Problem 793: Preimage Size of Factorial Zeroes Function

Problem Information

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

Problem Description

Let

f(x)

be the number of zeroes at the end of

x!

. Recall that

x! = 1 * 2 * 3 * ... * x

and by convention,

0! = 1

.

For example,

f(3) = 0

because

3! = 6

has no zeroes at the end, while

f(11) = 2

because

11! = 39916800

has two zeroes at the end.

Given an integer

k

, return the number of non-negative integers

x

have the property that

f(x) = k

.

Example 1:

Input:

k = 0

Output:

5

Explanation:

0!, 1!, 2!, 3!, and 4! end with k = 0 zeroes.

Example 2:

Input:

k = 5

Output:

0

Explanation:

There is no x such that x! ends in k = 5 zeroes.

Example 3:

Input:

k = 3

Output:

5

Constraints:

0 <= k <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int preimageSizeFZF(int k) {
```

```
}
};
```

Java:

```
class Solution {
public int preimageSizeFZF(int k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function preimageSizeFZF(k: number): number {

};
```

C#:

```
public class Solution {
public int PreimageSizeFZF(int k) {

}
}
```

C:

```
int preimageSizeFZF(int k) {

}
```

Go:

```
func preimageSizeFZF(k int) int {

}
```

Kotlin:

```
class Solution {
fun preimageSizeFZF(k: Int): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (preimage-size-fzf k)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Preimage Size of Factorial Zeroes Function
* Difficulty: Hard
* 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:
int preimageSizeFZF(int k) {

}
};
```

Java Solution:

```
/**
* Problem: Preimage Size of Factorial Zeroes Function
* Difficulty: Hard
* 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 int preimageSizeFZF(int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Preimage Size of Factorial Zeroes Function
Difficulty: Hard
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 preimageSizeFZF(self, k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Preimage Size of Factorial Zeroes Function
* Difficulty: Hard
```

```
 * 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} k
 * @return {number}
 */
var preimageSizeFZF = function(k) {

};
```

TypeScript Solution:

```
/**
 * Problem: Preimage Size of Factorial Zeroes Function
 * Difficulty: Hard
 * 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 preimageSizeFZF(k: number): number {

};
```

C# Solution:

```
/*
 * Problem: Preimage Size of Factorial Zeroes Function
 * Difficulty: Hard
 * 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 int PreimageSizeFZF(int k) {

}
}
```

C Solution:

```
/*
* Problem: Preimage Size of Factorial Zeroes Function
* Difficulty: Hard
* 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
*/

int preimageSizeFZF(int k) {

}
```

Go Solution:

```
// Problem: Preimage Size of Factorial Zeroes Function
// Difficulty: Hard
// 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 preimageSizeFZF(k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun preimageSizeFZF(k: Int): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Preimage Size of Factorial Zeroes Function
// Difficulty: Hard
// 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 preimage_size_fzf(k: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (preimage-size-fzf k)
(-> exact-integer? exact-integer?)
)
```