

TASK REPORT
ALGORITHMS AND DATA STRUCTURE
Search Algorithms



Created by:
Onic Agustino
L200234275
X
Date : April 29, 2025

INFORMATICS ENGINEERING
FACULTY OF COMMUNICATION AND INFORMATICS
UNIVERSITAS MUHAMMADIYAH SURAKARTA
2024/2025

Leetcode

1. Problem 1: Kth Missing Positive Number

- **Problem Explanation**

You are given a sorted array `arr` of positive integers and an integer `k`.

You need to find the `k`-th positive integer that is missing from the array.

Input:

- `arr` (list of positive integers, sorted)
- `k` (positive integer)

Output:

- Return the `k`-th missing positive number.

- **Solution Approach**

We use **binary search** to optimize the search.

For each `arr[mid]`, the number of missing numbers before it is `arr[mid] - (mid + 1)`.

We perform binary search to locate the smallest index where the number of missing integers is at least `k`.

If not found directly, the final answer will be `left + k`.

- **Code Implementation**



```
1 class Solution:
2     def findKthPositive(self, arr, k):
3         kiri, kanan = 0, len(arr) - 1
4
5         while kiri <= kanan:
6             mid = (kiri + kanan) // 2
7             missing = arr[mid] - (mid + 1)
8             if missing < k:
9                 kiri = mid + 1
10            else:
11                kanan = mid - 1
12
13        return kiri + k
14
```

- **Output**

Testcase

Test Result

Accepted Runtime: 0 ms

Case 1

Case 2

Input

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

k =
5

Output

9

Expected

9

- **Screenshot Submission**

Kth Missing Positive Number

Binary Search - LeetCode

leetcode.com/problems/kth-missing-positive-number/submissions/1620698042/

Problem List

Run

Submit

Premium

Description

Accepted

Editorial

Solutions

Submissions

All Submissions

Accepted 87 / 87 testcases passed

56nSINT1m submitted at Apr 29, 2025 09:20

Editorial

Solution

Runtime

0 ms Beat: 100.00%

Memory

12.66 MB Beat: 12.94%

Analyze Complexity

75%

50%

25%

0%

16ms

36ms

76ms

163ms

129ms

113ms

103ms

76ms

36ms

16ms

Code

Python

```
class Solution:
    def findKthPositive(self, arr, k):
        kiri, kanan = 0, len(arr) - 1

        while kiri <= kanan:
            mid = (kiri + kanan) // 2
            missing = arr[mid] - (mid + 1)
            if missing < k:
                kiri = mid + 1
            else:
                kanan = mid - 1

        return kiri + k
```

Testcase

Test Result

Accepted Runtime: 0 ms

Case 1

Case 2

Input

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

2. Problem 2: Binary Search

- **Problem Explanation**

You are given a sorted array `nums` and a target value.

You must return the index where the target value is found. If not found, return -1.

Input:

- `nums` (list of integers, sorted in ascending order)
- `target` (integer)

Output:

- Index of the target if found, else -1.

- **Solution Approach**

We use **binary search** which has a time complexity of **$O(\log n)$** .

At each step, compare the target with the middle element and adjust the search boundaries (left and right) accordingly.

- **Code Implementation**



```
</> Code
Python Auto
1 class Solution:
2     def search(self, nums, target):
3         left, right = 0, len(nums) - 1
4
5         while left <= right:
6             mid = (left + right) // 2
7             if nums[mid] == target:
8                 return mid
9             elif nums[mid] < target:
10                left = mid + 1
11            else:
12                right = mid - 1
13
14        return -1
15
```

- **Output**

Testcase

Test Result

Accepted Runtime: 0 ms

Case 1

Case 2

Input

nums =
[-1,0,3,5,9,12]

target =
9

Output

4

Expected

4

- **Screenshot Submission**

Binary Search - LeetCode

leetcode.com/problems/binary-search/submissions/1620699322/

Problem List

Run

Submit

Premium

Description

Accepted

Editorial

Solutions

Submissions

All Submissions

Accepted 47 / 47 testcases passed

56sINT1T1m submitted at Apr 29, 2025 09:23

Editorial

Solution

Runtime

0 ms Beat: 100.00%

Memory

13.52 MB Beat: 32.81%

Analyze Complexity

100%

50%

0%

1ms

2ms

3ms

4ms

5ms

6ms

7ms

8ms

9ms

10ms

Code | Python

class Solution:
def search(self, nums, target):
left, right = 0, len(nums) - 1

while left <= right:
mid = (left + right) // 2
if nums[mid] == target:
return mid
View more

Testcase

Test Result

Accepted Runtime: 0 ms

Case 1

Case 2

Input

nums =
[-1,0,3,5,9,12]

target =
9

Output

4

Expected

4

Contribute a testcase

Berita untuk Anda

Legislator: UU S...

Search

ENG

09:23

29/04/2025