

Classwork

First and Last Position of an Element In Sorted Array [Coding Ninjas]

Question Link:-

https://www.codingninjas.com/studio/problems/first-and-last-position-of-an-element-in-sorted-array_1082549?leftPanelTab=0

The screenshot shows the Coding Ninjas studio interface. On the left, the 'Current Submission' panel indicates a 'Correct Answer' with 50/50 test cases passed. The runtime graph shows a peak at 4416 ms. The right panel displays the Java code for the 'firstAndLastPosition' method, which uses binary search to find the first and last occurrence of an element in a sorted array.

Java Code

```
import java.io.*;
import java.util.*;

public class Solution {

    public static int[] firstAndLastPosition(ArrayList<Integer> arr, int
n, int k) {
        // Write your code here.
        int[] answer=new int[2];

        //First occurrence
        int start=0;
        int end=n-1;
        int ans=-1;
        int mid=start+(end-start)/2;
        while(start<=end) {
            if(arr.get(mid)==k) {
                ans=mid;
                end=mid-1;
            }else if(arr.get(mid)<k) {
```

```

        start=mid+1;
    }else{
        end=mid-1;
    }
    mid=start+(end-start)/2;
}
answer[0]=ans;

//Last occurrence
start=0;
end=n-1;
ans=-1;
mid=start+(end-start)/2;
while(start<=end){
    if(arr.get(mid)==k){
        ans=mid;
        start=mid+1;
    }else if(arr.get(mid)<k){
        start=mid+1;
    }else{
        end=mid-1;
    }
    mid=start+(end-start)/2;
}
answer[1]=ans;

return answer;
}

};

```

Number of occurrence [Coding Ninjas]

Question link:-

https://www.codingninjas.com/studio/problems/occurrence-of-x-in-a-sorted-array_630456?leftPanelTab=0

Topics
Problem
Submissions
Solution
Discuss

Current Submission Report

Correct Answer
Test Cases EXP:
Penalty
Java

few secs ago 30/30 ⚡ 80/80 0%

Runtime graph

Memory graph

You performed better than

46.99%

Runtime

2859 ms

Did you find these test cases useful?

👍
👎

```

30 end=n-1;
31 ans=0;
32 mid=start+(end-start)/2;
33 while(start<=end){
34     if(arr[mid]==x){
35         ans=mid;
36         start=mid+1;
37     }else if(arr[mid]<x){
38         start=mid+1;
39     }else{
40         end=mid-1;
41     }
42     mid=start+(end-start)/2;
43 }
44 answer[1]=ans;
45 
46 if(answer[0]==0 && answer[1]==0){
47     return 0;
48 }
49 numberOfOccurrence=answer[1]-answer[0]+1;
50 return numberOfOccurrence;
51 
52 }
53 
```

< Prev
Next >

View hints
Last saved on 11:10:17 AM

Run
Submit

Java Code

```
public class Solution {
    public static int count(int arr[], int n, int x) {
        //Your code goes here
        int[] answer=new int[2];
        answer[0]=0;
        answer[1]=0;
        int numberOfOccurrence=0;

        //First occurrence
        int start=0;
        int end=n-1;
        int ans=0;
        int mid=start+(end-start)/2;
        while(start<=end){
            if(arr[mid]==x){
                ans=mid;
                end=mid-1;
            }else if(arr[mid]<x){
                start=mid+1;
            }else{
                end=mid-1;
            }
            mid=start+(end-start)/2;
        }
        answer[0]=ans;
    }
}
```

```

        //Last occurrence
        start=0;
        end=n-1;
        ans=0;
        mid=start+(end-start)/2;
        while(start<=end) {
            if(arr[mid]==x) {
                ans=mid;
                start=mid+1;
            }else if(arr[mid]<x) {
                start=mid+1;
            }else{
                end=mid-1;
            }
            mid=start+(end-start)/2;
        }
        answer[1]=ans;

        if(answer[0]==0 && answer[1]==0) {
            return 0;
        }
        numberOfOccurrence=answer[1]-answer[0]+1;
        return numberOfOccurrence;
    }
}

```

Peak index in a Mountain Array [LeetCode]

Question Link:-

<https://leetcode.com/problems/peak-index-in-a-mountain-array/description/>

852. Peak Index in a Mountain Array

Medium

An array `arr` is a **mountain** if the following properties hold:

- `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]`

Runtime: 0 ms, Beats 100.00% of users with Java
Memory: 55.08 MB, Beats 75.39% of users with Java

```

1 class Solution {
2     public int peakIndexInMountainArray(int[] arr) {
3         int start=0;
4         int end=arr.length-1;
5         int mid=start+(end-start)/2;
6
7         while(start<=end){
8             if(arr[mid-1]<arr[mid] && arr[mid]>arr[mid+1]){
9                 return mid;
10            }else if(arr[mid]<arr[mid+1]){
11                start=mid+1;
12            }else {
13                end=mid;
14            }
15            mid=start+(end-start)/2;
16        }
17        return -1;
18    }
19 }
20

```

Java Code

```

class Solution {
    public int peakIndexInMountainArray(int[] arr) {
        int start=0;
        int end=arr.length-1;
        int mid=start+(end-start)/2;

        while(start<=end) {
            if(arr[mid-1]<arr[mid] && arr[mid]>arr[mid+1]) {
                return mid;
            }else if(arr[mid]<arr[mid+1]) {
                start=mid+1;
            }else {
                end=mid;
            }

            mid=start+(end-start)/2;
        }
        return -1;
    }
}

```

Homework

Find Pivot Index [LeetCode]

Question Link:-

<https://leetcode.com/problems/find-pivot-index/description/>

The screenshot shows the LeetCode interface for problem 724, "Find Pivot Index". The problem description states: "Given an array of integers `nums`, calculate the **pivot index** of this array. The **pivot index** is the index where the sum of all the numbers **strictly** to the left of the index is equal to the sum of all the numbers **strictly** to the index's right." The problem is marked as "Solved" and "Easy".

Testcase results show the solution is "Accepted" with a runtime of 1 ms (beating 98.82% of users) and a memory usage of 44.35 MB (beating 25.95% of users).

The Java code provided in the editor is as follows:

```
1 class Solution {
2     public int pivotIndex(int[] nums) {
3         if(nums.length == 0){
4             return - 1;
5         }
6
7         int leftSum = 0;
8         int rightSum = 0;
9         for(int num : nums) {
10             rightSum=rightSum+num;
11         }
12
13         for(int i = 0; i < nums.length; i++) {
14             rightSum=rightSum-nums[i];
15             if(rightSum == leftSum) {
16                 return i;
17             }
18             leftSum =leftSum+nums[i];
19         }
20         return - 1;
21     }
22 }
```

Java Code

```
class Solution {
    public int pivotIndex(int[] nums) {
        if(nums.length == 0){
            return - 1;
        }

        int leftSum = 0;
        int rightSum = 0;
        for(int num : nums) {
            rightSum=rightSum+num;
        }

        for(int i = 0; i < nums.length; i++) {
            rightSum=rightSum-nums[i];
            if(rightSum == leftSum) {
                return i;
            }
            leftSum =leftSum+nums[i];
        }
        return - 1;
    }
}
```