Problem 279: Perfect Squares

Problem Information

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

Problem Description

Given an integer

n

, return

the least number of perfect square numbers that sum to

n

.

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. For example,

1

,

4

,

9

, and

16

are perfect squares while

3

and

11

are not.

Example 1:

Input:

n = 12

Output:

3

Explanation:

12 = 4 + 4 + 4.

Example 2:

Input:

n = 13

Output:

2

Explanation:

13 = 4 + 9.

Constraints:

1 <= n <= 10

4

Code Snippets

C++:

```
class Solution {
public:
int numSquares(int n) {

}
};
```

Java:

```
class Solution {
public int numSquares(int n) {

}
}
```

Python3:

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

Python:

```
class Solution(object):
def numSquares(self, n):
```

```
"""
:type n: int
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

```
function numSquares(n: number): number {

};
```

C#:

```
public class Solution {
public int NumSquares(int n) {

}
}
```

C:

```
int numSquares(int n) {

}
```

Go:

```
func numSquares(n int) int {

}
```

Kotlin:

```
class Solution {
fun numSquares(n: Int): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int numSquares(int n) {

}
}
```

Scala:

```
object Solution {
def numSquares(n: Int): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec num_squares(N :: integer()) -> integer().
num_squares(N) ->
.
```

Racket:

```
(define/contract (num-squares n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Perfect Squares
* Difficulty: Medium
* Tags: dp, math, search
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int numSquares(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Perfect Squares
* Difficulty: Medium
* Tags: dp, math, search
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int numSquares(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Perfect Squares
Difficulty: Medium
Tags: dp, math, search
```

```
Approach: Dynamic programming with memoization or tabulation
Time Complexity: O(n * m) where n and m are problem dimensions
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Perfect Squares
* Difficulty: Medium
* Tags: dp, math, search
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Perfect Squares
* Difficulty: Medium
* Tags: dp, math, search
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

function numSquares(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Perfect Squares
* Difficulty: Medium
* Tags: dp, math, search
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int NumSquares(int n) {

}
}
```

C Solution:

```
/*
* Problem: Perfect Squares
* Difficulty: Medium
* Tags: dp, math, search
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

int numSquares(int n) {

}
```

Go Solution:

```
// Problem: Perfect Squares
// Difficulty: Medium
// Tags: dp, math, search
//
// Approach: Dynamic programming with memoization or tabulation
// Time Complexity: O(n * m) where n and m are problem dimensions
// Space Complexity: O(n) or O(n * m) for DP table

func numSquares(n int) int {

}
```

Kotlin Solution:

```
class Solution {
fun numSquares(n: Int): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Perfect Squares
// Difficulty: Medium
// Tags: dp, math, search
```

```
//
// Approach: Dynamic programming with memoization or tabulation
// Time Complexity: O(n * m) where n and m are problem dimensions
// Space Complexity: O(n) or O(n * m) for DP table

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int numSquares(int n) {

}
}
```

Scala Solution:

```
object Solution {
def numSquares(n: Int): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec num_squares(N :: integer()) -> integer().
num_squares(N) ->
.
```

Racket Solution:

```
(define/contract (num-squares n)
(-> exact-integer? exact-integer?)
)
```