Problem 1539: Kth Missing Positive Number

Problem Information

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

Problem Description

Given an array

arr

of positive integers sorted in a

strictly increasing order

, and an integer

k

.

Return

the

k

th

positive

integer that is

missing

from this array.

Example 1:

Input:

arr = [2,3,4,7,11], k = 5

Output:

9

Explanation:

The missing positive integers are [1,5,6,8,9,10,12,13,...]. The 5

th

missing positive integer is 9.

Example 2:

Input:

arr = [1,2,3,4], k = 2

Output:

6

Explanation:

The missing positive integers are [5,6,7,...]. The 2

nd

missing positive integer is 6.

Constraints:

1 <= arr.length <= 1000

1 <= arr[i] <= 1000

1 <= k <= 1000

arr[i] < arr[j]

for

1 <= i < j <= arr.length

Follow up:

Could you solve this problem in less than O(n) complexity?

Code Snippets

C++:

```
class Solution {
public:
int findKthPositive(vector<int>& arr, int k) {

}
};
```

Java:

```
class Solution {
public int findKthPositive(int[] arr, int k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function findKthPositive(arr: number[], k: number): number {

};
```

C#:

```
public class Solution {
public int FindKthPositive(int[] arr, int k) {

}
}
```

C:

```
int findKthPositive(int* arr, int arrSize, int k) {

}
```

Go:

```
func findKthPositive(arr []int, k int) int {

}
```

Kotlin:

```
class Solution {
fun findKthPositive(arr: IntArray, k: Int): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

```
# @param {Integer[]} arr
# @param {Integer} k
# @return {Integer}
def find_kth_positive(arr, k)

end
```

PHP:

```
class Solution {
```

```
/**
* @param Integer[] $arr
* @param Integer $k
* @return Integer
*/
function findKthPositive($arr, $k) {

}
}
```

Dart:

```
class Solution {
int findKthPositive(List<int> arr, int k) {

}
}
```

Scala:

```
object Solution {
def findKthPositive(arr: Array[Int], k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec find_kth_positive(arr :: [integer], k :: integer) :: integer
def find_kth_positive(arr, k) do

end
end
```

Erlang:

```
-spec find_kth_positive(Arr :: [integer()], K :: integer()) -> integer().
find_kth_positive(Arr, K) ->
.
```

Racket:

```
(define/contract (find-kth-positive arr k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Kth Missing Positive Number
* Difficulty: Easy
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int findKthPositive(vector<int>& arr, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Kth Missing Positive Number
* Difficulty: Easy
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int findKthPositive(int[] arr, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Kth Missing Positive Number
Difficulty: Easy
Tags: array, sort, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Kth Missing Positive Number
* Difficulty: Easy
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

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

};
```

TypeScript Solution:

```
/**
* Problem: Kth Missing Positive Number
* Difficulty: Easy
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function findKthPositive(arr: number[], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Kth Missing Positive Number
* Difficulty: Easy
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int FindKthPositive(int[] arr, int k) {

}
}
```

C Solution:

```
/*
* Problem: Kth Missing Positive Number
* Difficulty: Easy
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int findKthPositive(int* arr, int arrSize, int k) {

}
```

Go Solution:

```
// Problem: Kth Missing Positive Number
// Difficulty: Easy
// Tags: array, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func findKthPositive(arr []int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun findKthPositive(arr: IntArray, k: Int): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Kth Missing Positive Number
// Difficulty: Easy
// Tags: array, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn find_kth_positive(arr: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} arr
# @param {Integer} k
# @return {Integer}
def find_kth_positive(arr, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @param Integer $k
* @return Integer
*/
function findKthPositive($arr, $k) {
}
}
```

Dart Solution:

```
class Solution {
int findKthPositive(List<int> arr, int k) {

}
}
```

Scala Solution:

```
object Solution {
def findKthPositive(arr: Array[Int], k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec find_kth_positive(arr :: [integer], k :: integer) :: integer
def find_kth_positive(arr, k) do

end
end
```

Erlang Solution:

```
-spec find_kth_positive(Arr :: [integer()], K :: integer()) -> integer().
find_kth_positive(Arr, K) ->
.
```

Racket Solution:

```
(define/contract (find-kth-positive arr k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```