Friday, 22 November, 2024 08:55 PM

sorted ↑

First and Last Occurrences

Find 2 :

Difficulty: **Medium** Accuracy: **37.36%** Submissions: **271K+** Points: **4**

Given a sorted array **arr** with possibly some duplicates, the task is to find the first and last occurrences of an element **x** in the given array.

Note: If the number **x** is not found in the array then return both the indices as -1. {-1, -1}

Examples:

Input: arr[] = [1, 3, 5, 5, 5, 5, 67, 123, 125], x = 5
Output: [2, 5]
Explanation: First occurrence of 5 is at index 2 and last occurrence of 5 is at index 5

Input: arr[] = [1, 3, 5, 5, 5, 5, 7, 123, 125], x = 7
Output: [6, 6]
Explanation: First and last occurrence of 7 is at index 6

0 1
{ a_1 a_2 a_3 ... a_n } -

{ a_1, a_1, a_1, a_2, a_2, a_3, a_4 ... a_n } $x = a_1$

0 2.

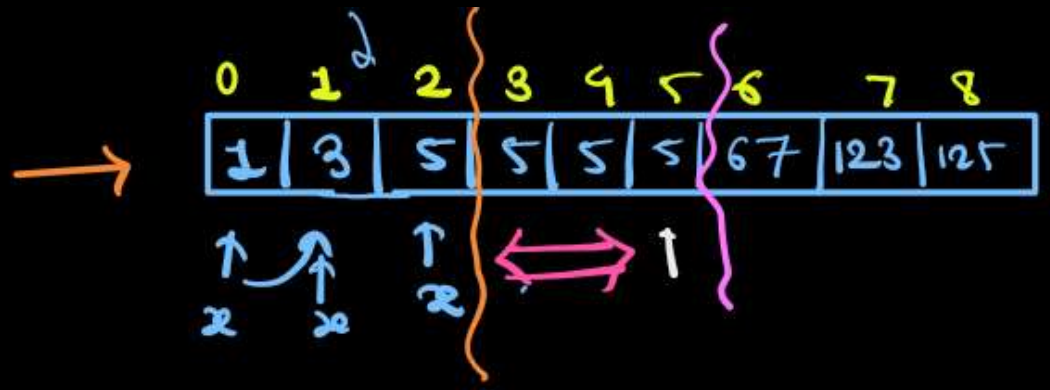
arr =

0	1	2	3	4	5	6	7	8
1	3	5	5	5	5	67	123	125

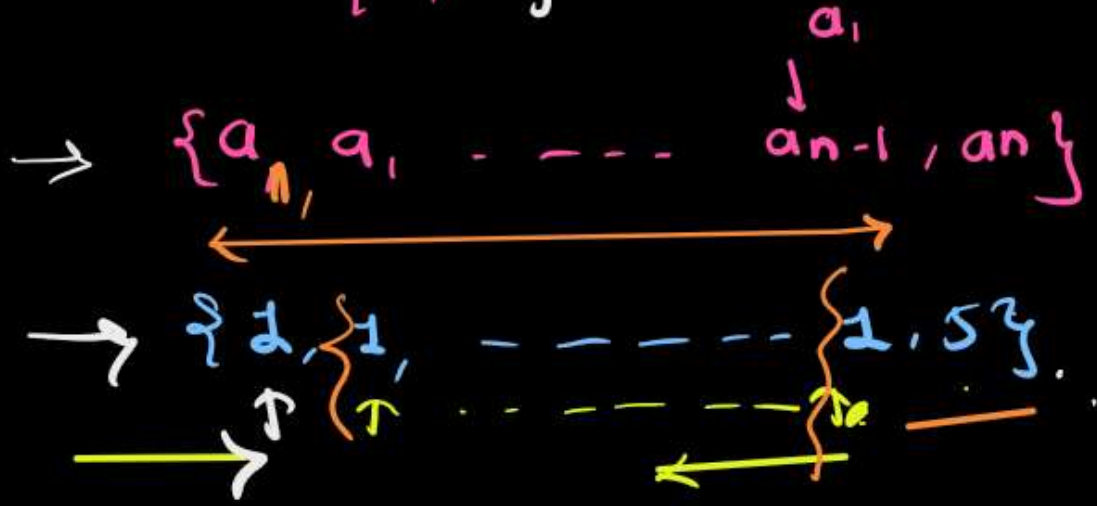
$x = 5$

{ 2, 5 }

Approach 1 -



{2, 5} →



{0, n-1}

```java
class GFG {
    ArrayList<Integer> find(int arr[], int x) {
        int n = arr.length;
        // code here
        ArrayList<Integer> ans = new ArrayList<>();
        int firstOcc = -1;
        int lastOcc = -1;
        // left to right
        for(int i = 0; i < n; i++){
            if(arr[i] == x){
                firstOcc = i;
                break;
            }
        }
        // right to left
        for(int i = n-1; i >= 0; i--){
            if(arr[i] == x){
                lastOcc = i;
                break;
            }
        }
        ans.add(firstOcc);
        ans.add(lastOcc);
        return ans;
    }
}
```

To exit full screen

O(n) + O(n)

Time Complexity

Best case = O(1)

Average case = O(n)

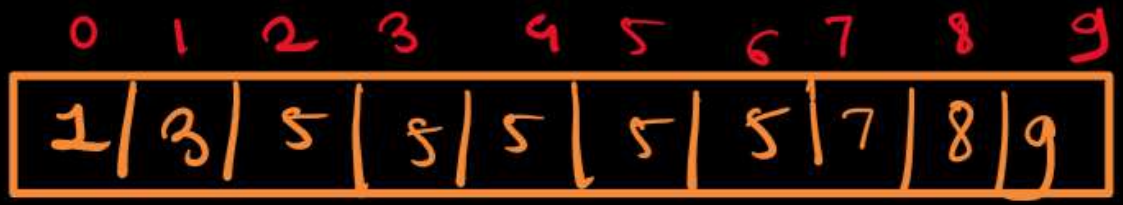
Worst case = O(n)

TC = O(n) + O(n)

O(2n)

O(n)

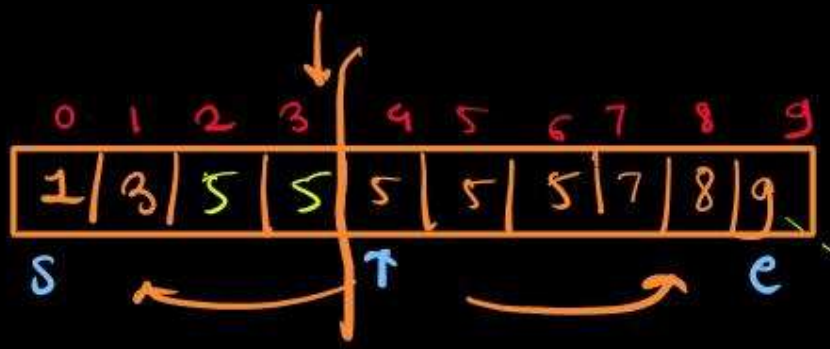
Approach 2



x = 5

FirstOcc = ?
lastOcc = ?

```
while (s<=e){
    mid = s+e/2;
    if (arr[mid] == x) return true;
    if (arr[mid] < x) s = mid+1
    else  e = mid - 1.
```

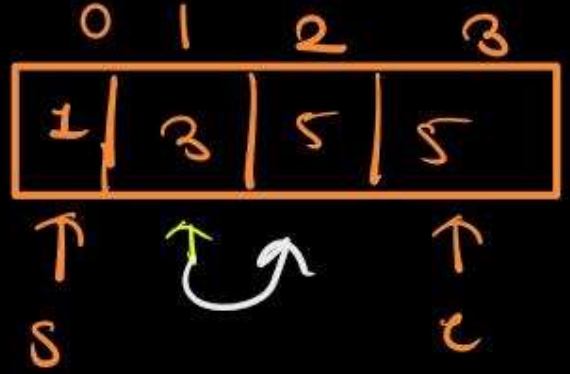


x = 5

mid = 0+9/2 = 4

firstOcc = 4

```
if (arr[m]==x) {
    firstOcc = m
    e = mid-1
}
```



firstOcc = 4

x = 5

$mid = 0+3/2 = 1.$

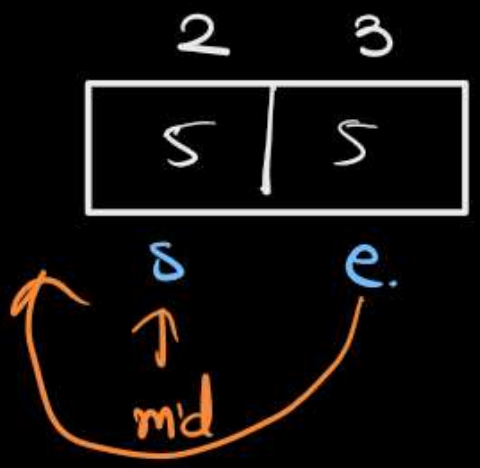
(middle == x)

s s

if (mid ele < x)

s = mid+e

else e = mid-1



$mid = 2+3/2 = 5/2 = 2$

firstOcc = 4

2,

if (mid ele == x) {

firstOcc = mid

e = mid - 1

}

s = 2, e = 1

```
FOcc = -1
while( s<=e){
   mid = s+e/2.
    if(arr[mid] == x){
       focc = mid
       e = mid -1
    }
    else if( arr[mid] < x) {
          s = mid+1;
    }else{
          e = mid-1;
    }
}
```

Last Occurence

lastOcc = ~~4~~ ~~5~~ 6

mid = 0+9/2 = 4

if (mid ele == x)
lastOcc = mid
s = mid+1

mid = 5+9/2 = 14/2 = 7

if (mid ele > x) {
e = mid-1
}

mid = 5+6/2 = 11/2 = 5

if (mid ele == x)
lastOcc = mid
s = mid+1
}

mid = 6+6/2 = 6

LO

while (s <= e) {

mid = s + e / 2

if (arr[mid] == x) {

ho = mid

s = mid + 1

}

else { — same as BS

}

```java
class GFG {
    int bsFirstOcc(int arr[], int x)
    {
        int start = 0;
        int end = arr.length - 1;
        int firstOcc = -1;
        while(start <= end){
            int mid = (start + end) / 2;
            if(arr[mid] == x){
                firstOcc = mid;
                end = mid - 1;
            }
            else if(arr[mid] < x){
                start = mid + 1;
            }
            else{
                end = mid - 1;
            }
        }
        return firstOcc;
    }
    int bslastOcc(int arr[], int x)
    {
        int start = 0;
        int end = arr.length - 1;
        int lastOcc = -1;
        while(start <= end){
            int mid = (start + end) / 2;
            if(arr[mid] == x){
                lastOcc = mid;
                start = mid + 1;
            }
            else if(arr[mid] < x){
                start = mid + 1;
            }
            else{
                end = mid - 1;
            }
        }
        return lastOcc;
    }
    ArrayList<Integer> find(int arr[], int x) {
        // code here
        int firstOcc = bsFirstOcc(arr, x);
        int lastOcc =   bslastOcc(arr, x);
        return new ArrayList<Integer>(Arrays.asList(firstOcc, lastOcc));
    }
}
```