Search in Rotated Sorted Array II

Try to solve the Search in Rotated Sorted Array II problem.

We'll cover the following

Statement

You are required to find an integer `t` in an array `arr` of non-distinct integers. *Prior* to being passed as input to your search function, `arr` has been processed as follows:

- It has been sorted in non-descending order.
- It has been rotated around some pivot k, such that, after rotation, it looks like this: `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]`. For example, `[10, 30, 40, 42, 42, 47, 78, 90, 901]`, rotated around pivot $k = 5$ becomes `[47, 78, 90, 901, 10, 30, 40, 42, 42]`.

Return TRUE if `t` exists in the rotated, sorted array `arr`, and FALSE otherwise, while minimizing the number of operations in the search.

> **Note:** In this problem, the value of k is not passed to your search function.

Constraints

- $1 \leq$ `arr.length` ≤ 5000
- $-10^4 \leq$ `arr[i]` $\leq 10^4$
- `arr` is guaranteed to be rotated at some pivot index.
- $-10^4 \leq$ `t` $\leq 10^4$

Examples

Sample example 1

Input

arr	39	46	63	-12	5	17
t	-12					

Hint: k = 3

Output

TRUE

Understand the problem

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

1 What is the value of the pivot k around which this sorted array has been rotated?

arr = [-10, -9, -7, 20, -29, -21]

A) 3

B) 2

C) 4

D) 6

Submit Answer

Question 1 of 4
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.

> **Note:** As an additional challenge, we have intentionally hidden the solution to this puzzle.

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 True.

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

Reset | Show Solution | Submit

Try it yourself

Implement your solution in the following coding playground.

> **Note:** We have left the solution to this challenge as an exercise for you. You may try to translate the logic of the solved puzzle into a coded solution.

usercode > Search.java

```
1  import java.util.*;
2  public class Search{
3     public static boolean search(int[] arr, int t) {
4
5        // Write your code here
6
7        return false;
8     }
9  }
```

Powered by AI | Submit

Test Cases | Results

Case 1 | Case 2 | Case 3

Input #1

[22,67,-14,16,19]

Input #2

19

Search in Rotated Sorted Array II

Back

Solution: Single Eleme...

Next

Subsets: Introduction

Mark as Completed