Problem 1387: Sort Integers by The Power Value

Problem Information

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

Problem Description

The power of an integer

x

is defined as the number of steps needed to transform

x

into

1

using the following steps:

if

x

is even then

x = x / 2

if

x

is odd then

x = 3 * x + 1

For example, the power of

x = 3

is

7

because

3

needs

7

steps to become

1

(

3 --> 10 --> 5 --> 16 --> 8 --> 4 --> 2 --> 1

).

Given three integers

lo

,

hi

and

k

. The task is to sort all integers in the interval

[lo, hi]

by the power value in

ascending order

, if two or more integers have

the same

power value sort them by

ascending order

.

Return the

k

th

integer in the range

[lo, hi]

sorted by the power value.

Notice that for any integer

x

(lo <= x <= hi)

it is

guaranteed

that

x

will transform into

1

using these steps and that the power of

x

is will

fit

in a 32-bit signed integer.

Example 1:

Input:

lo = 12, hi = 15, k = 2

Output:

13

Explanation:

The power of 12 is 9 (12 --> 6 --> 3 --> 10 --> 5 --> 16 --> 8 --> 4 --> 2 --> 1) The power of 13 is 9 The power of 14 is 17 The power of 15 is 17 The interval sorted by the power value [12,13,14,15]. For k = 2 answer is the second element which is 13. Notice that 12 and 13 have the same power value and we sorted them in ascending order. Same for 14 and 15.

Example 2:

Input:

lo = 7, hi = 11, k = 4

Output:

7

Explanation:

The power array corresponding to the interval [7, 8, 9, 10, 11] is [16, 3, 19, 6, 14]. The interval sorted by power is [8, 10, 11, 7, 9]. The fourth number in the sorted array is 7.

Constraints:

1 <= lo <= hi <= 1000

1 <= k <= hi - lo + 1

Code Snippets

C++:

```
class Solution {
public:
int getKth(int lo, int hi, int k) {

}
};
```

Java:

```
class Solution {
public int getKth(int lo, int hi, int k) {

}
}
```

Python3:

```
class Solution:
def getKth(self, lo: int, hi: int, k: int) -> int:
```

Python:

```
class Solution(object):
def getKth(self, lo, hi, k):
"""
:type lo: int
:type hi: int
:type k: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} lo
* @param {number} hi
* @param {number} k
* @return {number}
*/
var getKth = function(lo, hi, k) {

};
```

TypeScript:

```
function getKth(lo: number, hi: number, k: number): number {

};
```

C#:

```
public class Solution {
public int GetKth(int lo, int hi, int k) {

}
}
```

C:

```
int getKth(int lo, int hi, int k) {

}
```

Go:

```
func getKth(lo int, hi int, k int) int {

}
```

Kotlin:

```
class Solution {
fun getKth(lo: Int, hi: Int, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func getKth(_ lo: Int, _ hi: Int, _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn get_kth(lo: i32, hi: i32, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} lo
# @param {Integer} hi
# @param {Integer} k
# @return {Integer}
def get_kth(lo, hi, k)
```

```
end
```

PHP:

```
class Solution {

/**
* @param Integer $lo
* @param Integer $hi
* @param Integer $k
* @return Integer
*/
function getKth($lo, $hi, $k) {

}
}
```

Dart:

```
class Solution {
int getKth(int lo, int hi, int k) {

}
}
```

Scala:

```
object Solution {
def getKth(lo: Int, hi: Int, k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec get_kth(lo :: integer, hi :: integer, k :: integer) :: integer
def get_kth(lo, hi, k) do

end
end
```

Erlang:

```
-spec get_kth(Lo :: integer(), Hi :: integer(), K :: integer()) -> integer().
get_kth(Lo, Hi, K) ->
.
```

Racket:

```
(define/contract (get-kth lo hi k)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Sort Integers by The Power Value
* Difficulty: Medium
* Tags: array, dp, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int getKth(int lo, int hi, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Sort Integers by The Power Value
* Difficulty: Medium
* Tags: array, dp, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int getKth(int lo, int hi, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Sort Integers by The Power Value
Difficulty: Medium
Tags: array, dp, sort

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def getKth(self, lo: int, hi: int, k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def getKth(self, lo, hi, k):
"""
:type lo: int
:type hi: int
:type k: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Sort Integers by The Power Value
* Difficulty: Medium
* Tags: array, dp, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {number} lo
* @param {number} hi
* @param {number} k
* @return {number}
*/
var getKth = function(lo, hi, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Sort Integers by The Power Value
* Difficulty: Medium
* Tags: array, dp, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function getKth(lo: number, hi: number, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Sort Integers by The Power Value
* Difficulty: Medium
* Tags: array, dp, sort
```

```
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int GetKth(int lo, int hi, int k) {

}
}
```

C Solution:

```
/*
* Problem: Sort Integers by The Power Value
* Difficulty: Medium
* Tags: array, dp, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int getKth(int lo, int hi, int k) {

}
```

Go Solution:

```
// Problem: Sort Integers by The Power Value
// Difficulty: Medium
// Tags: array, dp, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func getKth(lo int, hi int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun getKth(lo: Int, hi: Int, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func getKth(_ lo: Int, _ hi: Int, _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Sort Integers by The Power Value
// Difficulty: Medium
// Tags: array, dp, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn get_kth(lo: i32, hi: i32, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} lo
# @param {Integer} hi
# @param {Integer} k
# @return {Integer}
def get_kth(lo, hi, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $lo
* @param Integer $hi
* @param Integer $k
* @return Integer
*/
function getKth($lo, $hi, $k) {

}
}
```

Dart Solution:

```
class Solution {
int getKth(int lo, int hi, int k) {

}
}
```

Scala Solution:

```
object Solution {
def getKth(lo: Int, hi: Int, k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec get_kth(lo :: integer, hi :: integer, k :: integer) :: integer
def get_kth(lo, hi, k) do

end
end
```

Erlang Solution:

```
-spec get_kth(Lo :: integer(), Hi :: integer(), K :: integer()) -> integer().
get_kth(Lo, Hi, K) ->
.
```

Racket Solution:

```
(define/contract (get-kth lo hi k)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```