Problem 2519: Count the Number of K-Big Indices

Problem Information

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

Problem Description

You are given a

0-indexed

integer array

nums

and a positive integer

k

.

We call an index

i

k-big

if the following conditions are satisfied:

There exist at least

k

different indices

idx1

such that

idx1 < i

and

nums[idx1] < nums[i]

.

There exist at least

k

different indices

idx2

such that

idx2 > i

and

nums[idx2] < nums[i]

.

Return

the number of k-big indices

.

Example 1:

Input:

nums = [2,3,6,5,2,3], k = 2

Output:

2

Explanation:

There are only two 2-big indices in nums: - i = 2 --> There are two valid idx1: 0 and 1. There are three valid idx2: 2, 3, and 4. - i = 3 --> There are two valid idx1: 0 and 1. There are two valid idx2: 3 and 4.

Example 2:

Input:

nums = [1,1,1], k = 3

Output:

0

Explanation:

There are no 3-big indices in nums.

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i], k <= nums.length

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

```
};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
```

```
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (k-big-indices nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count the Number of K-Big Indices
* Difficulty: Hard
* Tags: array, tree, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

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

}
};
```

Java Solution:

```
/**
* Problem: Count the Number of K-Big Indices
* Difficulty: Hard
* Tags: array, tree, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
}
```

Python3 Solution:

```
"""
Problem: Count the Number of K-Big Indices
Difficulty: Hard
Tags: array, tree, sort, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count the Number of K-Big Indices
* Difficulty: Hard
* Tags: array, tree, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Count the Number of K-Big Indices
* Difficulty: Hard
* Tags: array, tree, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

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

};
```

C# Solution:

```
/*
* Problem: Count the Number of K-Big Indices
* Difficulty: Hard
* Tags: array, tree, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
}
```

C Solution:

```
/*
* Problem: Count the Number of K-Big Indices
* Difficulty: Hard
* Tags: array, tree, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
```

Go Solution:

```
// Problem: Count the Number of K-Big Indices
// Difficulty: Hard
// Tags: array, tree, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Count the Number of K-Big Indices
// Difficulty: Hard
// Tags: array, tree, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

}
```

```
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (k-big-indices nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```