2,2,-4]

Output:

true

Explanation:

We can take two groups, [-2,-4] and [2,4] to form [-2,-4,2,4] or [2,4,-2,-4].

Constraints:

2 <= arr.length <= 3 * 10

4

arr.length

is even.

-10

5

<= arr[i] <= 10

5

Code Snippets

C++:

```
class Solution {
public:
bool canReorderDoubled(vector<int>& arr) {

}
};
```

Java:

```
class Solution {
public boolean canReorderDoubled(int[] arr) {

}
}
```

Python3:

```
class Solution:
def canReorderDoubled(self, arr: List[int]) -> bool:
```

Python:

```
class Solution(object):
def canReorderDoubled(self, arr):
"""
:type arr: List[int]
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {number[]} arr
* @return {boolean}
*/
var canReorderDoubled = function(arr) {

};
```

TypeScript:

```
function canReorderDoubled(arr: number[]): boolean {

};
```

C#:

```
public class Solution {
public bool CanReorderDoubled(int[] arr) {

}
}
```

C:

```
bool canReorderDoubled(int* arr, int arrSize) {

}
```

Go:

```
func canReorderDoubled(arr []int) bool {

}
```

Kotlin:

```
class Solution {
fun canReorderDoubled(arr: IntArray): Boolean {

}
}
```

Swift:

```
class Solution {
func canReorderDoubled(_ arr: [Int]) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn can_reorder_doubled(arr: Vec<i32>) -> bool {

}
}
```

Ruby:

```
# @param {Integer[]} arr
# @return {Boolean}
def can_reorder_doubled(arr)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $arr
* @return Boolean
*/
function canReorderDoubled($arr) {

}
}
```

Dart:

```
class Solution {
bool canReorderDoubled(List<int> arr) {

}
}
```

Scala:

```
object Solution {
def canReorderDoubled(arr: Array[Int]): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec can_reorder_doubled(arr :: [integer]) :: boolean
def can_reorder_doubled(arr) do

end
end
```

Erlang:

```
-spec can_reorder_doubled(Arr :: [integer()]) -> boolean().
can_reorder_doubled(Arr) ->
.
```

Racket:

```
(define/contract (can-reorder-doubled arr)
(-> (listof exact-integer?) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Array of Doubled Pairs
* 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 canReorderDoubled(vector<int>& arr) {

}
};
```

Java Solution:

```
/**
* Problem: Array of Doubled Pairs
* 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 canReorderDoubled(int[] arr) {

}
}
```

Python3 Solution:

```
"""
Problem: Array of Doubled Pairs
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 canReorderDoubled(self, arr: List[int]) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def canReorderDoubled(self, arr):
"""
:type arr: List[int]
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Array of Doubled Pairs
* 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[]} arr
* @return {boolean}
*/
var canReorderDoubled = function(arr) {

};
```

TypeScript Solution:

```
/**
* Problem: Array of Doubled Pairs
* 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 canReorderDoubled(arr: number[]): boolean {

};
```

C# Solution:

```
/*
* Problem: Array of Doubled Pairs
* 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 CanReorderDoubled(int[] arr) {

}
}
```

C Solution:

```
/*
* Problem: Array of Doubled Pairs
* 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 canReorderDoubled(int* arr, int arrSize) {

}
```

Go Solution:

```
// Problem: Array of Doubled Pairs
// 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 canReorderDoubled(arr []int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun canReorderDoubled(arr: IntArray): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func canReorderDoubled(_ arr: [Int]) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Array of Doubled Pairs
// 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 can_reorder_doubled(arr: Vec<i32>) -> bool {

}
}
```

Ruby Solution:

```
# @param {Integer[]} arr
# @return {Boolean}
def can_reorder_doubled(arr)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @return Boolean
*/
function canReorderDoubled($arr) {

}
}
```

Dart Solution:

```
class Solution {
bool canReorderDoubled(List<int> arr) {

}
}
```

Scala Solution:

```
object Solution {
def canReorderDoubled(arr: Array[Int]): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec can_reorder_doubled(arr :: [integer]) :: boolean
def can_reorder_doubled(arr) do

end
end
```

Erlang Solution:

```
-spec can_reorder_doubled(Arr :: [integer()]) -> boolean().
can_reorder_doubled(Arr) ->
.
```

Racket Solution:

```
(define/contract (can-reorder-doubled arr)
(-> (listof exact-integer?) boolean?)
)
```