Problem 367: Valid Perfect Square

Problem Information

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

Problem Description

Given a positive integer num, return

true

if

num

is a perfect square or

false

otherwise

.

A

perfect square

is an integer that is the square of an integer. In other words, it is the product of some integer with itself.

You must not use any built-in library function, such as

sqrt

.

Example 1:

Input:

num = 16

Output:

true

Explanation:

We return true because 4 * 4 = 16 and 4 is an integer.

Example 2:

Input:

num = 14

Output:

false

Explanation:

We return false because 3.742 * 3.742 = 14 and 3.742 is not an integer.

Constraints:

1 <= num <= 2

31

- 1

Code Snippets

C++:

```
class Solution {
public:
bool isPerfectSquare(int num) {

}
};
```

Java:

```
class Solution {
public boolean isPerfectSquare(int num) {

}
}
```

Python3:

```
class Solution:
def isPerfectSquare(self, num: int) -> bool:
```

Python:

```
class Solution(object):
def isPerfectSquare(self, num):
"""
:type num: int
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {number} num
* @return {boolean}
*/
var isPerfectSquare = function(num) {

};
```

TypeScript:

```
function isPerfectSquare(num: number): boolean {

};
```

C#:

```
public class Solution {
public bool IsPerfectSquare(int num) {

}
}
```

C:

```
bool isPerfectSquare(int num) {

}
```

Go:

```
func isPerfectSquare(num int) bool {

}
```

Kotlin:

```
class Solution {
fun isPerfectSquare(num: Int): Boolean {

}
}
```

Swift:

```
class Solution {
func isPerfectSquare(_ num: Int) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn is_perfect_square(num: i32) -> bool {

}
}
```

Ruby:

```
# @param {Integer} num
# @return {Boolean}
def is_perfect_square(num)

end
```

PHP:

```
class Solution {

/**
* @param Integer $num
* @return Boolean
*/
function isPerfectSquare($num) {

}
}
```

Dart:

```
class Solution {
bool isPerfectSquare(int num) {

}
}
```

Scala:

```
object Solution {
def isPerfectSquare(num: Int): Boolean = {

}
```

```
}
```

Elixir:

```
defmodule Solution do
@spec is_perfect_square(num :: integer) :: boolean
def is_perfect_square(num) do

end
end
```

Erlang:

```
-spec is_perfect_square(Num :: integer()) -> boolean().
is_perfect_square(Num) ->
.
```

Racket:

```
(define/contract (is-perfect-square num)
(-> exact-integer? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Valid Perfect Square
* Difficulty: Easy
* Tags: math, search
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
bool isPerfectSquare(int num) {
```

```
}
};
```

Java Solution:

```
/**
* Problem: Valid Perfect Square
* Difficulty: Easy
* Tags: math, search
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public boolean isPerfectSquare(int num) {

}
}
```

Python3 Solution:

```
"""
Problem: Valid Perfect Square
Difficulty: Easy
Tags: math, search

Approach: Optimized algorithm based on problem constraints
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

```
class Solution(object):
def isPerfectSquare(self, num):
"""
:type num: int
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Valid Perfect Square
* Difficulty: Easy
* Tags: math, search
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} num
* @return {boolean}
*/
var isPerfectSquare = function(num) {

};
```

TypeScript Solution:

```
/**
* Problem: Valid Perfect Square
* Difficulty: Easy
* Tags: math, search
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

function isPerfectSquare(num: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Valid Perfect Square
* Difficulty: Easy
* Tags: math, search
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public bool IsPerfectSquare(int num) {

}
}
```

C Solution:

```
/*
* Problem: Valid Perfect Square
* Difficulty: Easy
* Tags: math, search
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

bool isPerfectSquare(int num) {

}
```

Go Solution:

```
// Problem: Valid Perfect Square
// Difficulty: Easy
// Tags: math, search
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
```

```
// Space Complexity: O(1) to O(n) depending on approach

func isPerfectSquare(num int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isPerfectSquare(num: Int): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func isPerfectSquare(_ num: Int) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Valid Perfect Square
// Difficulty: Easy
// Tags: math, search
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn is_perfect_square(num: i32) -> bool {

}
}
```

Ruby Solution:

```
# @param {Integer} num
# @return {Boolean}
def is_perfect_square(num)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $num
* @return Boolean
*/
function isPerfectSquare($num) {

}
}
```

Dart Solution:

```
class Solution {
bool isPerfectSquare(int num) {

}
}
```

Scala Solution:

```
object Solution {
def isPerfectSquare(num: Int): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec is_perfect_square(num :: integer) :: boolean
def is_perfect_square(num) do

end
```

```
end
```

Erlang Solution:

```
-spec is_perfect_square(Num :: integer()) -> boolean().
is_perfect_square(Num) ->
.
```

Racket Solution:

```
(define/contract (is-perfect-square num)
(-> exact-integer? boolean?)
)
```