

44) Search in Rotated Sorted Array 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` ($1 \leq k < \text{nums.length}$) 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

CODE:

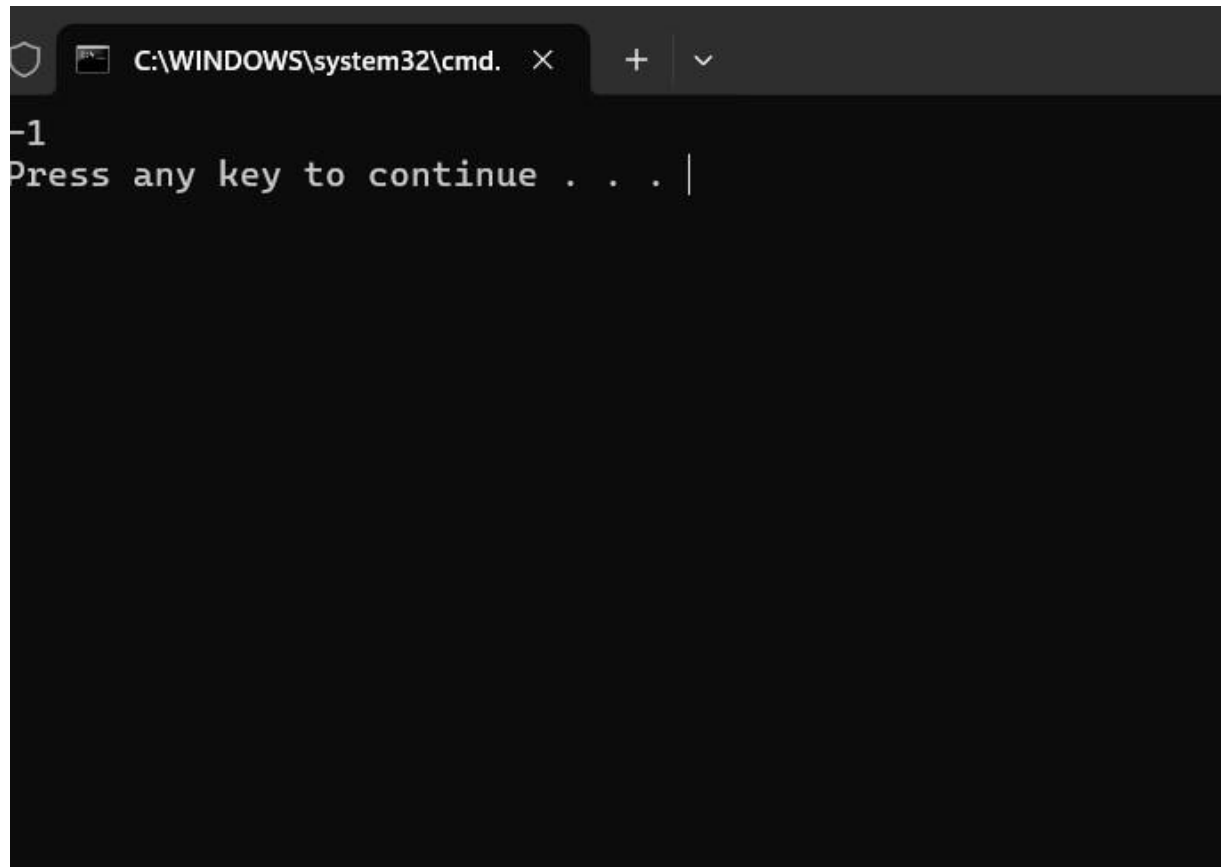
```
def search(a, t):  
    for i in range(len(a)):  
        if a[i] == t:  
            return i  
    return -1
```

```
a = [4, 3, 1, 2, 5, 0]
```

```
t = 10
```

```
print(search(a, t))
```

OUTPUT:

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.' and standard window controls. The command prompt displays the output '-1' on the first line, followed by the prompt 'Press any key to continue . . . |' on the second line. The background is black and the text is white.

TIME COMPLEXITY : $O(\log n)$