275. H-Index II

July 18, 2019 | 14.1K views

Average Rating: 3.93 (14 votes)

Given an array of citations **sorted in ascending order** (each citation is a non-negative integer) of a researcher, write a function to compute the researcher's h-index.

According to the definition of h-index on Wikipedia: "A scientist has index *h* if *h* of his/her *N* papers have **at least** *h* citations each, and the other *N − h* papers have **no more than** *h* citations each."

Example:

```
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, her h-index is 3.
```

Note:

If there are several possible values for *h*, the maximum one is taken as the h-index.

Follow up:

- This is a follow up problem to H-Index, where `citations` is now guaranteed to be sorted in ascending order.
- Could you solve it in logarithmic time complexity?

Solution

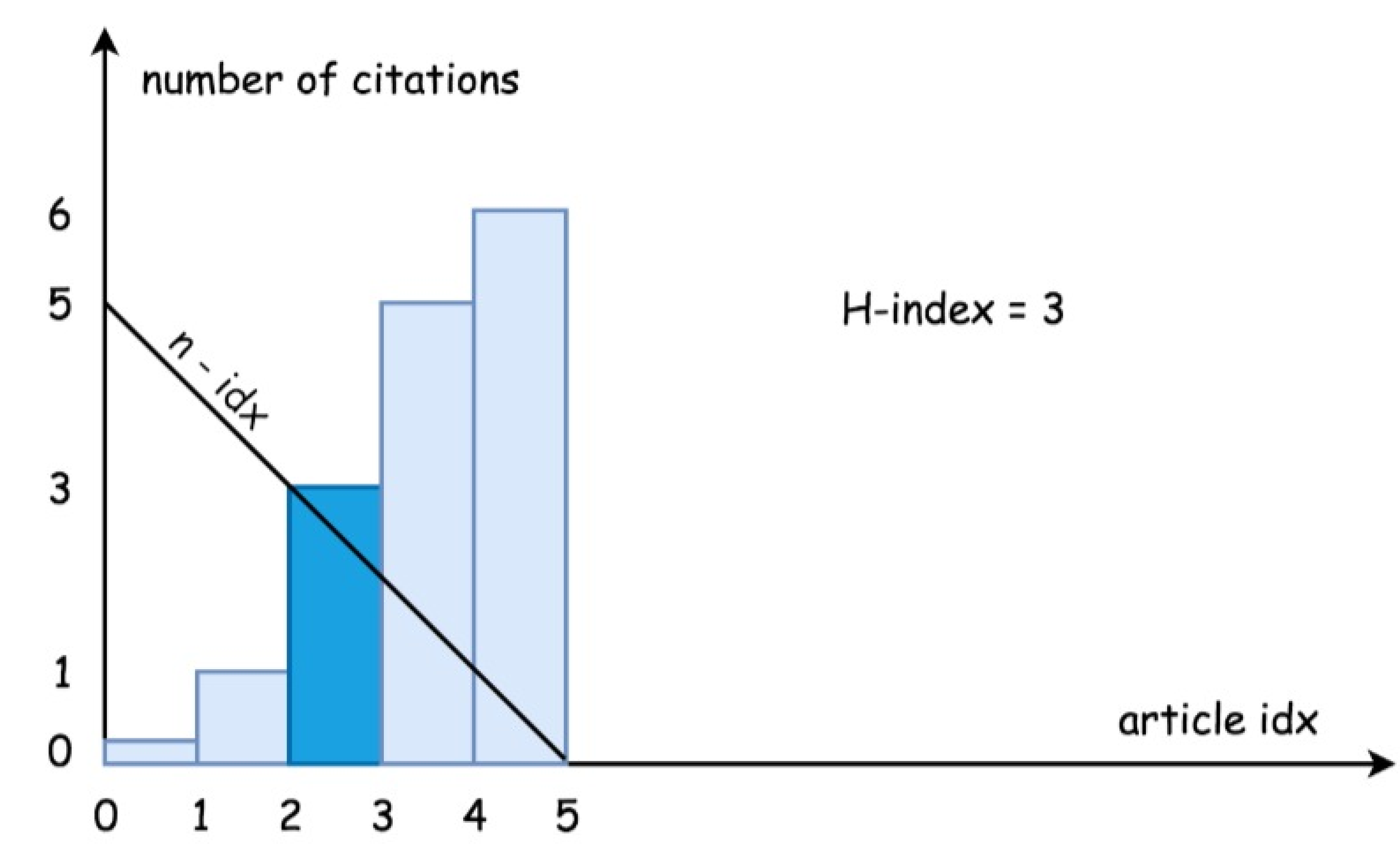
Approach 1: Linear search, O(k) time

Intuition

Thanks to the fact that the list of citation numbers is sorted in the ascending order, one could solve the problem in a single pass of iteration.

Let's consider an article whose citation number `c` is index at `i`, *i.e* `c = citations[i]`. We would know that the number of articles whose citation number is higher than `c` would be `n - i - 1`.
And together with the current article, there are `n - i` articles that are cited at least `c` times.

Given the definition of H-Index, we just need to find the first article at `i` whose citation number `c = citation[i]` is greater or equal to `n - i`, *i.e.* `c >= n - i`. As we know that all the articles following `i` would be cited at least `c` times, so in total *there are* `n - i` *articles that are cited at least* `c` *times*. As a result, according to the definition, the H-Index of the author should be `n - i`.



Following the above intuition, it is straightforward to implement the algorithm. We give some examples in the following.

Implementation

Java Python Copy

```python
class Solution:
    def hIndex(self, citations):
        """
        :type citations: List[int]
        :rtype: int
        """
        n = len(citations)
        for idx, c in enumerate(citations):
            if c >= n - idx:
                return n - idx
        return 0
```

Complexity Analysis

- Time complexity : $\mathcal{O}(N)$ where N is the length of the input list, since in the worse case we would iterate the entire list.
- Space complexity : $\mathcal{O}(1)$, it's a constant space solution.

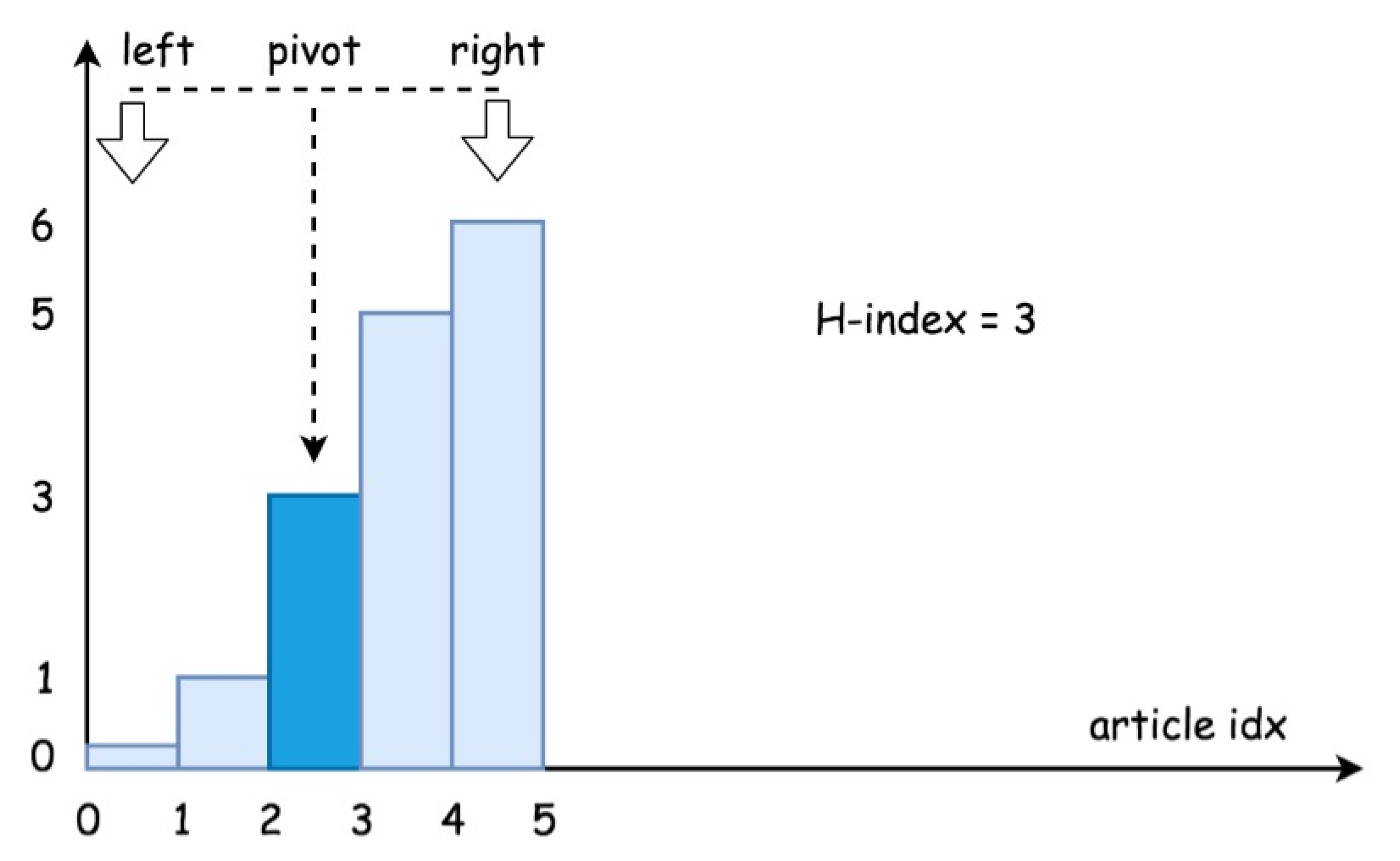
Approach 2: Binary Search, O(log N) time

Intuition

Following in the intuition we elaborated in the Approach 1, the problem is actually boiled down to the following task:

> Given a sorted list `citations` of size `n` in ascending order, one is asked to find the *first* number `citations[i]` which meets the constraint of `citations[i] >= n - i`.

With the above formulation of the problem, it becomes clear that one could apply the *binary search* algorithm to solve the problem. In binary search algorithm, we recursively reduce the searching scope into half, which leads to a more optimal $\mathcal{O}(\log N)$ time complexity comparing to the $\mathcal{O}(N)$ of the linear search.



Algorithm

First we pick a pivot element that is in the middle of the list, *i.e.* `citations[pivot]`, which would divide the original list into two sublists: `citations[0 : pivot - 1]` and `citations[pivot + 1 : n]`.

Then comparing between the values of `n - pivot` and `citations[pivot]` element, our binary search algorithm breaks down to the following 3 cases:

- `citations[pivot] == n - pivot`: We found the desired element !
- `citations[pivot] < n - pivot`: Since the desired element should be greater or equal to `n - pivot`, we then further look into the sublist on the right hand side, *i.e.* `citations[pivot + 1 : n]`.
- `citations[pivot] > n - pivot`: We should look into the sublist on the left hand side, *i.e.* `citations[0 : pivot-1]`.

A minor difference to the typical binary search algorithm, is that in this case we return the value of `n - pivot` as the result, rather than the value of the desired element.

Implementation

Java Python Copy

```python
class Solution:
    def hIndex(self, citations):
        """
        :type citations: List[int]
        :rtype: int
        """
        n = len(citations)
        left, right = 0, n - 1
        while left <= right:
            pivot = left + (right - left) // 2
            if citations[pivot] == n - pivot:
                return n - pivot
            elif citations[pivot] < n - pivot:
                left = pivot + 1
            else:
                right = pivot - 1

        return n - left
```

Complexity Analysis

- Time complexity : $\mathcal{O}(\log N)$ since we apply binary search algorithm here.
- Space complexity : $\mathcal{O}(1)$, it's a constant space solution.

Rate this article:

Previous Next

Comments: 14

Sort By

Type comment here... (Markdown is supported)

Preview Post

diiikoo ★ 19 August 11, 2019 7:41 AM
For the Binary search solution, I think the variable "idx" is of no use.
15 Share Reply

sunxiaohua ★ 3 October 21, 2019 7:49 AM
Why there has a condition: If there are several possible values for h, the maximum one is taken as the h-index?
Don't think multiple index could satisfied this.
3 Share Reply
SHOW 2 REPLIES

ping_pong ★ 826 June 19, 2020 9:49 AM
I think as per the question condition should be `c[i] == n - i` the problem is poorly phrased.
2 Share Reply

zzznotsomuch ★ 54 June 19, 2020 9:11 AM
In binary search I wonder how to decide between `while(lo < hi)` and `while(lo <= hi)`. I seem to always get those wrong. any suggestions?
2 Share Reply
SHOW 2 REPLIES

lenchen1112 ★ 1005 December 3, 2019 4:52 PM
For binary search approach, use pattern two will be easier to understand.

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

Read More
3 Share Reply
SHOW 1 REPLY

damhajan92 ★ 0 3 days ago
question description is either confusing or wrong. 😕
0 Share Reply

jaykpatel1996 ★ 1 July 10, 2020 10:41 AM
I think this definition of H index from Wikipedia makes this question a lot easier.
"The h-index is defined as the maximum value of h such that the given author/journal has published h papers that have each been cited at least h times."
0 Share Reply

jaykpatel1996 ★ 1 July 5, 2020 11:01 AM
This question should be marked as Hard.
0 Share Reply

zzznotsomuch ★ 54 June 19, 2020 9:10 AM Report
In Java Solution, can someone explain why are we returning `n-left` from outside the loop?
0 Share Reply

xl549 ★ 0 June 6, 2020 9:18 AM Report
for the approach 2, i was wondering why we "return len - start" rather than "return len - end"
0 Share Reply
SHOW 1 REPLY

‹ 1 2 ›