Problem 275: H-Index II

Problem Information

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

Problem Description

Given an array of integers

citations

where

citations[i]

is the number of citations a researcher received for their

i

th

paper and

citations

is sorted in

non-descending order

, return

the researcher's h-index

.

According to the

definition of h-index on Wikipedia

: The h-index is defined as the maximum value of

h

such that the given researcher has published at least

h

papers that have each been cited at least

h

times.

You must write an algorithm that runs in logarithmic time.

Example 1:

Input:

citations = [0,1,3,5,6]

Output:

3

Explanation:

[0,1,3,5,6] means the researcher has 5 papers in total and each of them had received 0, 1, 3, 5, 6 citations respectively. Since the researcher has 3 papers with at least 3 citations each and the remaining two with no more than 3 citations each, their h-index is 3.

Example 2:

Input:

citations = [1,2,100]

Output:

2

Constraints:

n == citations.length

1 <= n <= 10

5

0 <= citations[i] <= 1000

citations

is sorted in

ascending order

.

Code Snippets

C++:

```
class Solution {
public:
int hIndex(vector<int>& citations) {

}
};
```

Java:

```
class Solution {
public int hIndex(int[] citations) {

}
}
```

Python3:

```
class Solution:
def hIndex(self, citations: List[int]) -> int:
```

Python:

```
class Solution(object):
def hIndex(self, citations):
"""
:type citations: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} citations
* @return {number}
*/
var hIndex = function(citations) {

};
```

TypeScript:

```
function hIndex(citations: number[]): number {

};
```

C#:

```
public class Solution {
public int HIndex(int[] citations) {

}
}
```

C:

```
int hIndex(int* citations, int citationsSize) {

}
```

Go:

```
func hIndex(citations []int) int {

}
```

Kotlin:

```
class Solution {
fun hIndex(citations: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func hIndex(_ citations: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn h_index(citations: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} citations
# @return {Integer}
def h_index(citations)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $citations
* @return Integer
*/
function hIndex($citations) {

}
}
```

Dart:

```
class Solution {
int hIndex(List<int> citations) {

}
}
```

Scala:

```
object Solution {
def hIndex(citations: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec h_index(citations :: [integer]) :: integer
def h_index(citations) do

end
end
```

Erlang:

```
-spec h_index(Citations :: [integer()]) -> integer().
h_index(Citations) ->
.
```

Racket:

```
(define/contract (h-index citations)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: H-Index II
* Difficulty: Medium
* 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 hIndex(vector<int>& citations) {

}
};
```

Java Solution:

```
/**
* Problem: H-Index II
* Difficulty: Medium
* 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 hIndex(int[] citations) {
}
}
```

Python3 Solution:

```
"""
Problem: H-Index II
Difficulty: Medium
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 hIndex(self, citations: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def hIndex(self, citations):
"""
:type citations: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: H-Index II
* Difficulty: Medium
* 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[]} citations
* @return {number}
*/
var hIndex = function(citations) {

};
```

TypeScript Solution:

```
/**
* Problem: H-Index II
* Difficulty: Medium
* 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 hIndex(citations: number[]): number {

};
```

C# Solution:

```
/*
* Problem: H-Index II
* Difficulty: Medium
* 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 HIndex(int[] citations) {

}

}
```

C Solution:

```
/*
* Problem: H-Index II
* Difficulty: Medium
* 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 hIndex(int* citations, int citationsSize) {

}
```

Go Solution:

```
// Problem: H-Index II
// Difficulty: Medium
// 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 hIndex(citations []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun hIndex(citations: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func hIndex(_ citations: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: H-Index II
// Difficulty: Medium
// 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 h_index(citations: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} citations
# @return {Integer}
def h_index(citations)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $citations
* @return Integer
*/
function hIndex($citations) {

}
}
```

Dart Solution:

```
class Solution {
int hIndex(List<int> citations) {

}
}
```

Scala Solution:

```
object Solution {
def hIndex(citations: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec h_index(citations :: [integer]) :: integer
def h_index(citations) do

end
end
```

Erlang Solution:

```
-spec h_index(Citations :: [integer()]) -> integer().
h_index(Citations) ->
.
```

Racket Solution:

```
(define/contract (h-index citations)
(-> (listof exact-integer?) exact-integer?)
)
```