"श्री कृष्ण गोविंद हरे मुरारी, हे नाथ नारायण वासुदेवा"

Data Structure and Algorithm

By

Malay Tripathi

BINARY SEARCH 3

FIRST AND LAST OCCURENCES IN ARRAY + COUNT OCCURENCES

Q. find the first and last occurence of element x.

arr = [2, 4, 6, 8, 8, 8, 11, 13]

ind → 0 1 2 3 4 5 6 7

So, we have to find first and last occurence index of x.

# $x=8$

{3, 5}

first index of occu. — last index of occu.

# $x=10$

{-1, -1}

first index of occu. — last index of occu.

# $x=11$

{6, 6}

first index of occu. — last index of occu.

# LINEAR SEARCH

arr = [2 4 6 8 8 8 11 13]

0 1 2 3 4 5 6 7

→ when we find $x=8$

first = ~~-1~~ last = -1

3 ------> last = ~~-1~~ ~~3~~ ~~4~~ 5

3 ------> 5

{3, 5}.

```
first = -1, last = -1
for(i=0 to n-1)
{
    if (arr[i] == x)
    {
        if (first == -1)
            first = i
        last = i
    }
}
```

$\boxed{\text{TIME COMPLEXITY} = O(N)}$

“श्री कृष्ण गोविंद हरे मुरारी, हे नाथ नारायण वासुदेवा”

Data Structure and Algorithm

By

Malay Tripathi

As we know that the given array is Sorted, we can apply BinarySearch for first and last Occurence.

```
       0  1 2 3 4 5  6  7
arr=[ 2, 4,6, 8,8,8 ,11, 13]
              ↑      ↑
```

as we know about upperbound and lowerbound.

lower bound → that it is smallest index , A[index] >= x

upper bound → A[index] > x

given me lowerbound of (8) ⟶ A[ind] ⩾ n → it will point at index 3

give me upperbound of (8) → A[index] > n → it will point at index 6

```
arr = [ 2,4, 6, 8,8,8, 11, 13]
                ↑    ↑   ↑
          lowerbound  |  upperbound
        [upperbound(8) -1]
```

# it will not covering all the cases, if the value is not in array

for $\boxed{x=10}$

since 10 is not in array, you will get 11 as the lowerbound.

$\boxed{x=14}$ lb(14)

⟶ it is start pointing hypothetical index of 8.

if((lb == n || (arr[lb] != x)) return {-1, -1};

means element is not present in the array.

return {lb , upperBound (arr, n,k)-1 } ;

TIME COMPLEXITY = $\underset{\text{lower bound}}{\underset{\uparrow}{O(\log_2 n)}} + \underset{\text{upper bound}}{\underset{\uparrow}{O(\log_2 n)}}$

$$\boxed{\text{TIME COMPLEXITY} = 2 * O(\log_2 n)}$$

APPROACH 2

USE SIMPLE BINARY SEARCH

$$arr = \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [2, & 8, & 8, & 8, & 8, & 8, & 11, & 13] \\ \uparrow & & & \uparrow & & & & \uparrow \\ & & & mid & & & & \end{array}$$

first = -1 ← - - - - mid

$\therefore$ mid = $\frac{0+7}{2} = 3.5 \simeq 3$

first = ~~-1~~ 3 , so for finding the first index, look for the left hand side. So right search space will be eliminated.

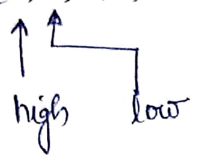
$$arr = \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [2, & 8, & 8, & 8, & 8, & 8, & 11, & 13] \\ \uparrow & \uparrow & \uparrow & & & & & \\ low & mid & high & & & & & \end{array}$$

first = ~~-1~~ ~~3~~ 1

$$arr = \begin{array}{cccccccc} [2, & 8, & 8, & 8, & 8, & 8, & 11, & 13] \\ \uparrow\uparrow & & & & & & & \\ low\ high & & & & & & & \end{array}$$

→ this element is not equal to x, thus we have to move in the right search space.

arr = [2, 8, 8, 8, 8, 8, 11, 13]



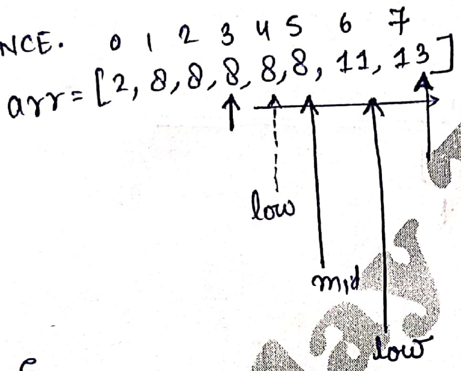
→ low crosses high, thus first = ~~-1~~ ~~3~~ 1

$\boxed{\text{FIRST OCCURENCE} = 1}$

// FIND LAST OCCURENCE.

last = -1

Last = ~~-1~~ 3



* mid = $\frac{4+7}{2}$ = 5

a[mid] = 8 = so, last = ~~-1~~ ~~3~~ 5

* mid = $\frac{6+7}{2}$ = 6

a[mid] = a[6] = 11

8 ≠ 11. So we can't find, 8 on the right.

$\boxed{\text{last occurrence} = 5}$

“श्री कृष्ण गोविंद हरे मुरारी, हे नाथ नारायण वासुदेवा”

Data Structure and Algorithm

By

Malay Tripathi

Pseudo Code for first Occurence.

```
fn (arr, n, x)
  {
  low=0, high=n-1
   first=-1
  while( low<= high)
    {
     mid = (low + high)/2
    if (a[mid] = = x
        first = mid , high = mid-1

    else if ( a[mid] < n)
          low= mid +1

     else
        high= mid-1
```

Pseudo Code for last Occurence.

```
fn(arr, n, x)
  {
   low =0 , high=n-1;
   last =-1;
  while (low<=high)
   {
   mid=( low+high)/2
   if (arr[mid] = =n)
      last = mid;
      low= mid +1;
   else if ( a[mid]<x)
         low = mid+1
   else
      high= mid-1;
```

```
pair <int,int> firstANDLastPosition (vector<int> & arr , int n, int k)
  { int first = firstOccurence (arr, n, k);
    if (first== -1) return {-1,-1};
    int last = lastOccurence (arr, n, k);
    return ( first, last);
  }
```

"श्री कृष्ण गोविंद हरे मुरारी, हे नाथ नारायण वासुदेवा"

Data Structure and Algorithm

By

Malay Tripathi

COUNT TOTAL NUMBER OF OCCURENCE OF X.

(last occurence - first occurence) + 1 = total no. of x element in the array.

```
int count (vector <int> & arr, int n, int x) {
    pair <int,int> ans = firstAndLastPosition (arr, n, x);
    if(ans.first == -1) return 0;
    return ans.second - ans.first + 1;
}
```