33. Search in Rotated Sorted Array

Add to List

Question | Editorial Solution | My Submissions (/problems/search-in-rotated-sorted-array/submissions/)

Total Accepted: **134396** Total Submissions: **423660** Difficulty: **Hard** Contributors: **Admin**

Suppose a sorted array 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.

Notes

Hide Company Tags | LinkedIn (/company/linkedin/) | Bloomberg (/company/bloomberg/) | Uber (/company/uber/) | Facebook (/company/facebook/) | Microsoft (/company/microsoft/)

Hide Tags | Binary Search (/tag/binary-search/) | Array (/tag/array/)

Hide Similar Problems | (M) Search in Rotated Sorted Array II (/problems/search-in-rotated-sorted-array-ii/) | (M) Find Minimum in Rotated Sorted Array (/problems/find-minimum-in-rotated-sorted-array/)

Have you met this question in a real interview? Yes No

Discuss (https://leetcode.com/discuss/questions/oj/search-in-rotated-sorted-array)

Pick One (/problems/random-one-question/)

C++

```
class Solution {
public:
// 6 binary search problems
// http://algorithmsandme.in/2015/04/binary-search-algorithm-and-related-problems/
/**
 * If mid is equal to element we are looking for, return mid.
Case 1 :
1. If the lower half of the array is sorted.
2. Check if A[mid] >= A[start] Check if the element being looked for is greater than A[start]  and less than A[mid]
3. Discard the upper array we have to look in lower array.
4. Else look in upper subarray.
Case 2 :
1. If lower array is not sorted that is A[mid] <= A[start]
2. If element is greater than A[mid] && less than A[end], look in upper sub array.
3. Else look in lower sub array.
*/
int search(vector<int>& nums, int target) {
    int lo = 0, hi = nums.size() - 1;
    while(lo <= hi) {
        int mid = (hi - lo) / 2 + lo;
        if(nums[mid] == target) return mid;
        if(nums[lo] <= nums[mid]) {
            if(nums[lo] <= target && target < nums[mid])
                hi = mid - 1;                               // if both side do -1 and +1, then lo <= hi
            else                                            // if only one side do -1 or +1, then lo < hi
                lo = mid + 1;
        } else {
            if(nums[mid] < target && target <= nums[hi]) {
                lo = mid + 1;
            } else {
                hi = mid - 1;
            }
        }
    }
    return -1;
}

// Second time 11/29/2016
// Solution 1 Find the Min element first (rotation point first), then do standard binary_search
//    int binary_search(vector<int>& nums, int lo, int hi, int target) {
//        while(lo <= hi) {
//            int mid = (hi - lo) / 2 + lo;
//            if (nums[mid] < target) {
//                lo = mid + 1;
//            } else if(nums[mid] > target) {
//                hi = mid - 1;
//            } else {
//                return mid;
//            }
//        }
//        return -1;
//    }

//   int search(vector<int>& nums, int target) {
//       int lo = 0, hi = nums.size() - 1;
//       while(nums[lo] > nums[hi]) {
//           int mid = (hi - lo) / 2 + lo;
//           if(nums[mid] > nums[hi]) {
//               lo = mid + 1;
//           } else {
//               hi = mid;
//           }
//       }
//       int minPos = lo;

//       int found = binary_search(nums, 0, minPos-1, target);
//       if(found != -1) {
//           return found;
```

Send Feedback (mailto:admin@leetcode.com?subject=Feedback)

```
69  //       } else {
70  //           return binary_search(nums, minPos, nums.size()-1, target);
71  //       }
72  //   }
73
74
75
76
77      // first time Leetcode so confusing
78      // need to do it again https://discuss.leetcode.com/topic/16580/java-ac-solution-using-once-binary-search
79      // If mid is equal to element we are looking for, return mid.
80      // Case 1 :
81      // 1. If the lower half of the array is sorted.
82      // 2. Check if A[mid] >= A[start] Check if the element being looked for is greater than A[start]  and less than A[mid]
83      // 3. Discard the upper array we have to look in lower array.
84      // 4. Else look in upper subarray.
85      // Case 2 :
86      // 1. If lower array is not sorted that is A[mid] <= A[start]
87      // 2. If element is greater than A[mid] && less than A[end], look in upper sub array.
88      // 3. Else look in lower sub array.
89      // int search(vector<int>& nums, int target) {
90      //     int left = 0, right = nums.size()-1;
91      //     while(left <= right) {
92      //         int mid = (right - left) / 2 + left;
93      //         if (nums[mid] == target) return mid;
94      //         if(nums[left] <= nums[mid]) {
95      //             if(nums[left] <= target && target < nums[mid])
96      //                 right = mid - 1;
97      //             else
98      //                 left = mid + 1;
99      //         }
100     //         if(nums[mid] <= nums[right]) {
101     //             if(nums[mid] < target && target <= nums[right])
102     //                 left = mid + 1;
103     //             else
104     //                 right = mid - 1;
105     //         }
106     //     }
107     //     return -1;
108     // }
```

Custom Testcase ☐ Shortcut: Command + enter

Run Code | Submit Solution

Submission Result: Accepted (/submissions/detail/84262749/)

More Details › (/submissions/detail/84262749/)

Next challenges: (H) Dungeon Game (/problems/dungeon-game) (H) Split Array Largest Sum (/problems/split-array-largest-sum) (M) Find All Duplicates in an Array (/problems/find-all-duplicates-in-an-array)

Share your acceptance!

11 27

Notes

Send Feedback (mailto:admin@leetcode.com?subject=Feedback)