33. Search in Rotated Sorted Array (/problems/search-in-rotated-sorted-array/)

Jan. 10, 2019 | 43.2K views

Average Rating: 3.65 (26 votes)

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand.

(i.e., `[0,1,2,4,5,6,7]` might become `[4,5,6,7,0,1,2]`).

You are given a target value to search. If found in the array return its index, otherwise return `-1`.

You may assume no duplicate exists in the array.

Your algorithm's runtime complexity must be in the order of *O*(log *n*).

Example 1:

```
Input: nums = [4,5,6,7,0,1,2], target = 0
Output: 4
```

Example 2:

```
Input: nums = [4,5,6,7,0,1,2], target = 3
Output: -1
```

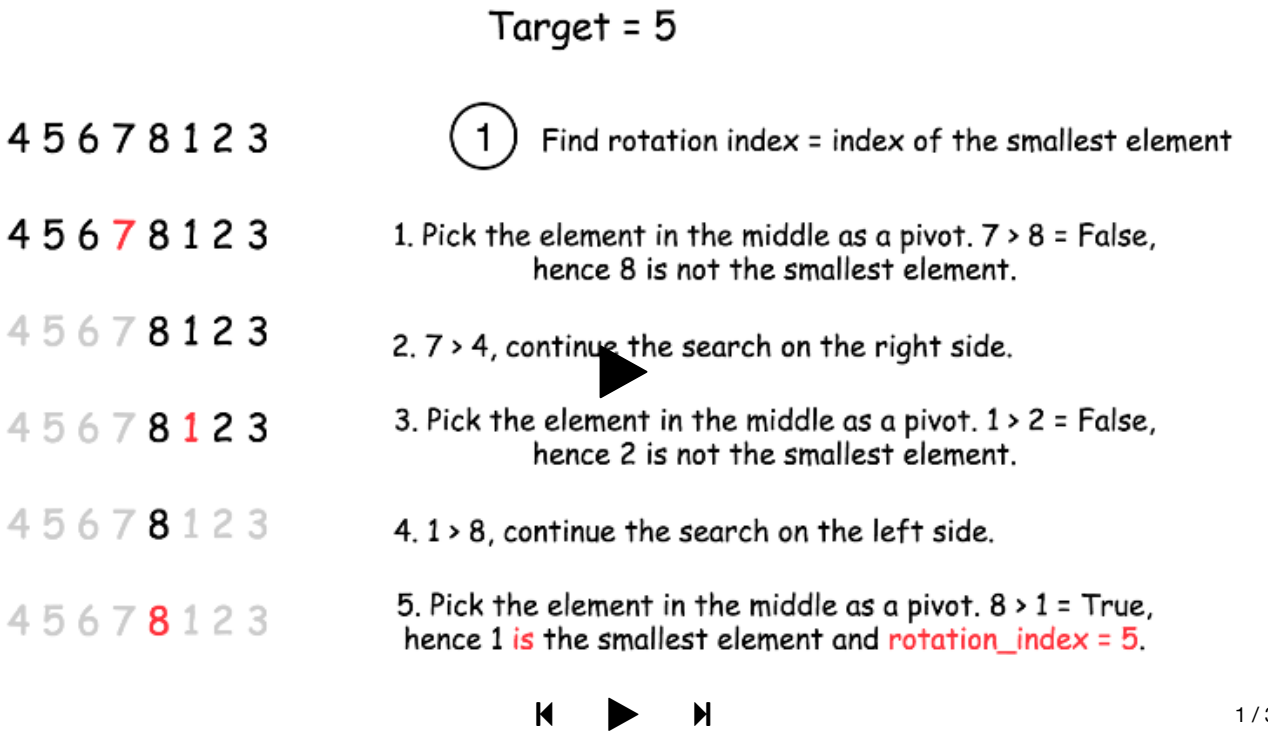
Solution

Approach 1: Binary search

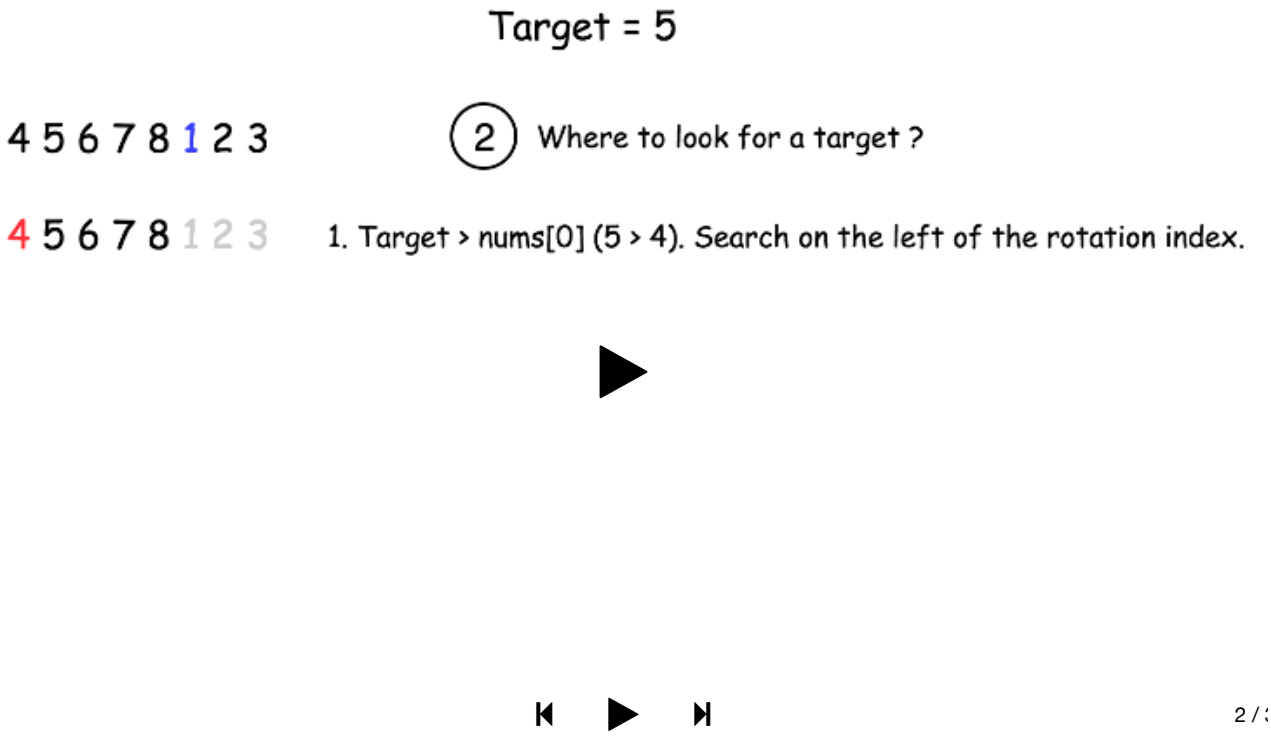
The problem is to implement a search in $\mathcal{O}(\log(N))$ time that gives an idea to use a binary search.

The algorithm is quite straightforward :

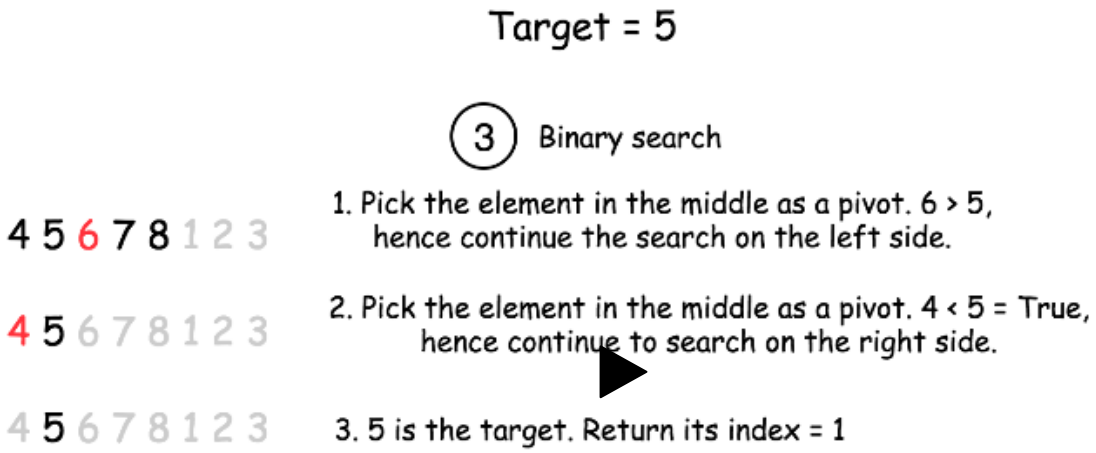
- Find a rotation index `rotation_index`, *i.e.* index of the smallest element in the array. Binary search works just perfect here.
- `rotation_index` splits array in two parts. Compare `nums[0]` and `target` to identify in which part one has to look for `target`.
- Perform a binary search in the chosen part of the array.



- Find a rotation index `rotation_index`, *i.e.* index of the smallest element in the array. Binary search works just perfect here.
- `rotation_index` splits array in two parts. Compare `nums[0]` and `target` to identify in which part one has to look for `target`.
- Perform a binary search in the chosen part of the array.



- Find a rotation index `rotation_index`, *i.e.* index of the smallest element in the array. Binary search works just perfect here.
- `rotation_index` splits array in two parts. Compare `nums[0]` and `target` to identify in which part one has to look for `target`.
- Perform a binary search in the chosen part of the array.



3 / 3

Java | Python

Articles > 33. Search in Rotated Sorted Array

Copy

```
class Solution:
    def search(self, nums, target):
        """
        :type nums: List[int]
        :type target: int
        :rtype: int
        """
        def find_rotate_index(left, right):
            if nums[left] < nums[right]:
                return 0

            while left <= right:
                pivot = (left + right) // 2
                if nums[pivot] > nums[pivot + 1]:
                    return pivot + 1
                else:
                    if nums[pivot] < nums[left]:
                        right = pivot - 1
                    else:
                        left = pivot + 1

        def search(left, right):
            """
            Binary search
            """
            while left <= right:
                pivot = (left + right) // 2
```

Complexity Analysis

- Time complexity : $\mathcal{O}(\log(N))$.
- Space complexity : $\mathcal{O}(1)$.

Approach 2: One pass

Instead of two passes, all this could be done in one pass. Kudos for this solution go to @haoyangfan (https://leetcode.com/haoyangfan/).

Algorithm

- Initiate start to be equal to 0, and end to be equal to `n - 1`.
- Perform standard binary search. While `start <= end`:
    - Take an index in the middle `mid` as a pivot.
    - If `nums[mid] == target`, the job is done, return `mid`.
    - Now there could be two situations:
        - Pivot element is larger than the first element in the array, i.e. the part of array from the first element to the pivot one is non-rotated.

- If the target is in that non-rotated part as well: go left: `end = mid - 1`.
- Otherwise: go right: `start = mid + 1`.

- Pivot element is smaller than the first element of the array, i.e. the rotation index is somewhere between `0` and `mid`. That means that the part of array from the pivot element to the last one is non-rotated.
  - If target is in that non-rotated part as well: go right: `end = mid + 1`.
  - Otherwise: go left: `start = mid - 1`.

- We're here because the target is not found. Return -1.

Implementation

Java | Python

Copy

```
class Solution:
    def search(self, nums: List[int], target: int) -> int:
        start, end = 0, len(nums) - 1
        while start <= end:
            mid = start + (end - start) // 2
            if nums[mid] == target:
                return mid
            elif nums[mid] >= nums[start]:
                if target >= nums[start] and target < nums[mid]:
                    end = mid - 1
                else:
                    start = mid + 1
            else:
                if target <= nums[end] and target > nums[mid]:
                    start = mid + 1
                else:
                    end = mid - 1
        return -1
```

Complexity Analysis

- Time complexity : $\mathcal{O}(\log(N))$.
- Space complexity : $\mathcal{O}(1)$.

Analysis written by @liaison (https://leetcode.com/liaison/) and @andvary (https://leetcode.com/andvary/)

Rate this article:

Previous (/articles/reverse-linked-list-ii/) Next (/articles/valid-sudoku/)

Comments: 22

Articles > 33. Search in Rotated Sorted Array

Type comment here... (Markdown is supported)

Preview Post

haoyangfan (haoyangfan) ★ 384 January 24, 2019 6:42 PM

(/haoyangfan)

16-line C solution beats 100% that uses most basic form of binary search

```
int search(int* nums, int numsSize, int target) {
    int start = 0, end = numsSize - 1;
    while (start <= end) {
```

Read More

33 Share Reply

SHOW 2 REPLIES

reyou (reyou) ★ 31 April 26, 2019 6:32 PM

(/reyou)

Formula: If a sorted array is shifted, if you take the middle, always one side will be sorted. Take the recursion according to that rule.

1- take the middle and compare with target, if matches return.
2- if middle is bigger than left side, it means left is sorted

Read More

7 Share Reply

SHOW 1 REPLY

Sithis (sithis) ★ 4455 January 11, 2019 4:04 AM

(/sithis)

One pass solution.

```
public int search(int[] a, int key) {
    int lo = 0;
    int hi = a.length - 1;
```

Read More

10 Share Reply

SHOW 6 REPLIES

lim142857 (lim142857) ★ 10 July 26, 2019 9:21 PM

Python 3 modified binary search (readable)(fast)(short)

(/lim142857)

```
        # Initilize two pointers
        begin = 0
        end = len(nums) - 1
```

Read More

4 Share Reply

SHOW 1 REPLY

softwareshortcut (softwareshortcut) ★ 87 April 18, 2019 7:34 AM

(/softwareshortcut)

Creating helper methods is key otherwise you'd easily get lost dealing with indices. Great exercise, thanks for sharing. I see shorter solutions in the comments, but they might be tricky to come up with during an interview.

2 Share Reply

elstestnewway (elstestnewway) ★ 2 January 13, 2019 8:04 AM

(/elstestnewway)

Language C
0 ms

int search(int *nums, int numsSize, int target)
{

Read More

2 Share Reply

SHOW 1 REPLY

nish_d (nish_d) ★ 12 January 10, 2019 6:29 AM

(/nish_d)

find_rotate_index and search are doing almost the same thing. Is there a way to combine them both? We might have come across target in find_rotate_index already, and just need to end the program there.

2 Share Reply

SHOW 1 REPLY

yiqi_yan (yiqi_yan) ★ 22 January 16, 2019 11:57 AM

(/yiqi_yan)

The Java code has a bug: find_rotate_index doesn't deal with the case where the array is completely rotated.

e.g.: input
[5,4,3,1]

Read More

0 Share Reply

SHOW 3 REPLIES

(/wrg)

wrg (wrg) ★ 0 July 11, 2019 7:27 PM

Articles › 33. Search in Rotated Sorted Array

In the worst case isn't the time complexity of the solution O(n + log(n)) ? The worst case is when the pivot is at index n - 1.

So if I understand this correctly when n = 1 000 000 in worst case scenario (pivot is at index n -1))

Read More

0 Share Reply

SHOW 1 REPLY

(/softwareshortcut)

softwareshortcut (softwareshortcut) ★ 87 July 8, 2019 10:40 AM

Recursive version in Java.

```
class Solution {
    public int search(int[] nums, int target) {
        int size = nums.length;
```

Read More

0 Share Reply

‹ 1 2 3 ›

Copyright © 2019 LeetCode

(/privacy/) United States (/region/)