Problem 633: Sum of Square Numbers

Problem Information

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

Problem Description

Given a non-negative integer

c

, decide whether there're two integers

a

and

b

such that

a

2

+ b

2

= c

.

Example 1:

Input:

c = 5

Output:

true

Explanation:

1 * 1 + 2 * 2 = 5

Example 2:

Input:

c = 3

Output:

false

Constraints:

0 <= c <= 2

31

- 1

Code Snippets

C++:

```
class Solution {
public:
```

```
bool judgeSquareSum(int c) {

}
};
```

Java:

```
class Solution {
public boolean judgeSquareSum(int c) {

}
}
```

Python3:

```
class Solution:
def judgeSquareSum(self, c: int) -> bool:
```

Python:

```
class Solution(object):
def judgeSquareSum(self, c):
"""
:type c: int
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {number} c
* @return {boolean}
*/
var judgeSquareSum = function(c) {

};
```

TypeScript:

```
function judgeSquareSum(c: number): boolean {

};
```

C#:

```
public class Solution {
public bool JudgeSquareSum(int c) {

}
}
```

C:

```
bool judgeSquareSum(int c) {

}
```

Go:

```
func judgeSquareSum(c int) bool {

}
```

Kotlin:

```
class Solution {
fun judgeSquareSum(c: Int): Boolean {

}
}
```

Swift:

```
class Solution {
func judgeSquareSum(_ c: Int) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn judge_square_sum(c: i32) -> bool {

}
}
```

Ruby:

```
# @param {Integer} c
# @return {Boolean}
def judge_square_sum(c)

end
```

PHP:

```
class Solution {

/**
* @param Integer $c
* @return Boolean
*/
function judgeSquareSum($c) {

}
}
```

Dart:

```
class Solution {
bool judgeSquareSum(int c) {

}
}
```

Scala:

```
object Solution {
def judgeSquareSum(c: Int): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec judge_square_sum(c :: integer) :: boolean
def judge_square_sum(c) do
```

```
end
end
```

Erlang:

```
-spec judge_square_sum(C :: integer()) -> boolean().
judge_square_sum(C) ->
.
```

Racket:

```
(define/contract (judge-square-sum c)
(-> exact-integer? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Sum of Square Numbers
* Difficulty: Medium
* Tags: array, math, search
*
* 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 judgeSquareSum(int c) {

}
};
```

Java Solution:

```
/**
* Problem: Sum of Square Numbers
```

```
* Difficulty: Medium
* Tags: array, math, search
*
* 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 judgeSquareSum(int c) {

}
}
```

Python3 Solution:

```
"""
Problem: Sum of Square Numbers
Difficulty: Medium
Tags: array, math, search

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 judgeSquareSum(self, c: int) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def judgeSquareSum(self, c):
"""
:type c: int
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Sum of Square Numbers
* Difficulty: Medium
* Tags: array, math, search
*
* 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} c
* @return {boolean}
*/
var judgeSquareSum = function(c) {

};
```

TypeScript Solution:

```
/**
* Problem: Sum of Square Numbers
* Difficulty: Medium
* Tags: array, math, search
*
* 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 judgeSquareSum(c: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Sum of Square Numbers
* Difficulty: Medium
* Tags: array, math, search
*
* 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 JudgeSquareSum(int c) {

}
}
```

C Solution:

```
/*
* Problem: Sum of Square Numbers
* Difficulty: Medium
* Tags: array, math, search
*
* 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 judgeSquareSum(int c) {

}
```

Go Solution:

```
// Problem: Sum of Square Numbers
// Difficulty: Medium
// Tags: array, math, search
//
// 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 judgeSquareSum(c int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun judgeSquareSum(c: Int): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func judgeSquareSum(_ c: Int) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Sum of Square Numbers
// Difficulty: Medium
// Tags: array, math, search
//
// 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 judge_square_sum(c: i32) -> bool {

}
}
```

Ruby Solution:

```
# @param {Integer} c
# @return {Boolean}
def judge_square_sum(c)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param Integer $c
* @return Boolean
*/
function judgeSquareSum($c) {

}
}
```

Dart Solution:

```
class Solution {
bool judgeSquareSum(int c) {

}
}
```

Scala Solution:

```
object Solution {
def judgeSquareSum(c: Int): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec judge_square_sum(c :: integer) :: boolean
def judge_square_sum(c) do

end
end
```

Erlang Solution:

```
-spec judge_square_sum(C :: integer()) -> boolean().
judge_square_sum(C) ->
.
```

Racket Solution:

```
(define/contract (judge-square-sum c)
(-> exact-integer? boolean?)
)
```