Description | Editorial | Solutions (10K) | Submissions

33. Search in Rotated Sorted Array

Medium 22.3K 1.3K

Companies

There is an integer array `nums` sorted in ascending order (with **distinct** values).

Prior to being passed to your function, `nums` is **possibly rotated** at an unknown pivot index `k` (`1 <= k < nums.length`) such that the resulting array is `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]` (**0-indexed**). For example, `[0,1,2,4,5,6,7]` might be rotated at pivot index `3` and become `[4,5,6,7,0,1,2]`.

Given the array `nums` **after** the possible rotation and an integer `target`, return *the index of* `target` *if it is in* `nums`, *or* `-1` *if it is not in* `nums`.

You must write an algorithm with `O(log n)` runtime complexity.

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
```

Example 3:

Java | Auto

```java
class Solution {
    public int search(int[] nums, int target) {
        int start = 0, end = nums.length - 1;
        int mid = (start + end) / 2;
        while (start <= end) {
            mid = (start + end) / 2;
            if (target == nums[mid]) {
                return mid;
            }
            if (nums[start] <= nums[mid]) {
                if (nums[start] <= target && nums[mid] >= target) {
                    end = mid - 1;
                } else {
                    start = mid + 1;
                }
            } else {
                if (nums[end] >= target && nums[mid] <= target) {
                    start = mid + 1;
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

nums =
[4,5,6,7,0,1,2]

target =
0

Output

Console | Run | Submit

Description | Editorial | Solutions (2.7K) | Submissions

735. Asteroid Collision

Hint

Medium 6.4K 642

Companies

We are given an array `asteroids` of integers representing asteroids in a row.

For each asteroid, the absolute value represents its size, and the sign represents its direction (positive meaning right, negative meaning left). Each asteroid moves at the same speed.

Find out the state of the asteroids after all collisions. If two asteroids meet, the smaller one will explode. If both are the same size, both will explode. Two asteroids moving in the same direction will never meet.

Example 1:

```
Input: asteroids = [5,10,-5]
Output: [5,10]
Explanation: The 10 and -5 collide resulting in 10. The 5 and 10 never collide.
```

Example 2:

```
Input: asteroids = [8,-8]
Output: []
Explanation: The 8 and -8 collide exploding each other.
```

Example 3:

Java | Auto

```java
import java.util.*;

class Solution {
    public int[] asteroidCollision(int[] asteroids) {
        int n = asteroids.length;
        Deque<Integer> stk = new ArrayDeque<>();

        for (int i = 0; i < n; i++) {
            if (stk.isEmpty() || asteroids[i] > 0) {
                stk.push(asteroids[i]);
            } else {
                while (!stk.isEmpty() && stk.peek() > 0 && stk.peek() < Math.abs(asteroids[i])) {
                    stk.pop();
                }

                if (!stk.isEmpty() && stk.peek() == Math.abs(asteroids[i])) {
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

asteroids =
[5,10,-5]

Output

[5,10]

Expected

Console | Run | Submit