Problem 266: Palindrome Permutation

Problem Information

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

Problem Description

Given a string

s

, return

true

if a permutation of the string could form a

palindrome

and

false

otherwise

.

Example 1:

Input:

s = "code"

Output:

false

Example 2:

Input:

s = "aab"

Output:

true

Example 3:

Input:

s = "carerac"

Output:

true

Constraints:

1 <= s.length <= 5000

s

consists of only lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
```

```
bool canPermutePalindrome(string s) {

}
};
```

Java:

```
class Solution {
public boolean canPermutePalindrome(String s) {

}
}
```

Python3:

```
class Solution:
def canPermutePalindrome(self, s: str) -> bool:
```

Python:

```
class Solution(object):
def canPermutePalindrome(self, s):
"""
:type s: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {boolean}
*/
var canPermutePalindrome = function(s) {

};
```

TypeScript:

```
function canPermutePalindrome(s: string): boolean {

};
```

C#:

```
public class Solution {
public bool CanPermutePalindrome(string s) {

}
}
```

C:

```
bool canPermutePalindrome(char* s) {

}
```

Go:

```
func canPermutePalindrome(s string) bool {

}
```

Kotlin:

```
class Solution {
fun canPermutePalindrome(s: String): Boolean {

}
}
```

Swift:

```
class Solution {
func canPermutePalindrome(_ s: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn can_permute_palindrome(s: String) -> bool {

}
}
```

Ruby:

```
# @param {String} s
# @return {Boolean}
def can_permute_palindrome(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Boolean
*/
function canPermutePalindrome($s) {

}
}
```

Dart:

```
class Solution {
bool canPermutePalindrome(String s) {

}
}
```

Scala:

```
object Solution {
def canPermutePalindrome(s: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec can_permute_palindrome(s :: String.t) :: boolean
def can_permute_palindrome(s) do
```

```
end
end
```

Erlang:

```
-spec can_permute_palindrome(S :: unicode:unicode_binary()) -> boolean().
can_permute_palindrome(S) ->
.
```

Racket:

```
(define/contract (can-permute-palindrome s)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Palindrome Permutation
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
bool canPermutePalindrome(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Palindrome Permutation
```

```
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public boolean canPermutePalindrome(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Palindrome Permutation
Difficulty: Easy
Tags: string, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def canPermutePalindrome(self, s: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def canPermutePalindrome(self, s):
"""
:type s: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Palindrome Permutation
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} s
* @return {boolean}
*/
var canPermutePalindrome = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Palindrome Permutation
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function canPermutePalindrome(s: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Palindrome Permutation
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
```

```
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public bool CanPermutePalindrome(string s) {

}
}
```

C Solution:

```
/*
* Problem: Palindrome Permutation
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

bool canPermutePalindrome(char* s) {

}
```

Go Solution:

```
// Problem: Palindrome Permutation
// Difficulty: Easy
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func canPermutePalindrome(s string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun canPermutePalindrome(s: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func canPermutePalindrome(_ s: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Palindrome Permutation
// Difficulty: Easy
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn can_permute_palindrome(s: String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {Boolean}
def can_permute_palindrome(s)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param String $s
* @return Boolean
*/
function canPermutePalindrome($s) {

}
}
```

Dart Solution:

```
class Solution {
bool canPermutePalindrome(String s) {

}
}
```

Scala Solution:

```
object Solution {
def canPermutePalindrome(s: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec can_permute_palindrome(s :: String.t) :: boolean
def can_permute_palindrome(s) do

end
end
```

Erlang Solution:

```
-spec can_permute_palindrome(S :: unicode:unicode_binary()) -> boolean().
can_permute_palindrome(S) ->
.
```

Racket Solution:

```
(define/contract (can-permute-palindrome s)
(-> string? boolean?)
)
```