Problem 202: Happy Number

Problem Information

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

Problem Description

Write an algorithm to determine if a number

n

is happy.

A

happy number

is a number defined by the following process:

Starting with any positive integer, replace the number by the sum of the squares of its digits.

Repeat the process until the number equals 1 (where it will stay), or it

loops endlessly in a cycle

which does not include 1.

Those numbers for which this process

ends in 1

are happy.

Return

true

if

n

is a happy number, and

false

if not

.

Example 1:

Input:

n = 19

Output:

true

Explanation:

1

2

+ 9

2

= 82 8

2

+ 2

2

= 68 6

2

+ 8

2

= 100 1

2

+ 0

2

+ 0

2

= 1

Example 2:

Input:

n = 2

Output:

false

Constraints:

1 <= n <= 2

31

- 1

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
bool isHappy(int n) {

}
```

Go:

```
func isHappy(n int) bool {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Happy Number
* Difficulty: Easy
* Tags: array, math, hash
*
* 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 isHappy(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Happy Number
* Difficulty: Easy
* Tags: array, math, hash
*
* 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 isHappy(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Happy Number
Difficulty: Easy
Tags: array, math, hash

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

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Happy Number
* Difficulty: Easy
* Tags: array, math, hash
*
* 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} n
* @return {boolean}
*/
var isHappy = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Happy Number
* Difficulty: Easy
* Tags: array, math, hash
*
* 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 isHappy(n: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Happy Number
* Difficulty: Easy
* Tags: array, math, hash
*
* 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 IsHappy(int n) {

}
}
```

C Solution:

```
/*
* Problem: Happy Number
* Difficulty: Easy
* Tags: array, math, hash
*
* 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 isHappy(int n) {

}
```

Go Solution:

```
// Problem: Happy Number
// Difficulty: Easy
// Tags: array, math, hash
//
// 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 isHappy(n int) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Happy Number
// Difficulty: Easy
// Tags: array, math, hash
//
// 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_happy(n: i32) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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