Week No 4

classmate
Date
Page 104

Searching & Sorting - Level - 1

(9MIVVVV)

Linearly Searching

Time Complexity → O(N)

Space complexity → ~~O(N)~~ O(1)

Binary searching conditions :-

Monotonic → sorted in Ascending or descending.

Time complexity of B.S ⇒

$n \rightarrow n/2^0 \rightarrow n/2^1$

$\downarrow$

$n/2^2$

$\downarrow$

$n/2^3$

So on

$\downarrow$

For K^{th} iteration, $n/2^K$

$$\frac{n}{2^K} = 1$$

$$n = 2^K$$

$$\log_2 n = K$$

T.C → $O(\log_2 N)$

Dry Run ⇒

0	1	2	3	4	5	6	7	8
10	20	30	40	50	60	70	80	90

↑ mid

s = 0, e = 8

mid = $\frac{0+8}{2}$ = 4

target = 90

90 > 50

5	6	7	8
~~50~~ 60	70	80	90

↑ mid

s = 5

e = 8

mid = $\frac{13}{2}$ 90 > 70

mid = 6

target = 90

7	8
80	90

↑

s = 7

e = 8

mid = $\frac{15}{2}$

mid = 7

90 > 80

8
90

s = 8

e = 8

mid = $\frac{8+8}{2}$ = 8

90 == 90 ⇒ True.

Rules

(1) Found
if(arr[mid] == target)
return mid
(2) if(target > arr[mid])
s = mid + 1
(3) if(target < arr[mid])
e = mid - 1

(Q1)

Code:

```
int binary Search (int arr[], int n, int target){
    int start = 0;
    int end = n-1;
// There is some flaw in below line, HW
    int mid = (start + end)/2;
    while(start <= end){
        // found
        if(arr[mid] == target){
        // return index of the found element
            return mid;
        }
        else if(target > arr[mid]){
            // Right mejao
            start = mid + 1;
        }
        else if(target < arr[mid]){
            // left mejao
            end = mid - 1;
        }
        // mid update
```

mid = (start + end) / 2;

}

// agar yaha tk pohohche ho
//iske matlab element nahi mila tumhe
return -1;

}

Q.2) Find First Occurence of a number in a sorted Array.

	10	20	30	30	30	30	40	50
arr →	0	1	2	3	4	5	6	7

ans = 2
target = 30

Code:

```
int findFirstOccurence (int arr[], int n,
                        int target) {
  int s = 0;
  int e = n-1;
  int mid = (s+e) / 2;
  int ans = -1;

  while (s<=e) {
    if (arr[mid] == target) {
      // ans store
      ans = mid;
      // left me jao
      e = mid-1;
    }
```

```
else if (target > arr[mid]) {
    //right me jao
    s = mid+1;
}
else if (target < arr[mid]) {
    //left me jao
    e = mid-1;
}
//galti yaha krte h hmesha
mid = (s+e)/2;
}
return ans;
}
```

Best Practice ⇩

```
int mid = s+(e-s)/2;
```

Q.3 Find last occurence → sorted Array.

10 | 20 | 30 | 30 | 30 | 30 | 30 | 30 | 40 | 50

Right me jao after getting ans.

Code:

```
int findLastOccurence (int arr[],
int n, int target) {
    int s = 0;
    int e = n-1;
    int mid = s+(e-s)/2;
    int ans = -1;
```

```
while (s<=e) {
    if (arr[mid] == target) {
        //ans store
        ans = mid;
        //right me jao
    } s = mid+1;
    else if (target > arr[mid]) {
        //right me jao
        s = mid+1;
    }
    else if (target < arr[mid]) {
        //left me jao
        e = mid-1;
    }
    //galti yaha karte h hmesha
} mid = (s+e)/2;
}
return ans;
}
```

Q.4) Find Total occurerence.

10	20	30	30	30	30	40	50
0	1	2	3	4	5	6	7

target = 30

Total occurence = 4

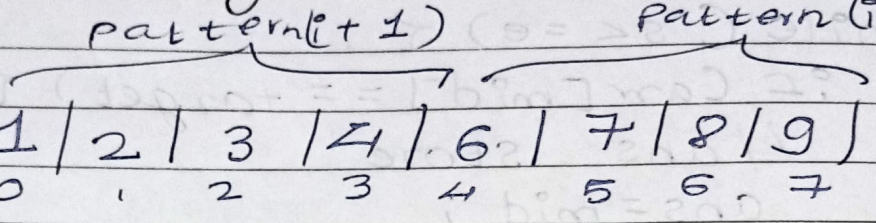
How?

First occurrence = 2

last occurrence = 5

Last Occurrence - firstOccurence + 1

(Q.5) Find missing element in a sorted Array



1 ⟶ n

ans → 5

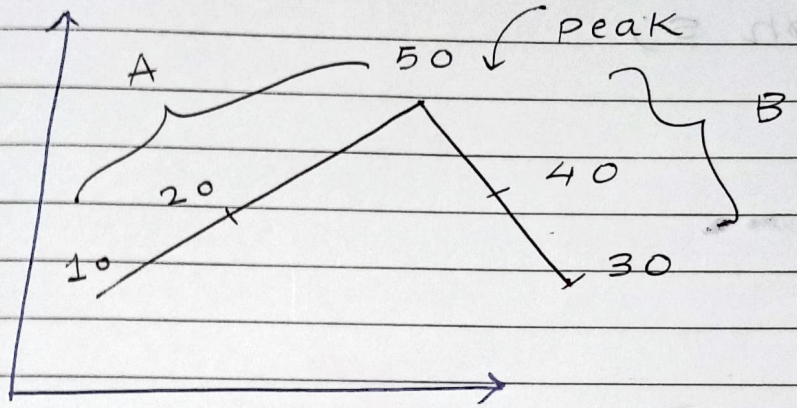
Code:

```
int findMissingNumber(int arr[], int n){
    int s=0;
    int e= n-1;
    int mid= s+(e-s)/2;
    int ans= -1;
while(s<=e){
    int diff = arr[mid]-mid;
    if(diff==1){
        //right me jao
        s= mid+1;
    }
    else{
        // ans store
        ans = mid;
        // left me jao
        e= mid-1;
    }
    mid = s+(e-s)/2;
}
// H.W → How can we remove this,
// by modifying the above logic.
    if(ans+1==0){
        return n+1;
    }
return ans+1;
```

Peak in a Mountain Array

I/P ⇒ {10, 20, 50, 40, 30}

ans = 50 ⇒ O/P



(A) arr[i] < arr[i+1]
(B) arr[i] > arr[i+1]
(C) arr[i-1] < arr[i] > arr[i+1]
↑ Peak

Code:

```
int peakIndexInMountainArray(vector<int>&arr){
    int n = arr.size();
    int s = 0;
    int e = n-1;
    int mid = s+(e-s)/2;
while(s<e){
    if(arr[mid]<arr[mid+1]){
//A wali line me ~~hai~~ hu
//Peak right me exist krti h
        s = mid+1;
}
else{
    //yaa toh main B line pr hu
    //ya toh main peak element pr hu.
```

H.W ⇒ Find pivot element.


```
        } e=mid;
        //mid Update
        mid = s+(e-s)/2;
    }
    return s;
}
};
```