Problem 2507: Smallest Value After Replacing With Sum of Prime Factors

Problem Information

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

Problem Description

You are given a positive integer

n

.

Continuously replace

n

with the sum of its

prime factors

.

Note that if a prime factor divides

n

multiple times, it should be included in the sum as many times as it divides

n

.

Return

the smallest value

n

will take on.

Example 1:

Input:

n = 15

Output:

5

Explanation:

Initially, n = 15. 15 = 3 * 5, so replace n with 3 + 5 = 8. 8 = 2 * 2 * 2, so replace n with 2 + 2 + 2 = 6. 6 = 2 * 3, so replace n with 2 + 3 = 5. 5 is the smallest value n will take on.

Example 2:

Input:

n = 3

Output:

3

Explanation:

Initially, n = 3. 3 is the smallest value n will take on.

Constraints:

2 <= n <= 10

5

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number} n
```

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int smallestValue(int n) {

}
```

Go:

```
func smallestValue(n int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (smallest-value n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Smallest Value After Replacing With Sum of Prime Factors
* Difficulty: Medium
* Tags: math
*
* 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:
int smallestValue(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Smallest Value After Replacing With Sum of Prime Factors
* Difficulty: Medium
* Tags: math
*
* 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 int smallestValue(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Smallest Value After Replacing With Sum of Prime Factors
Difficulty: Medium
Tags: math

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 smallestValue(self, n: int) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Smallest Value After Replacing With Sum of Prime Factors
* Difficulty: Medium
* Tags: math
*
* 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} n
* @return {number}
*/
var smallestValue = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Smallest Value After Replacing With Sum of Prime Factors
* Difficulty: Medium
* Tags: math
*
* 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 smallestValue(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Smallest Value After Replacing With Sum of Prime Factors
* Difficulty: Medium
* Tags: math
*
* 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 int SmallestValue(int n) {

}
}
```

C Solution:

```
/*
* Problem: Smallest Value After Replacing With Sum of Prime Factors
* Difficulty: Medium
* Tags: math
*
* 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
*/

int smallestValue(int n) {

}
```

Go Solution:

```
// Problem: Smallest Value After Replacing With Sum of Prime Factors
// Difficulty: Medium
// Tags: math
//
// 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 smallestValue(n int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Smallest Value After Replacing With Sum of Prime Factors
// Difficulty: Medium
// Tags: math
//
// 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 smallest_value(n: i32) -> i32 {

}
```

```
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (smallest-value n)
(-> exact-integer? exact-integer?)
)
```