Problem 3375: Minimum Operations to Make Array Values Equal to K

Problem Information

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

Problem Description

You are given an integer array

nums

and an integer

k

.

An integer

h

is called

valid

if all values in the array that are

strictly greater

than

h

are

identical

.

For example, if

nums = [10, 8, 10, 8]

, a

valid

integer is

h = 9

because all

nums[i] > 9

are equal to 10, but 5 is not a

valid

integer.

You are allowed to perform the following operation on

nums

:

Select an integer

h

that is

valid

for the

current

values in

nums

.

For each index

i

where

nums[i] > h

, set

nums[i]

to

h

.

Return the

minimum

number of operations required to make every element in

nums

equal

to

k

. If it is impossible to make all elements equal to

k

, return -1.

Example 1:

Input:

nums = [5,2,5,4,5], k = 2

Output:

2

Explanation:

The operations can be performed in order using valid integers 4 and then 2.

Example 2:

Input:

nums = [2,1,2], k = 2

Output:

-1

Explanation:

It is impossible to make all the values equal to 2.

Example 3:

Input:

nums = [9,7,5,3], k = 1

Output:

4

Explanation:

The operations can be performed using valid integers in the order 7, 5, 3, and 1.

Constraints:

1 <= nums.length <= 100

1 <= nums[i] <= 100

1 <= k <= 100

Code Snippets

C++:

```
class Solution {
public:
int minOperations(vector<int>& nums, int k) {

}
};
```

Java:

```
class Solution {
public int minOperations(int[] nums, int k) {
```

```
}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function minOperations(nums: number[], k: number): number {

};
```

C#:

```
public class Solution {
public int MinOperations(int[] nums, int k) {

}
```

```
}
```

C:

```
int minOperations(int* nums, int numsSize, int k) {

}
```

Go:

```
func minOperations(nums []int, k int) int {

}
```

Kotlin:

```
class Solution {
fun minOperations(nums: IntArray, k: Int): Int {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn min_operations(nums: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer} k
```

```
# @return {Integer}
def min_operations(nums, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $k
* @return Integer
*/
function minOperations($nums, $k) {

}
}
```

Dart:

```
class Solution {
int minOperations(List<int> nums, int k) {

}
}
```

Scala:

```
object Solution {
def minOperations(nums: Array[Int], k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_operations(nums :: [integer], k :: integer) :: integer
def min_operations(nums, k) do

end
```

```
end
```

Erlang:

```
-spec min_operations(Nums :: [integer()], K :: integer()) -> integer().
min_operations(Nums, K) ->
.
```

Racket:

```
(define/contract (min-operations nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Operations to Make Array Values Equal to K
* Difficulty: Easy
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int minOperations(vector<int>& nums, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Operations to Make Array Values Equal to K
* Difficulty: Easy
```

```
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int minOperations(int[] nums, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Operations to Make Array Values Equal to K
Difficulty: Easy
Tags: array, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def minOperations(self, nums: List[int], k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Operations to Make Array Values Equal to K
* Difficulty: Easy
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {number[]} nums
* @param {number} k
* @return {number}
*/
var minOperations = function(nums, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Operations to Make Array Values Equal to K
* Difficulty: Easy
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function minOperations(nums: number[], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Operations to Make Array Values Equal to K
* Difficulty: Easy
* Tags: array, hash
*
```

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

public class Solution {
public int MinOperations(int[] nums, int k) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Operations to Make Array Values Equal to K
* Difficulty: Easy
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int minOperations(int* nums, int numsSize, int k) {

}
```

Go Solution:

```
// Problem: Minimum Operations to Make Array Values Equal to K
// Difficulty: Easy
// Tags: array, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func minOperations(nums []int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minOperations(nums: IntArray, k: Int): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Operations to Make Array Values Equal to K
// Difficulty: Easy
// Tags: array, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn min_operations(nums: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer} k
# @return {Integer}
def min_operations(nums, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $k
* @return Integer
*/
function minOperations($nums, $k) {

}
}
```

Dart Solution:

```
class Solution {
int minOperations(List<int> nums, int k) {

}
}
```

Scala Solution:

```
object Solution {
def minOperations(nums: Array[Int], k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_operations(nums :: [integer], k :: integer) :: integer
def min_operations(nums, k) do

end
end
```

Erlang Solution:

```
-spec min_operations(Nums :: [integer()], K :: integer()) -> integer().
min_operations(Nums, K) ->
.
```

Racket Solution:

```
(define/contract (min-operations nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```