Problem 3159: Find Occurrences of an Element in an Array

Problem Information

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

Problem Description

You are given an integer array

nums

, an integer array

queries

, and an integer

x

.

For each

queries[i]

, you need to find the index of the

queries[i]

th

occurrence of

x

in the

nums

array. If there are fewer than

queries[i]

occurrences of

x

, the answer should be -1 for that query.

Return an integer array

answer

containing the answers to all queries.

Example 1:

Input:

nums = [1,3,1,7], queries = [1,3,2,4], x = 1

Output:

[0,-1,2,-1]

Explanation:

For the 1

st

query, the first occurrence of 1 is at index 0.

For the 2

nd

query, there are only two occurrences of 1 in

nums

, so the answer is -1.

For the 3

rd

query, the second occurrence of 1 is at index 2.

For the 4

th

query, there are only two occurrences of 1 in

nums

, so the answer is -1.

Example 2:

Input:

nums = [1,2,3], queries = [10], x = 5

Output:

[-1]

Explanation:

For the 1

st

query, 5 doesn't exist in

nums

, so the answer is -1.

Constraints:

1 <= nums.length, queries.length <= 10

5

1 <= queries[i] <= 10

5

1 <= nums[i], x <= 10

4

Code Snippets

C++:

```
class Solution {
public:
vector<int> occurrencesOfElement(vector<int>& nums, vector<int>& queries, int
x) {

}
};
```

Java:

```
class Solution {
public int[] occurrencesOfElement(int[] nums, int[] queries, int x) {

}
}
```

Python3:

```
class Solution:
def occurrencesOfElement(self, nums: List[int], queries: List[int], x: int)
-> List[int]:
```

Python:

```
class Solution(object):
def occurrencesOfElement(self, nums, queries, x):
"""
:type nums: List[int]
:type queries: List[int]
:type x: int
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {number[]} nums
* @param {number[]} queries
* @param {number} x
* @return {number[]}
*/
var occurrencesOfElement = function(nums, queries, x) {

};
```

TypeScript:

```
function occurrencesOfElement(nums: number[], queries: number[], x: number):
number[] {

};
```

C#:

```
public class Solution {
public int[] OccurrencesOfElement(int[] nums, int[] queries, int x) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* occurrencesOfElement(int* nums, int numsSize, int* queries, int
queriesSize, int x, int* returnSize) {

}
```

Go:

```
func occurrencesOfElement(nums []int, queries []int, x int) []int {

}
```

Kotlin:

```
class Solution {
fun occurrencesOfElement(nums: IntArray, queries: IntArray, x: Int): IntArray
{

}
}
```

Swift:

```
class Solution {
func occurrencesOfElement(_ nums: [Int], _ queries: [Int], _ x: Int) -> [Int]
{

}
}
```

Rust:

```
impl Solution {
pub fn occurrences_of_element(nums: Vec<i32>, queries: Vec<i32>, x: i32) ->
Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer[]} queries
# @param {Integer} x
# @return {Integer[]}
def occurrences_of_element(nums, queries, x)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer[] $queries
* @param Integer $x
* @return Integer[]
*/
function occurrencesOfElement($nums, $queries, $x) {

}
}
```

Dart:

```
class Solution {
List<int> occurrencesOfElement(List<int> nums, List<int> queries, int x) {

}
}
```

Scala:

```
object Solution {
def occurrencesOfElement(nums: Array[Int], queries: Array[Int], x: Int):
Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec occurrences_of_element(nums :: [integer], queries :: [integer], x ::
integer) :: [integer]
def occurrences_of_element(nums, queries, x) do

end
end
```

Erlang:

```
-spec occurrences_of_element(Nums :: [integer()], Queries :: [integer()], X
:: integer()) -> [integer()].
occurrences_of_element(Nums, Queries, X) ->
.
```

Racket:

```
(define/contract (occurrences-of-element nums queries x)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer? (listof
exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Occurrences of an Element in an Array
* Difficulty: Medium
* 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:
vector<int> occurrencesOfElement(vector<int>& nums, vector<int>& queries, int
x) {

}
};
```

Java Solution:

```
/**
* Problem: Find Occurrences of an Element in an Array
* Difficulty: Medium
* 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[] occurrencesOfElement(int[] nums, int[] queries, int x) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Occurrences of an Element in an Array
Difficulty: Medium
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 occurrencesOfElement(self, nums: List[int], queries: List[int], x: int)
-> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def occurrencesOfElement(self, nums, queries, x):
"""
:type nums: List[int]
:type queries: List[int]
:type x: int
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Find Occurrences of an Element in an Array
* Difficulty: Medium
* 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[]} queries
* @param {number} x
* @return {number[]}
*/
var occurrencesOfElement = function(nums, queries, x) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Occurrences of an Element in an Array
* Difficulty: Medium
* 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 occurrencesOfElement(nums: number[], queries: number[], x: number):
number[] {

};
```

C# Solution:

```
/*
* Problem: Find Occurrences of an Element in an Array
* Difficulty: Medium
* 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[] OccurrencesOfElement(int[] nums, int[] queries, int x) {

}
}
```

C Solution:

```
/*
* Problem: Find Occurrences of an Element in an Array
* Difficulty: Medium
* 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
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* occurrencesOfElement(int* nums, int numsSize, int* queries, int
queriesSize, int x, int* returnSize) {

}
```

Go Solution:

```
// Problem: Find Occurrences of an Element in an Array
// Difficulty: Medium
// 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 occurrencesOfElement(nums []int, queries []int, x int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun occurrencesOfElement(nums: IntArray, queries: IntArray, x: Int): IntArray
{

}
}
```

Swift Solution:

```
class Solution {
func occurrencesOfElement(_ nums: [Int], _ queries: [Int], _ x: Int) -> [Int]
{

}
}
```

Rust Solution:

```
// Problem: Find Occurrences of an Element in an Array
// Difficulty: Medium
// 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 occurrences_of_element(nums: Vec<i32>, queries: Vec<i32>, x: i32) ->
Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer[]} queries
# @param {Integer} x
# @return {Integer[]}
def occurrences_of_element(nums, queries, x)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer[] $queries
* @param Integer $x
* @return Integer[]
*/
function occurrencesOfElement($nums, $queries, $x) {

}
}
```

Dart Solution:

```
class Solution {
List<int> occurrencesOfElement(List<int> nums, List<int> queries, int x) {

}
}
```

Scala Solution:

```
object Solution {
def occurrencesOfElement(nums: Array[Int], queries: Array[Int], x: Int):
Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec occurrences_of_element(nums :: [integer], queries :: [integer], x ::
integer) :: [integer]
def occurrences_of_element(nums, queries, x) do

end
end
```

Erlang Solution:

```
-spec occurrences_of_element(Nums :: [integer()], Queries :: [integer()], X
:: integer()) -> [integer()].
occurrences_of_element(Nums, Queries, X) ->
.
```

Racket Solution:

```
(define/contract (occurrences-of-element nums queries x)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer? (listof
exact-integer?))
)
```