Problem Solving - Divide and Conquer

prob-divide-conquer-starter-code.zip 3.8KB

In this exercise, you'll utilize problem solving patterns to solve the following code challenges:

countZeroes

Given an array of 1s and 0s which has all 1s first followed by all 0s, write a function called ***countZeroes***, which returns the number of zeroes in the array.

Constraints:

Time Complexity: O(log N)

Examples:

```
countZeroes([1,1,1,1,0,0]) // 2 countZeroes([1,0,0,0,0]) // 4 countZeroes
([0,0,0]) // 3 countZeroes([1,1,1,1]) // 0
```

sortedFrequency

Given a sorted array and a number, write a function called ***sortedFrequency*** that counts the occurrences of the number in the array

Constraints:

Time Complexity: O(log N)

Examples:

```
sortedFrequency([1,1,2,2,2,2,3],2) // 4 sortedFrequency([1,1,2,2,2,2,3],3)
// 1 sortedFrequency([1,1,2,2,2,2,3],1) // 2 sortedFrequency([1,1,2,2,2,2,
3],4) // -1
```

findRotatedIndex

Write a function called ***findRotatedIndex*** which accepts a rotated array of sorted numbers and an integer. The function should return the index of num in the array. If the value is not found, return -1.

Constraints:

Time Complexity: O(log N)

Examples:

```
findRotatedIndex([3,4,1,2],4) // 1 findRotatedIndex([6, 7, 8, 9, 1, 2, 3,
4], 8) // 2 findRotatedIndex([6, 7, 8, 9, 1, 2, 3, 4], 3) // 6 findRotatedIn
dex([37,44,66,102,10,22],14) // -1 findRotatedIndex([6, 7, 8, 9, 1, 2, 3,
4], 12) // -1
```

findRotationCount

Write a function called ***findRotationCount*** which accepts an array of distinct numbers sorted in increasing order. The array has been rotated counter-clockwise n number of times. Given such an array, find the value of n.

Constraints:

Time Complexity: O(log N)

Examples:

```
findRotationCount([15, 18, 2, 3, 6, 12]) // 2 findRotationCount([7, 9, 11, 1
2, 5]) // 4 findRotationCount([7, 9, 11, 12, 15]) // 0
```

findFloor

Write a function called ***findFloor*** which accepts a sorted array and a value x, and returns the floor of x in the array. The floor of x in an array is the largest element in the array which is smaller than or equal to x. If the floor does not exist, return -1.

Examples:

```
findFloor([1,2,8,10,10,12,19], 9) // 8 findFloor([1,2,8,10,10,12,19], 20) //
19 findFloor([1,2,8,10,10,12,19], 0) // -1
```

Constraints

Time Complexity: O(log N)

Further Study

https://leetcode.com/tag/divide-and-conquer/

Solution

prob-divide-conquer-solution.zip 4.9KB