Problem 33: Search in Rotated Sorted Array

Problem Information

Difficulty: Medium
Acceptance Rate: 43.68%
Paid Only: No
Tags: Array, Binary Search

Problem Description

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

Prior to being passed to your function, `nums` is **possibly left rotated** at an unknown 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 left rotated by `3` indices 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:

Input: nums = [1], target = 0 **Output:** -1

Constraints:

* `1 <= nums.length <= 5000` * `-104 <= nums[i] <= 104` * All values of `nums` are **unique**. * `nums` is an ascending array that is possibly rotated. * `-104 <= target <= 104`

Code Snippets

C++:

```
class Solution {
public:
int search(vector<int>& nums, int target) {

}
};
```

Java:

```
class Solution {
public int search(int[] nums, int target) {

}
}
```

Python3:

```
class Solution:
def search(self, nums: List[int], target: int) -> int:
```