Binary Search Questions

17 August 2024 22:10

Q1. Ceiling of a number

Tip:1 : When you see Sorted Array, use BS first then check for other approaches.

-> arr = [2, 3, 5, 9, 14, 16, 18]

If target = 14, find Ceiling of that target number.

Ceiling = Smallest element in array >or = to target element

Ceiling (arr, target = 14) = 14

Ceiling (arr, target = 15) = 16

target = 4 = 5

target = 9 = 9

s m e

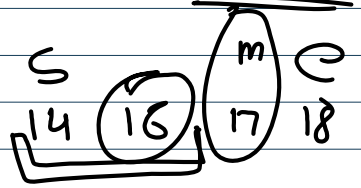
0 1 2 3 4 5 6

arr = [2, 3, 5, 9, 14, 16, 18] Tgt = 15

mid = 3 (3rd index)

s	m	e
4	5	6
14	16	18

Imagine that



Psbl ans

se	m	e
4	5	6
14	16	18

sme

14

=> target > mid + 1

=> s = m + 1

e 5

=> s = m+1

e s
14, 16, 18

Condition that element isn't found (Breaking of loop in BS)

e s
14 5 16 6 18

e target s => for loop violated condition

while (s ≤ end) => when while loop breaks

s = e+1

Give next big number when no answer is found
return 's'

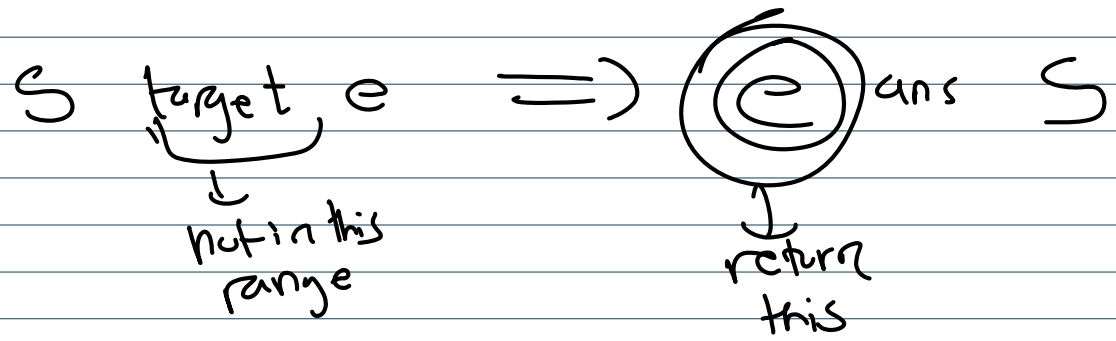
watch from 32:00 to 32:55 for clarity.

Q.1 Floor of a number

Floor = greatest number < or = target

arr = [2, 3, 5, 9, 14, 16, 18] Flour of 15

// same thing as before, just return "end" instead of -1 if the answer isn't found



83. Find Smallest Letter greater than target (Leetcode 744)

744. Find Smallest Letter Greater Than Target

Easy | Topics | Companies | Hint

→ ascending

You are given an array of characters `letters` that is sorted in **non-decreasing order**, and a character `target`. There are **at least two different** characters in `letters`.

Return *the smallest character in* `letters` *that is lexicographically greater than* `target`. If such a character does not exist, return the first character in `letters`.

Example 1:

```
Input: letters = ["c","f","j"], target = "a"
Output: "c"
Explanation: The smallest character that is lexicographically greater than 'a' in letters is 'c'.
```

Example 2:

```
Input: letters = ["c","f","j"], target = "c"
Output: "f"
Explanation: The smallest character that is lexicographically greater than 'c' in letters is 'f'.
```

Example 3:

```
Input: letters = ["x","x","y","y"], target = "z"
Output: "x"
Explanation: There are no characters in letters that is lexicographically greater than 'z' so we return letters[0].
```

(1) Exact same approach as of the ceiling wala problem

(2) Ignore the target = element condition

(3) If arr = [c, d, f, j] (4); target = j

then return "c"

Condition violated : start = end + 1 ⟹ length of array that is 'N'

In this case, return start % N

where N = length of array

Q4.

34. Find First and Last Position of Element in Sorted Array

Medium | Topics | Companies

ascending but not strictly increasing

Given an array of integers `nums` sorted in non-decreasing order, find the starting and ending position of a given `target` value.

If `target` is not found in the array, return `[-1, -1]`.

You must write an algorithm with `O(log n)` runtime complexity.

Example 1:

```
Input: nums = [5,7,7,8,8,10], target = 8
Output: [3,4]
```

Example 2:

```
Input: nums = [5,7,7,8,8,10], target = 6
Output: [-1,-1]
```

Example 3:

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

→ arr: [5, 7, 7, 7, 7, 8, 8, 10] (0 1 2 3 4 5 6 7), target = 7

to return → [1, 4] : target is '7'

One Approach: Run BS 2 times & in 1st one → find 1st occurence of 7

2nd one → last occurence of 7

Find 1st occurence of 7

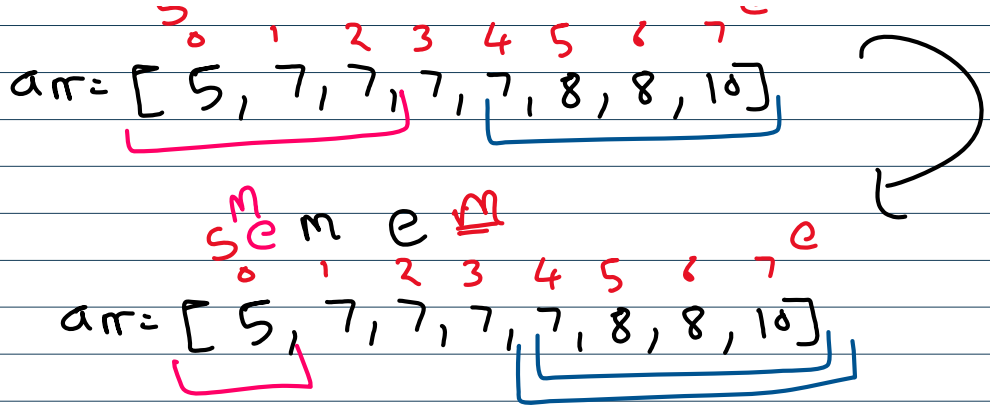
arr: [5, 7, 7, 7, 7, 8, 8, 10] (s = 0, m = 3, e = 7)

1st possible ans: Index '3'

There is a possibility that 7 can be on the left & for that run BS 2nd twice & do end = mid - 1

For last occurence ⇒ start = mid + 1, run BS 2nd time

arr: [5, 7, 7, 7, 7, 8, 8, 10] (s = 0, m = 3, e = 7)



Q5.

Find position of an element in a sorted array of infinite numbers

Difficulty Level : Medium • Last Updated : 07 May, 2021

Suppose you have a sorted array of infinite numbers, how would you search an element in the array?
Source: Amazon Interview Experience.
Since array is sorted, the first thing clicks into mind is binary search, but the problem here is that we don't know size of array.
If the array is infinite, that means we don't have proper bounds to apply binary search. So in order to find position of key, first we find bounds and then apply binary search algorithm.
Let low be pointing to 1st element and high pointing to 2nd element of array, Now compare key with high index element,
->if it is greater than high index element then copy high index in low index and double the high index.
->if it is smaller, then apply binary search on high and low indices found.

arr: [2, 3, 5, 6, 7, 8, 10, 11, 12, 15, 20, 23, 30]

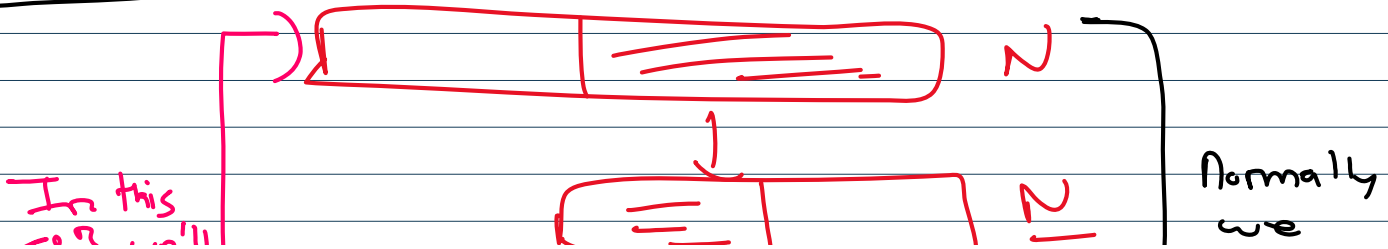
Target = 15

→ Challenge: We don't know start, end since this is an infinite array

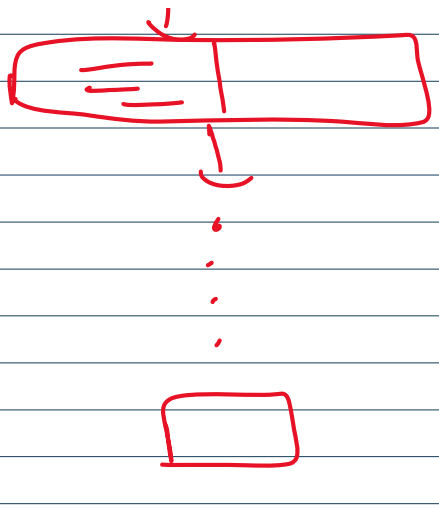
→ Solution: Divide the array in small chunks & each time you don't find the element in that chunk, double the chunk size everytime

(Assume that initial size is just '1')

Explanation:



In this cs^n, we'll go like this i.e exponentially doubling the size of array



Normally we do this

arr: [2, 3, 5, 6, 7, 8, 10, 11, 12, 15, 20, 23, 30]

we'll do this

when we are finding in a range, if tgt element is greater than "end", then don't need to check start.

Finding size of array by indices:

arr = [4, 13, 15, 16, 123, 200]

s-1 elements

$e - (s-1) = (e - s + 1)$

$4 - 2 + 1 = 3$

this

Q6

852. Peak Index in a Mountain Array

Medium | Topics | Companies

You are given an integer **mountain** array `arr` of length `n` where the values increase to a **peak element** and then decrease.

Return the index of the peak element.

Your task is to solve it in `O(log(n))` time complexity.

Example 1:

Input: `arr = [0,1,0]`

Output: 1

Example 2:

Input: `arr = [0,2,1,0]`

Output: 1

Example 3:

Input: `arr = [0,10,5,2]`

Output: 1

0 1 2 3 4

[2, 5, 10, 5, 0]

return "2"

↓

index of array

2 ways

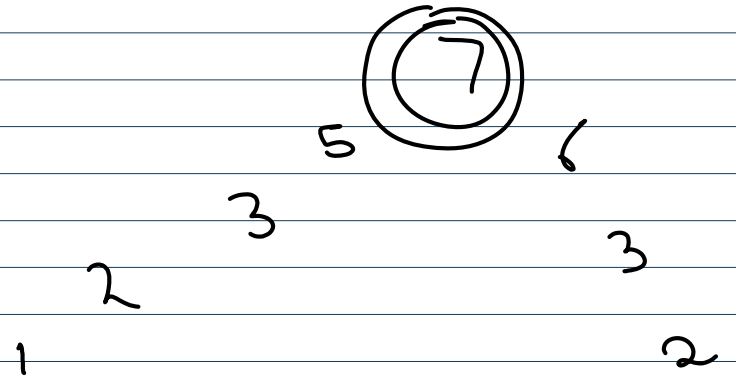
1] Linear Srch → ✗ since O(log n) is reqd

2] Binary Srch

↳ Best option

0 1 2 3 4 5 6 7

arr = [1, 2, 3, 5, 7, 6, 3, 2]



Bitonic Array

① If arr[mid] > arr[mid+1] ⇒ You are in the decreasing part of the array

end = mid // checking in the Left

② arr[mid] < arr[mid+1] ⇒ You are in the increasing part of the array

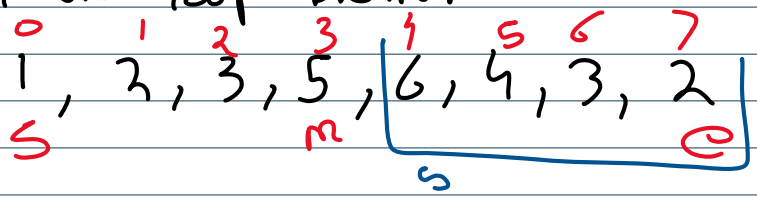
start = mid+1

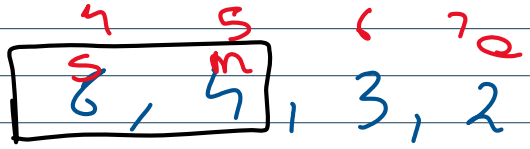
start = mid+1

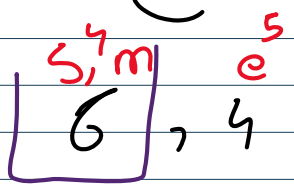
↓

doing this bcoz here we know that there is an element greater than that. In the above one, we weren't aware

③ When will loop break?







In the end, 's' & 'e' will both point to the largest number

Ans → Why? bcoz ① & ②

④

162. Find Peak Element

Medium · Topics · Companies

A peak element is an element that is strictly greater than its neighbors.

Given a **0-indexed** integer array `nums`, find a peak element, and return its index. If the array contains multiple peaks, return the index to **any of the peaks**.

You may imagine that `nums[-1] = nums[n] = -∞`. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

You must write an algorithm that runs in `O(log n)` time.

Example 1:

```
Input: nums = [1,2,3,1]
Output: 2
```

162. Find Peak Element

Medium · Topics · Companies

A peak element is an element that is strictly greater than its neighbors.

Given a **0-indexed** integer array `nums`, find a peak element, and return its index. If the array contains multiple peaks, return the index to **any of the peaks**.

You may imagine that `nums[-1] = nums[n] = -∞`. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

You must write an algorithm that runs in `O(log n)` time.

Example 1:

```
Input: nums = [1,2,3,1]
Output: 2
Explanation: 3 is a peak element and your function should return the index number 2.
```

Example 2:

```
Input: nums = [1,2,1,3,5,6,4]
Output: 5
Explanation: Your function can return either index number 1 where the peak element is 2, or index number 5 where the peak element is 6.
```

Same soln as above Problem (Submit it & Check!!)

1095. Find in Mountain Array

Hard · Topics · Companies · Hint

*(This problem is an **interactive problem**.)*

You may recall that an array `arr` is a **mountain array** if and only if:

- `arr.length >= 3`
- There exists some `i` with `0 < i < arr.length - 1` such that:
  - `arr[0] < arr[1] < ... < arr[i - 1] < arr[i]`
  - `arr[i] > arr[i + 1] > ... > arr[arr.length - 1]`

Given a mountain array `mountainArr`, return the **minimum** `index` such that `mountainArr.get(index) == target`. If such an `index` does not exist, return `-1`.

You cannot access the mountain array directly. You may only access the array using a `MountainArray` interface:

- `MountainArray.get(k)` returns the element of the array at index `k` (0-indexed).
- `MountainArray.length()` returns the length of the array.

Submissions making more than `100` calls to `MountainArray.get` will be judged *Wrong Answer*. Also, any solutions that attempt to circumvent the judge will result in disqualification.

Example 1:

```
Input: array = [1,2,3,4,5,3,1], target = 3
Output: 2
Explanation: 3 exists in the array, at index=2 and index=5. Return the minimum index, which is 2.
```

Example 2:

```
Input: array = [0,1,2,4,2,1], target = 3
Output: -1
Explanation: 3 does not exist in the array, so we return -1.
```

Q:

arr = [1, 2, 3, 4, 5, 3, 1] (indices 0 1 2 3 4 5 6)

target = 3

Ans:

1. Find peak element ⇒ 4 index
2. Binary Search in asc array ⇒ (0, 4)
3. If not found, binary search in dce array = [4, 6]

2. Search in Rotated Array

33. Search in Rotated Sorted Array

Medium · Topics · Companies

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 <= 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 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.

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
```

What is a rotated array?

arr = [2, 4, 5, 7, 8, 9, 10, 12] (indices 0 1 2 3 4 5 6 7)

After 1 rotation

arr = [12, 2, 4, 5, 7, 8, 9, 10] (indices 0 1 2 3 4 5 6 7)

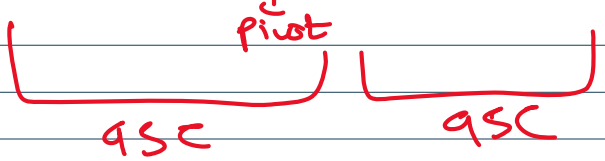
2nd rotation

arr = [10, 12, 2, 4, 5, 7, 8, 9]
(0 1 2 3 4 5 6 7)
↑ pivot

Approach 1: Find the pivot in the array & then apply B.S

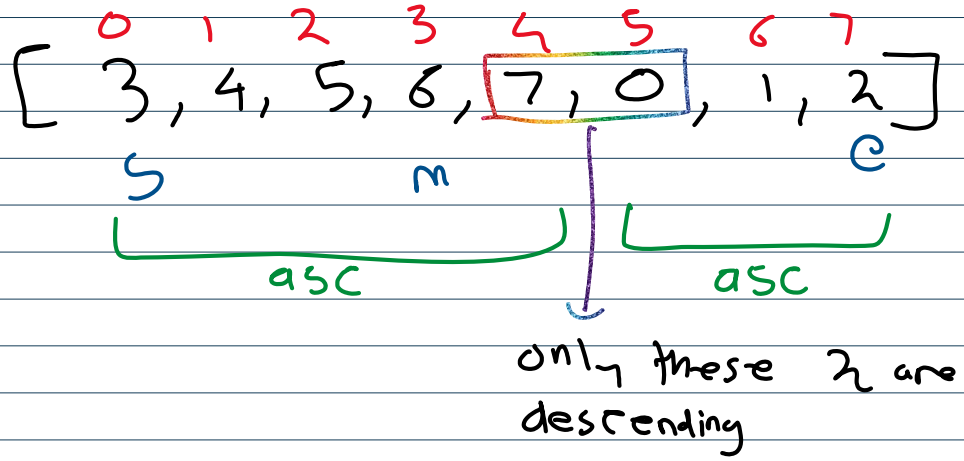
pivot => From where next numbers are ascending

arr = [3, 4, 5, 6, 7, 0, 1, 2]



- ~~Find pivot~~
- ~~Srch in 1st half of array: Simple B.S (0, pivot)~~
- ~~Otherwise, Srch in next half: (pivot+1, end)~~

So now, our q^n is to srch the pivot

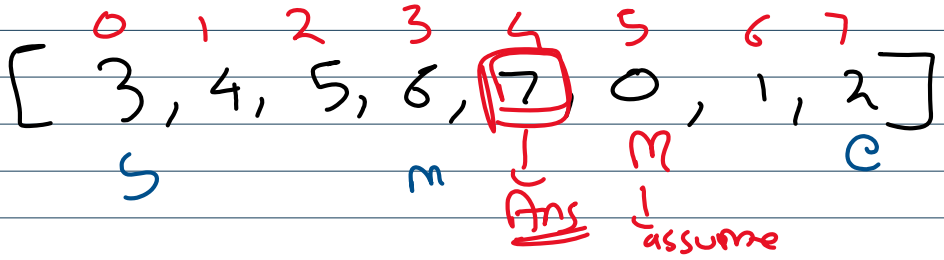


Case 1: When you find that mid > mid+1, i.e the "mid" is pivot

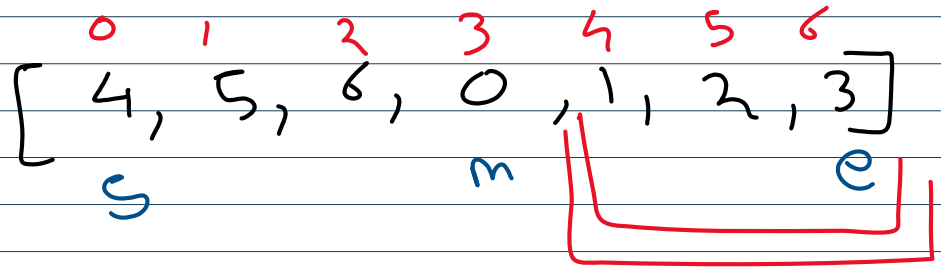
Case 2: mid element < (mid-1) element

Case 2: mid element < (mid - 1) element

∴ ans => $(m-1)^{th}$ index pe jo element hoga

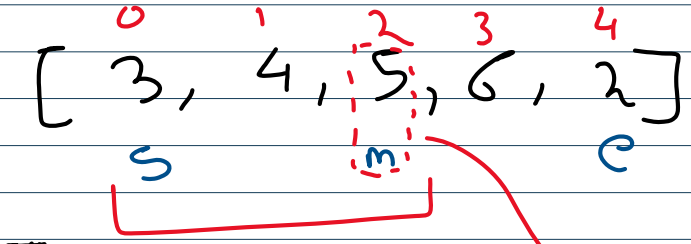


Case 3: Start element ≥ mid element



∴ end = mid - 1 => Since we are only looking for the peak element. We can ignore all those smaller than "start"

Case 4: Start element < mid element



This part is definately sorted since "s < m" It means that bigger numbers lie ahead ∴ we can ignore till "mid" & set

→ If this was the pivot then it would've been already returned in Case 1 & 2 :)

numbers lie ahead ✓s we can ignore till "mid" & set start = mid + 1

already returned in Case 1 & 2 :)

```
static int search(int[] nums, int target) {  no usages
    int pivot = findPivot(nums);

    // If pivot is not found , i.e array is not rotated. In that case , do normal Binary Search
    if(pivot == -1){
        return binarySearch(nums , target , start: 0 , end: nums.length-1);
    }

    // if pivot is found , you have 2 ascending sorted arrays
    if(nums[pivot] == target){
        return pivot;
    }

}
```

Explanation for this fxn

arr = [4, 5, 6, 7, 0, 1, 2]
S P E

Case 1 : Pivot element == target // Ans

Case 2 : Target >= start element eg. (tgt : 6)

srch space : (S, p-1) => Bcoz all numbers after pivot are smaller than start

Case 3 : Target < Start element

i.e all elements from start till pivot are > than target & thus need to be ignored eg. (tgt : 1)

srch space : (p+1, end)

```
// wont work for duplicate values
static int findPivot(int[] arr){  1 usage
    int start = 0;
    int end = arr.length-1;

    while (start <=end){
        int mid = start + (end-start) /2;
        // 4 cases over here

        // case 1

        if(mid < end && arr[mid] > arr[mid+1]){
            return mid;
        }

        // case 2

        if(mid >start && arr[mid] < arr[mid-1]){
            return (mid-1);
        }
```

This won't work for array having duplicate values

arr = [2, 2, 2, 2, 9]

|| Rotate Twice

[2, 9, 2, 2, 2]
S M E

```
if(mid >start && arr[mid] < arr[mid-1]){
    return (mid-1);
}

// case 3

if(arr[mid] <= arr[start]){
    end = mid-1;
}

// case 4

else{
```

[2, 9, 2, 2, 2]
s m e

If start = mid = end ignore the start & end & increment them. This will not bring any impart on the answer.

Find the Rotation Count in Rotated Sorted array

Last Updated : 12 Jul, 2024

Given an array **arr[]** of size **N** having distinct numbers sorted in increasing order and the array has been right rotated (i.e, the last element will be cyclically shifted to the starting position of the array) **k** number of times, the task is to find the value of **k**.

Examples:

Input: *arr[] = {15, 18, 2, 3, 6, 12}*
Output: *2*
Explanation: *Initial array must be {2, 3, 6, 12, 15, 18}.*
We get the given array after rotating the initial array twice.

Input: *arr[] = {7, 9, 11, 12, 5}*
Output: *4*

Input: *arr[] = {7, 9, 11, 12, 15};*
Output: *0*

Eg. arr = [4, 5, 6, 7, 0, 1, 2]

Find how many times this array has been rotated.

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

0 1 2 3 4 5 6 — Rotated 4 times from orginal one

[4, 5, 6, 7, 0, 1, 2, 3] ~~4, 5, 6, 7~~

Pivot = 3

∴ Ans = Pivot + 1

410. Split Array Largest Sum

Hard · Topics · Companies

Given an integer array `nums` and an integer `k`, split `nums` into `k` non-empty subarrays such that the largest sum of any subarray is **minimized**.

Return *the minimized largest sum of the split.*

A **subarray** is a contiguous part of the array.

Example 1:

410. Split Array Largest Sum

Hard | Topics | Companies

Given an integer array `nums` and an integer `k`, split `nums` into `k` non-empty subarrays such that the largest sum of any subarray is **minimized**.

Return *the minimized largest sum of the split.*

A **subarray** is a contiguous part of the array.

Example 1:

```
Input: nums = [7,2,5,10,8], k = 2
Output: 18
Explanation: There are four ways to split nums into two subarrays.
The best way is to split it into [7,2,5] and [10,8], where the largest sum among the two subarrays is only 18.
```

Example 2:

```
Input: nums = [1,2,3,4,5], k = 2
Output: 9
Explanation: There are four ways to split nums into two subarrays.
The best way is to split it into [1,2,3] and [4,5], where the largest sum among the two subarrays is only 9.
```

arr: [7, 2, 5, 10, 8] , k = 2
(2 subarrays to be made)
continuous

How can we split the array?

1st sub Array	2nd Subarray	Largest Sum
7, 2, 5, 10] → 24	8] → 8	24
7, 2, 5] → 14	10, 8] → 18	18 ⇒ Ans
7, 2] → 9	5, 10, 8] → 23	23
7] → 7	2, 5, 10, 8] → 25	25

Minimum in all the largest sums

① Min. no of partitions that we can make = 1

② Max number of partitions that we can make is 'N'

arr: [3, 4, 1, 2] ⇒ [3], [4], [1], [2]

Ans in Case 1: Entire array

Ans in Case 1: Entire array

[7, 2, 5, 10, 8] => Sum = (32)

Ans in Case 2:

Ans will be = [4] => The largest among all the subarrays

Max value of ans of question: Case - 1 //

min value of ans of question: Case - 2 //

---

minAns = max value in Array

maxAns = Sum of all values in array

According to example we took,

minAns = 10
maxAns = 32

[10, 32]

Start = 10 end = 32

$mid = \frac{s+e}{2} = 21$ //

Now, try to see if you can split the array with 21 being the max Sum

	Pieces
7, 2, 5, 8, 10	
[7, 2, 5] [8, 10]	2

[7, 2, 5] [8, 10] | 2

Check 1 What if (pieces ≤ m) ⟹ This means that individual sum of all the subarrays is less → thus we need to check if we can increase the sum

end = mid

s = 10 , e = 21

mid = 15

7, 2, 5, 8, 10

[7, 2, 5] , [8] , [10] | Pieces 3

allowed

what if pieces = 2 ?

Check 2 if pieces > m

Start = mid + 1

s = 16 , e = 21

7, 2, 5, 8, 10

$s = 16$, $e = 21$

$7, 2, 5, 8, 10$

$m = 18$

$[7, 2, 5]$, $[8, 10]$ pieces = 2

$\Rightarrow$ $s = 11$, $e = 18$

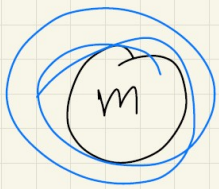
$m = 17$ $7, 2, 5, 8, 10$

$[7, 2, 5]$, $[8]$, $[17]$ pieces = 3

$s = m + 1 = 18$

$s = 18$, $e = 18$

$m = 18$ Ans



// Ans will definetly exist, just apply 2 checks & you will reach the answer. //