

# Problem 1414: Find the Minimum Number of Fibonacci Numbers Whose Sum Is K

## Problem Information

Difficulty: **Medium**

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

Given an integer

$k$

,

return the minimum number of Fibonacci numbers whose sum is equal to

$k$

. The same Fibonacci number can be used multiple times.

The Fibonacci numbers are defined as:

$F$

$1$

$= 1$

$F$

$2$

$= 1$

F

n

= F

n-1

+ F

n-2

for

n > 2.

It is guaranteed that for the given constraints we can always find such Fibonacci numbers that sum up to

k

.

Example 1:

Input:

k = 7

Output:

2

Explanation:

The Fibonacci numbers are: 1, 1, 2, 3, 5, 8, 13, ... For k = 7 we can use 2 + 5 = 7.

Example 2:

Input:

$k = 10$

Output:

2

Explanation:

For  $k = 10$  we can use  $2 + 8 = 10$ .

Example 3:

Input:

$k = 19$

Output:

3

Explanation:

For  $k = 19$  we can use  $1 + 5 + 13 = 19$ .

Constraints:

$1 \leq k \leq 10$

9

## Code Snippets

**C++:**

```
class Solution {  
public:
```

```
int findMinFibonacciNumbers(int k) {  
  
}  
};
```

### Java:

```
class Solution {  
    public int findMinFibonacciNumbers(int k) {  
  
    }  
}
```

### Python3:

```
class Solution:  
    def findMinFibonacciNumbers(self, k: int) -> int:
```

### Python:

```
class Solution(object):  
    def findMinFibonacciNumbers(self, k):  
        """  
        :type k: int  
        :rtype: int  
        """
```

### JavaScript:

```
/**  
 * @param {number} k  
 * @return {number}  
 */  
var findMinFibonacciNumbers = function(k) {  
  
};
```

### TypeScript:

```
function findMinFibonacciNumbers(k: number): number {  
  
};
```

**C#:**

```
public class Solution {  
    public int FindMinFibonacciNumbers(int k) {  
  
    }  
}
```

**C:**

```
int findMinFibonacciNumbers(int k) {  
  
}
```

**Go:**

```
func findMinFibonacciNumbers(k int) int {  
  
}
```

**Kotlin:**

```
class Solution {  
    fun findMinFibonacciNumbers(k: Int): Int {  
  
    }  
}
```

**Swift:**

```
class Solution {  
    func findMinFibonacciNumbers(_ k: Int) -> Int {  
  
    }  
}
```

**Rust:**

```
impl Solution {  
    pub fn find_min_fibonacci_numbers(k: i32) -> i32 {  
  
    }  
}
```

## Ruby:

```
# @param {Integer} k
# @return {Integer}
def find_min_fibonacci_numbers(k)

end
```

## PHP:

```
class Solution {

    /**
     * @param Integer $k
     * @return Integer
     */
    function findMinFibonacciNumbers($k) {

    }

}
```

## Dart:

```
class Solution {
  int findMinFibonacciNumbers(int k) {

  }
}
```

## Scala:

```
object Solution {
  def findMinFibonacciNumbers(k: Int): Int = {

  }
}
```

## Elixir:

```
defmodule Solution do
  @spec find_min_fibonacci_numbers(k :: integer) :: integer
  def find_min_fibonacci_numbers(k) do
```

```
end
end
```

### Erlang:

```
-spec find_min_fibonacci_numbers(K :: integer()) -> integer().
find_min_fibonacci_numbers(K) ->
.
```

### Racket:

```
(define/contract (find-min-fibonacci-numbers k)
  (-> exact-integer? exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
 * Problem: Find the Minimum Number of Fibonacci Numbers Whose Sum Is K
 * Difficulty: Medium
 * Tags: greedy, math
 *
 * Approach: Greedy algorithm with local optimal choices
 * Time Complexity: O(n) to O(n^2) depending on approach
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
public:
    int findMinFibonacciNumbers(int k) {

    }
};
```

### Java Solution:

```
/**
 * Problem: Find the Minimum Number of Fibonacci Numbers Whose Sum Is K
```

```

* Difficulty: Medium
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int findMinFibonacciNumbers(int k) {

}

}

```

### Python3 Solution:

```

"""
Problem: Find the Minimum Number of Fibonacci Numbers Whose Sum Is K
Difficulty: Medium
Tags: greedy, math

Approach: Greedy algorithm with local optimal choices
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def findMinFibonacciNumbers(self, k: int) -> int:
# TODO: Implement optimized solution
pass

```

### Python Solution:

```

class Solution(object):
def findMinFibonacciNumbers(self, k):
"""
:type k: int
:rtype: int
"""

```

### JavaScript Solution:

```

/**
 * Problem: Find the Minimum Number of Fibonacci Numbers Whose Sum Is K
 * Difficulty: Medium
 * Tags: greedy, math
 *
 * Approach: Greedy algorithm with local optimal choices
 * Time Complexity: O(n) to O(n^2) depending on approach
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * @param {number} k
 * @return {number}
 */
var findMinFibonacciNumbers = function(k) {

};

```

### TypeScript Solution:

```

/**
 * Problem: Find the Minimum Number of Fibonacci Numbers Whose Sum Is K
 * Difficulty: Medium
 * Tags: greedy, math
 *
 * Approach: Greedy algorithm with local optimal choices
 * Time Complexity: O(n) to O(n^2) depending on approach
 * Space Complexity: O(1) to O(n) depending on approach
 */

function findMinFibonacciNumbers(k: number): number {

};

```

### C# Solution:

```

/*
 * Problem: Find the Minimum Number of Fibonacci Numbers Whose Sum Is K
 * Difficulty: Medium
 * Tags: greedy, math
 *
 * Approach: Greedy algorithm with local optimal choices

```

```

* 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 int FindMinFibonacciNumbers(int k) {

}

}

```

### C Solution:

```

/*
* Problem: Find the Minimum Number of Fibonacci Numbers Whose Sum Is K
* Difficulty: Medium
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

int findMinFibonacciNumbers(int k) {

}

```

### Go Solution:

```

// Problem: Find the Minimum Number of Fibonacci Numbers Whose Sum Is K
// Difficulty: Medium
// Tags: greedy, math
//
// Approach: Greedy algorithm with local optimal choices
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func findMinFibonacciNumbers(k int) int {

}

```

### Kotlin Solution:

```

class Solution {
    fun findMinFibonacciNumbers(k: Int): Int {

    }
}

```

### Swift Solution:

```

class Solution {
    func findMinFibonacciNumbers(_ k: Int) -> Int {

    }
}

```

### Rust Solution:

```

// Problem: Find the Minimum Number of Fibonacci Numbers Whose Sum Is K
// Difficulty: Medium
// Tags: greedy, math
//
// Approach: Greedy algorithm with local optimal choices
// 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 find_min_fibonacci_numbers(k: i32) -> i32 {

    }
}

```

### Ruby Solution:

```

# @param {Integer} k
# @return {Integer}
def find_min_fibonacci_numbers(k)

end

```

### PHP Solution:

```

class Solution {

```

```

/**
 * @param Integer $k
 * @return Integer
 */
function findMinFibonacciNumbers($k) {

}

}

```

### Dart Solution:

```

class Solution {
  int findMinFibonacciNumbers(int k) {

  }
}

```

### Scala Solution:

```

object Solution {
  def findMinFibonacciNumbers(k: Int): Int = {

  }
}

```

### Elixir Solution:

```

defmodule Solution do
  @spec find_min_fibonacci_numbers(k :: integer) :: integer
  def find_min_fibonacci_numbers(k) do

  end
end

```

### Erlang Solution:

```

-spec find_min_fibonacci_numbers(K :: integer()) -> integer().
find_min_fibonacci_numbers(K) ->
.

```

### Racket Solution:

```
(define/contract (find-min-fibonacci-numbers k)
  (-> exact-integer? exact-integer?)
)
```