Problem 3116: Kth Smallest Amount With Single Denomination Combination

Problem Information

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

Problem Description

You are given an integer array

coins

representing coins of different denominations and an integer

k

.

You have an infinite number of coins of each denomination. However, you are

not allowed

to combine coins of different denominations.

Return the

k

th

smallest

amount that can be made using these coins.

Example 1:

Input:

coins = [3,6,9], k = 3

Output:

9

Explanation:

The given coins can make the following amounts:

Coin 3 produces multiples of 3: 3, 6, 9, 12, 15, etc.

Coin 6 produces multiples of 6: 6, 12, 18, 24, etc.

Coin 9 produces multiples of 9: 9, 18, 27, 36, etc.

All of the coins combined produce: 3, 6,

9

, 12, 15, etc.

Example 2:

Input:

coins = [5,2], k = 7

Output:

12

Explanation:

The given coins can make the following amounts:

Coin 5 produces multiples of 5: 5, 10, 15, 20, etc.

Coin 2 produces multiples of 2: 2, 4, 6, 8, 10, 12, etc.

All of the coins combined produce: 2, 4, 5, 6, 8, 10,

12

, 14, 15, etc.

Constraints:

1 <= coins.length <= 15

1 <= coins[i] <= 25

1 <= k <= 2 * 10

9

coins

contains pairwise distinct integers.

Code Snippets

C++:

```
class Solution {
public:
long long findKthSmallest(vector<int>& coins, int k) {

}
};
```

Java:

```
class Solution {
public long findKthSmallest(int[] coins, int k) {

}
}
```

Python3:

```
class Solution:
def findKthSmallest(self, coins: List[int], k: int) -> int:
```

Python:

```
class Solution(object):
def findKthSmallest(self, coins, k):
"""
:type coins: List[int]
:type k: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} coins
* @param {number} k
* @return {number}
*/
var findKthSmallest = function(coins, k) {

};
```

TypeScript:

```
function findKthSmallest(coins: number[], k: number): number {

};
```

C#:

```
public class Solution {
public long FindKthSmallest(int[] coins, int k) {
```

```
}
}
```

C:

```
long long findKthSmallest(int* coins, int coinsSize, int k) {

}
```

Go:

```
func findKthSmallest(coins []int, k int) int64 {

}
```

Kotlin:

```
class Solution {
fun findKthSmallest(coins: IntArray, k: Int): Long {

}
}
```

Swift:

```
class Solution {
func findKthSmallest(_ coins: [Int], _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn find_kth_smallest(coins: Vec<i32>, k: i32) -> i64 {

}
}
```

Ruby:

```
# @param {Integer[]} coins
# @param {Integer} k
# @return {Integer}
def find_kth_smallest(coins, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $coins
* @param Integer $k
* @return Integer
*/
function findKthSmallest($coins, $k) {

}
}
```

Dart:

```
class Solution {
int findKthSmallest(List<int> coins, int k) {

}
}
```

Scala:

```
object Solution {
def findKthSmallest(coins: Array[Int], k: Int): Long = {

}
}
```

Elixir:

```
defmodule Solution do
@spec find_kth_smallest(coins :: [integer], k :: integer) :: integer
def find_kth_smallest(coins, k) do
```

```
end
end
```

Erlang:

```
-spec find_kth_smallest(Coins :: [integer()], K :: integer()) -> integer().
find_kth_smallest(Coins, K) ->
.
```

Racket:

```
(define/contract (find-kth-smallest coins k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Kth Smallest Amount With Single Denomination Combination
* Difficulty: Hard
* 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:
long long findKthSmallest(vector<int>& coins, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Kth Smallest Amount With Single Denomination Combination
```

```
* Difficulty: Hard
* 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 long findKthSmallest(int[] coins, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Kth Smallest Amount With Single Denomination Combination
Difficulty: Hard
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 findKthSmallest(self, coins: List[int], k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def findKthSmallest(self, coins, k):
"""
:type coins: List[int]
:type k: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Kth Smallest Amount With Single Denomination Combination
* Difficulty: Hard
* 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[]} coins
* @param {number} k
* @return {number}
*/
var findKthSmallest = function(coins, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Kth Smallest Amount With Single Denomination Combination
* Difficulty: Hard
* 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 findKthSmallest(coins: number[], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Kth Smallest Amount With Single Denomination Combination
* Difficulty: Hard
* 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 long FindKthSmallest(int[] coins, int k) {

}
}
```

C Solution:

```
/*
* Problem: Kth Smallest Amount With Single Denomination Combination
* Difficulty: Hard
* 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
*/

long long findKthSmallest(int* coins, int coinsSize, int k) {

}
```

Go Solution:

```
// Problem: Kth Smallest Amount With Single Denomination Combination
// Difficulty: Hard
// 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 findKthSmallest(coins []int, k int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun findKthSmallest(coins: IntArray, k: Int): Long {

}
}
```

Swift Solution:

```
class Solution {
func findKthSmallest(_ coins: [Int], _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Kth Smallest Amount With Single Denomination Combination
// Difficulty: Hard
// 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 find_kth_smallest(coins: Vec<i32>, k: i32) -> i64 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} coins
# @param {Integer} k
# @return {Integer}
def find_kth_smallest(coins, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $coins
* @param Integer $k
* @return Integer
*/
function findKthSmallest($coins, $k) {

}
}
```

Dart Solution:

```
class Solution {
int findKthSmallest(List<int> coins, int k) {

}
}
```

Scala Solution:

```
object Solution {
def findKthSmallest(coins: Array[Int], k: Int): Long = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec find_kth_smallest(coins :: [integer], k :: integer) :: integer
def find_kth_smallest(coins, k) do

end
end
```

Erlang Solution:

```
-spec find_kth_smallest(Coins :: [integer()], K :: integer()) -> integer().
find_kth_smallest(Coins, K) ->
.
```

Racket Solution:

```
(define/contract (find-kth-smallest coins k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```