Problem 869: Reordered Power of 2

Problem Information

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

Problem Description

You are given an integer

n

. We reorder the digits in any order (including the original order) such that the leading digit is not zero.

Return

true

if and only if we can do this so that the resulting number is a power of two

.

Example 1:

Input:

n = 1

Output:

true

Example 2:

Input:

n = 10

Output:

false

Constraints:

1 <= n <= 10

9

Code Snippets

C++:

```
class Solution {
public:
bool reorderedPowerOf2(int n) {

}
};
```

Java:

```
class Solution {
public boolean reorderedPowerOf2(int n) {

}
}
```

Python3:

```
class Solution:
def reorderedPowerOf2(self, n: int) -> bool:
```

Python:

```
class Solution(object):
def reorderedPowerOf2(self, n):
"""
:type n: int
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {number} n
* @return {boolean}
*/
var reorderedPowerOf2 = function(n) {

};
```

TypeScript:

```
function reorderedPowerOf2(n: number): boolean {

};
```

C#:

```
public class Solution {
public bool ReorderedPowerOf2(int n) {

}
}
```

C:

```
bool reorderedPowerOf2(int n) {

}
```

Go:

```
func reorderedPowerOf2(n int) bool {

}
```

Kotlin:

```
class Solution {
fun reorderedPowerOf2(n: Int): Boolean {

}
}
```

Swift:

```
class Solution {
func reorderedPowerOf2(_ n: Int) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn reordered_power_of2(n: i32) -> bool {

}
}
```

Ruby:

```
# @param {Integer} n
# @return {Boolean}
def reordered_power_of2(n)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return Boolean
*/
function reorderedPowerOf2($n) {

}
}
```

Dart:

```
class Solution {
bool reorderedPowerOf2(int n) {

}
}
```

Scala:

```
object Solution {
def reorderedPowerOf2(n: Int): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec reordered_power_of2(n :: integer) :: boolean
def reordered_power_of2(n) do

end
end
```

Erlang:

```
-spec reordered_power_of2(N :: integer()) -> boolean().
reordered_power_of2(N) ->
.
```

Racket:

```
(define/contract (reordered-power-of2 n)
(-> exact-integer? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Reordered Power of 2
* Difficulty: Medium
* Tags: math, hash, sort
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
bool reorderedPowerOf2(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Reordered Power of 2
* Difficulty: Medium
* Tags: math, hash, sort
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

class Solution {
public boolean reorderedPowerOf2(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Reordered Power of 2
Difficulty: Medium
Tags: math, hash, sort
```

```
Approach: Use hash map for O(1) lookups
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(n) for hash map
"""

class Solution:
def reorderedPowerOf2(self, n: int) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def reorderedPowerOf2(self, n):
"""
:type n: int
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Reordered Power of 2
* Difficulty: Medium
* Tags: math, hash, sort
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

/**
* @param {number} n
* @return {boolean}
*/
var reorderedPowerOf2 = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Reordered Power of 2
* Difficulty: Medium
* Tags: math, hash, sort
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

function reorderedPowerOf2(n: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Reordered Power of 2
* Difficulty: Medium
* Tags: math, hash, sort
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

public class Solution {
public bool ReorderedPowerOf2(int n) {

}
}
```

C Solution:

```
/*
* Problem: Reordered Power of 2
* Difficulty: Medium
* Tags: math, hash, sort
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
```

```
*/

bool reorderedPowerOf2(int n) {

}
```

Go Solution:

```
// Problem: Reordered Power of 2
// Difficulty: Medium
// Tags: math, hash, sort
//
// Approach: Use hash map for O(1) lookups
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(n) for hash map

func reorderedPowerOf2(n int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun reorderedPowerOf2(n: Int): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func reorderedPowerOf2(_ n: Int) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Reordered Power of 2
// Difficulty: Medium
// Tags: math, hash, sort
```

```
//
// Approach: Use hash map for O(1) lookups
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(n) for hash map

impl Solution {
pub fn reordered_power_of2(n: i32) -> bool {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @return {Boolean}
def reordered_power_of2(n)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @return Boolean
*/
function reorderedPowerOf2($n) {

}
}
```

Dart Solution:

```
class Solution {
bool reorderedPowerOf2(int n) {

}
}
```

Scala Solution:

```
object Solution {
def reorderedPowerOf2(n: Int): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec reordered_power_of2(n :: integer) :: boolean
def reordered_power_of2(n) do

end
end
```

Erlang Solution:

```
-spec reordered_power_of2(N :: integer()) -> boolean().
reordered_power_of2(N) ->
.
```

Racket Solution:

```
(define/contract (reordered-power-of2 n)
(-> exact-integer? boolean?)
)
```