Search in Rotated Sorted Array

Try to solve the Search in Rotated Sorted Array problem.

We'll cover the following

Statement

Given a sorted integer array, `nums`, and an integer value, `target`, the array is rotated by some arbitrary number. Search and return the index of `target` in this array. If the `target` does not exist, return `-1`.

An original sorted array before rotation is given below:

1	10	20	47	59	63	75	88	99	107	120	133	155	162	176	188	199	200	210	222

After rotating this array 6 times, it changes to:

176	188	199	200	210	222	1	10	20	47	59	63	75	88	99	107	120	133	155	162

Constraints

- All values in `nums` are unique.
- The values in `nums` are in sorted in ascending order.
- The array may have been rotated by some arbitrary number.
- $1 \leq$ `nums.length` ≤ 5000
- $-10^4 \leq$ `nums[i]` $\leq 10^4$
- $-10^4 \leq$ `target` $\leq 10^4$

Examples

Sample example 1

Input

nums	6	7	1	2	3	4	5

```
target = 2
```

Output

```
Index = 3
```

Understand the problem

Let’s take a moment to make sure you’ve correctly understood the problem. The quiz below helps us to check if you’re solving the correct problem:

Search in Rotated Sorted Array

1 What is the output if the following rotated sorted array and target are given as input?

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

target = 1

A) 0

B) 1

C) 5

D) 6

Submit Answer

Question 1 of 3
0 attempted

Reset Quiz

Figure it out!

We have a game for you to play. Rearrange the logical building blocks to develop a clearer understanding of how to solve this problem.

Drag and drop the cards to rearrange them in the correct sequence

Declare two low and high pointers that will initially point to the first and last indexes of the array, respectively.

Declare a middle pointer that will initially point to the middle index of the array. This divides the array into two halves.

Check if the target is present at the position of the middle pointer. If it is, return its index.

If the first half of the array is sorted and the target lies in this range, update the high pointer to the middle pointer in order to search in the first half.

Else, if the second half of the array is sorted and the target lies in this range, update the low pointer to mid in order to search in the second half.

If the high pointer becomes greater or equal to the low pointer and we still haven't found the target, return -1.

Reset | Show Solution | Submit

Try it yourself

Implement your solution in the following coding playground:

usercode > main.java

```
import java.util.*;
public class Main{
    public static int binarySearchRotated(List<Integer> nums, int target) {

        // Write your code here

        return -1;
    }
}
```

Powered by AI | Submit

Test Cases | Results

Case 1 | Case 2 | Case 3

Input #1

[6,7,1,2,3,4,5]

Input #2

3

Search in Rotated Sorted Array

Back

Modified Binary Searc...

Next

Solution: Search in Ro...

Mark as Completed