Problem 2396: Strictly Palindromic Number

Problem Information

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

Problem Description

An integer

n

is

strictly palindromic

if, for

every

base

b

between

2

and

n - 2

(

inclusive

), the string representation of the integer

n

in base

b

is

palindromic

.

Given an integer

n

, return

true

if

n

is

strictly palindromic

and

false

otherwise

.

A string is

palindromic

if it reads the same forward and backward.

Example 1:

Input:

n = 9

Output:

false

Explanation:

In base 2: 9 = 1001 (base 2), which is palindromic. In base 3: 9 = 100 (base 3), which is not palindromic. Therefore, 9 is not strictly palindromic so we return false. Note that in bases 4, 5, 6, and 7, n = 9 is also not palindromic.

Example 2:

Input:

n = 4

Output:

false

Explanation:

We only consider base 2: 4 = 100 (base 2), which is not palindromic. Therefore, we return false.

Constraints:

4 <= n <= 10

5

Code Snippets

C++:

```
class Solution {
public:
bool isStrictlyPalindromic(int n) {

}
};
```

Java:

```
class Solution {
public boolean isStrictlyPalindromic(int n) {

}
}
```

Python3:

```
class Solution:
def isStrictlyPalindromic(self, n: int) -> bool:
```

Python:

```
class Solution(object):
def isStrictlyPalindromic(self, n):
"""
:type n: int
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {number} n
* @return {boolean}
*/
var isStrictlyPalindromic = function(n) {

};
```

TypeScript:

```
function isStrictlyPalindromic(n: number): boolean {

};
```

C#:

```
public class Solution {
public bool IsStrictlyPalindromic(int n) {

}
}
```

C:

```
bool isStrictlyPalindromic(int n) {

}
```

Go:

```
func isStrictlyPalindromic(n int) bool {

}
```

Kotlin:

```
class Solution {
fun isStrictlyPalindromic(n: Int): Boolean {

}
}
```

Swift:

```
class Solution {
func isStrictlyPalindromic(_ n: Int) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn is_strictly_palindromic(n: i32) -> bool {

}
}
```

Ruby:

```
# @param {Integer} n
# @return {Boolean}
def is_strictly_palindromic(n)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return Boolean
*/
function isStrictlyPalindromic($n) {

}
}
```

Dart:

```
class Solution {
bool isStrictlyPalindromic(int n) {

}
}
```

Scala:

```
object Solution {
def isStrictlyPalindromic(n: Int): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec is_strictly_palindromic(n :: integer) :: boolean
def is_strictly_palindromic(n) do

end
end
```

Erlang:

```
-spec is_strictly_palindromic(N :: integer()) -> boolean().
is_strictly_palindromic(N) ->
.
```

Racket:

```
(define/contract (is-strictly-palindromic n)
(-> exact-integer? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Strictly Palindromic Number
* Difficulty: Medium
* Tags: array, string, math
*
* 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:
bool isStrictlyPalindromic(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Strictly Palindromic Number
* Difficulty: Medium
* Tags: array, string, math
*
* 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 boolean isStrictlyPalindromic(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Strictly Palindromic Number
Difficulty: Medium
Tags: array, string, math

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 isStrictlyPalindromic(self, n: int) -> bool:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
def isStrictlyPalindromic(self, n):
"""
:type n: int
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Strictly Palindromic Number
* Difficulty: Medium
* Tags: array, string, math
*
* 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} n
* @return {boolean}
*/
var isStrictlyPalindromic = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Strictly Palindromic Number
* Difficulty: Medium
* Tags: array, string, math
*
* 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 isStrictlyPalindromic(n: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Strictly Palindromic Number
* Difficulty: Medium
* Tags: array, string, math
*
* 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 bool IsStrictlyPalindromic(int n) {

}
}
```

C Solution:

```
/*
* Problem: Strictly Palindromic Number
* Difficulty: Medium
* Tags: array, string, math
*
* 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
*/

bool isStrictlyPalindromic(int n) {

}
```

Go Solution:

```
// Problem: Strictly Palindromic Number
// Difficulty: Medium
// Tags: array, string, math
//
// 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 isStrictlyPalindromic(n int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isStrictlyPalindromic(n: Int): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func isStrictlyPalindromic(_ n: Int) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Strictly Palindromic Number
// Difficulty: Medium
// Tags: array, string, math
//
// 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 is_strictly_palindromic(n: i32) -> bool {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @return {Boolean}
def is_strictly_palindromic(n)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @return Boolean
*/
function isStrictlyPalindromic($n) {

}
}
```

Dart Solution:

```
class Solution {
bool isStrictlyPalindromic(int n) {

}
}
```

Scala Solution:

```
object Solution {
def isStrictlyPalindromic(n: Int): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec is_strictly_palindromic(n :: integer) :: boolean
def is_strictly_palindromic(n) do

end
end
```

Erlang Solution:

```
-spec is_strictly_palindromic(N :: integer()) -> boolean().
is_strictly_palindromic(N) ->
.
```

Racket Solution:

```
(define/contract (is-strictly-palindromic n)
(-> exact-integer? boolean?)
)
```