# SEARCHING & SORTING [LEVEL-3]

Date 20/09/23

## # Questions on search space :-

Q1. We have 2 numbers as input and we have to divide them using Binary search. Return the integer value of quotient.

Ex:- i/p → $\frac{29}{7}$ o/p → 4

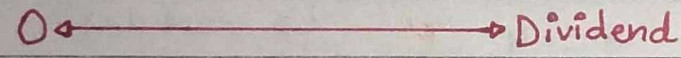
Divisor ) Dividend ( Quotient

Remainder

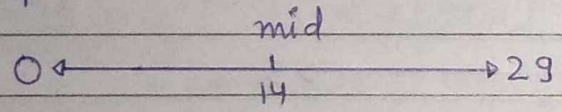
↳ Quotient * Divisor + Remainder = Dividend

**Quotient * Divisor ≤ Dividend**

Search space for possible answers will be -




Ex:- i/p → $\frac{29}{7}$ → dividend




s=0, e=29, mid=14

↳ Is this a possible ans?

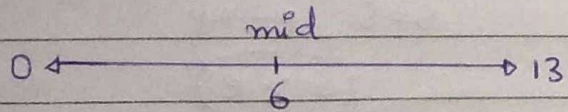
Quotient x divisor <= Dividend

14 x 7 = 98 ≠ 29

98 > 29 → false

↳ Left me jao → e = mid-1

e = 14-1 = 13



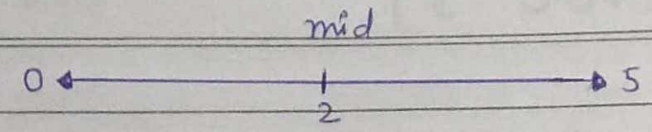

s=0, e=13, mid = $\frac{0+13}{2}$ = 6

Q x Divisor ≤ dividend

6 x 7 29

42 > 29 → Left → e = mid-1 = 6-1 = 5

Date ..../..../.....




$s=0, e=5, mid=\frac{0+5}{2}=2$

Q. x divisor ≤ dividend

2 x 7 <= 29

14 <= 29 → Valid ans

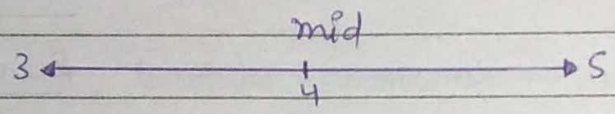
ans = 2

↳ ans store

↳ Right

↳ s = mid + 1

s = 2 + 1 = 3




$s=3, e=5, mid=\frac{3+5}{2}=\frac{8}{2}=4$

Q x divisor ≤ dividend

4 X 7 ≤ 29

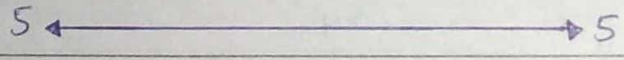
28 ≤ 29 → Valid Ans

ans = 4

↳ store ans.

↳ Right → s = mid + 1

s = 4 + 1 = 5




$s=5, e=5, mid=5$

Q x divisor ≤ dividend

5 x 7 ≤ 29

35 ≤ 29 → False

↳ Left → e = mid - 1

e = 5 - 1 = 4

s = 5, e = 4

(s > e)

↳ STOPPING CONDITION

Date ..../..../.....

Conditions -

- Quotient x divisor == dividend
  - Q is final Ans
- Quotient x divisor < dividend
  - ans store
  - right
- Quotient x divisor > dividend
  - Left me jao

Code:-

```
int getQuotient (int divisor , int dividend){
    int s=0; int e = dividend;
    int mid= s +(e-s)/2;
    int ans =-1;
    while ( s <= e) {
        if ( mid * divisor == dividend){
                return mid;
        }
        else if (mid * dividend < dividend){
            // store ans
            // right
            ans = mid;
            s= mid+1;
        }
        else{
            // left
            e= mid-1;
        }
        // Update mid
        mid = s + (e-s)/2;
    }
    return ans;
}
```

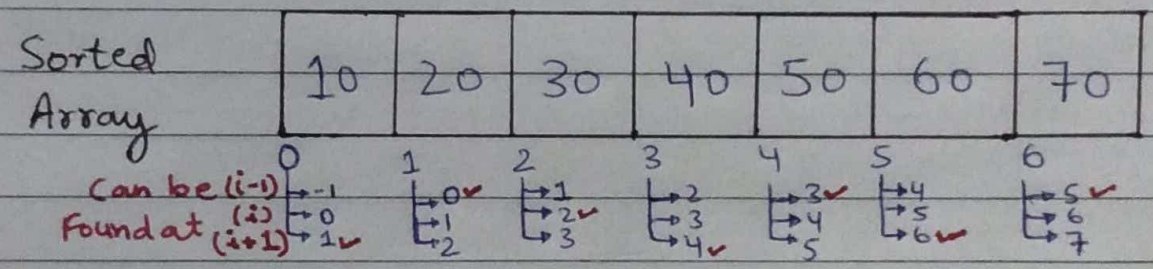
Date ..../..../.....

```
int main(){
    int dividend = 29;
    int divisor = 7;
    int ans = getQuotient(abs(divisor), abs(dividend));
    // now we need to decide the sign either +ve or -ve
    if((divisor > 0 && dividend < 0) || (divisor < 0 && dividend > 0)){
        ans = 0 - ans;
    }
    cout << "final ans is : " << ans << endl;
}
```

Output:-

Final ans is : 4

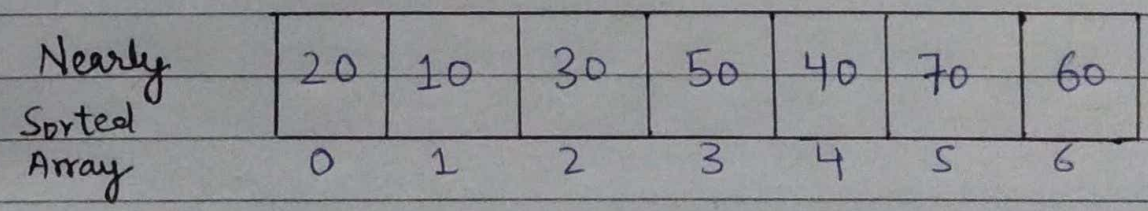
Q2. Binary Search on nearly sorted array.




If a no. is at $i^{th}$ index in sorted array

Then it can be present at any position among them in nearly sorted array

→ (i-1) index
→ (i) index
→ (i+1) index



Date ..../..../.....

## Binary Search in Normal sorted array

Conditions
↳ if(arr[mid] == target)
return mid

↳ if(target > arr[mid])
↳ Right

else
↳ Left

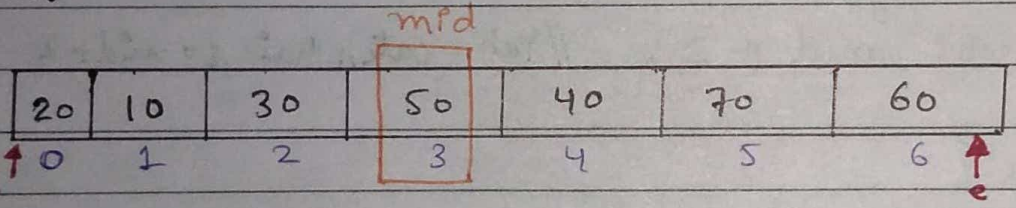
But here ans can be at
(mid-1)
(mid)
(mid+1)

## Binary Search in Nearly Sorted array

if(arr[mid] == target)
return mid;
if(arr[mid-1] == target)
return mid-1;
if(arr[mid+1] == target)
return mid+1;

if(target > arr[mid])
↳ Right → Catch → {Yaha on galti karung mai}
else
↳ Left → catch

Dry Run:-

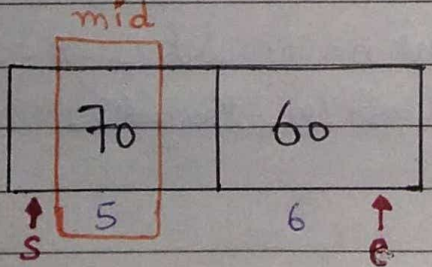



target = 70

s=0, e=6, mid = 3

arr[mid-1] == 70 → 30 == 70 → F
arr[mid] == 70 → 50 == 70 → F
arr[mid+1] == 70 → 40 == 70 → F

if (target > arr[mid])
70 > 50 → Right → s = mid + 1 ✗
s = mid + 2 (catch)
To skip the already checked number and same is applicable to Left wala case. ∴ e = mid-2 and not use mid-1 here.




s=5, e=6, mid = 5
arr[mid-1] Not exist
arr[mid] == 70 → 70 == 70 → True
return mid

So, s = 3+2 = 5

Date ..../..../.....

Code:-

```
int searchNearlySorted (int arr[], int n, int target) {
    int s=0; int e= n-1, int mid = s+(e-s)/2;
    while ( s<=e) {
        if ( arr[mid-1] == target)
                return mid-1;
        if ( arr[mid] == target)
                return mid;
        if (arr[mid+1] == target)
                return mid+1;

        if ( target > arr[mid]){
            // right
            s= mid + 2;   //Yaha catch hai s= mid+2
        }
        else{
            // Left
            e= mid - 2;   //Yaha catch hai e=mid-2
        }
        mid = s + (e-s)/2;
    }
    return -1;
}
int main(){
    int arr[]={10,20,30,50,40,70,60}; int n=7; int target=10;
    int targetIndex = searchNearlySorted (arr, n, target);
    if (targetIndex == -1)
      cout<<"Target Not found";
    else
      cout<<"Target found at index: "<< targetIndex;
}
```

Output:-

Target found at index: 0

Spiral

Date ..../..../.....

Imp Hard

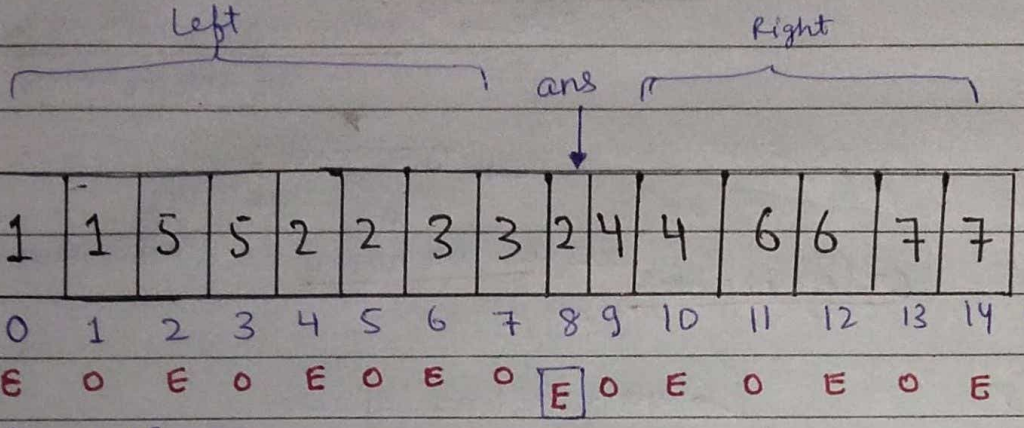
Q Find the odd occuring element.

↳ All elements are occuring even no. of times except one

↳ Even no. of elements are in pairs and pairs are not repeated.

↳ Ek baar me koi bhi no. 2 se jada baar nahi aa sakta

find that element which occurs odd times.




Types of B.S Questions

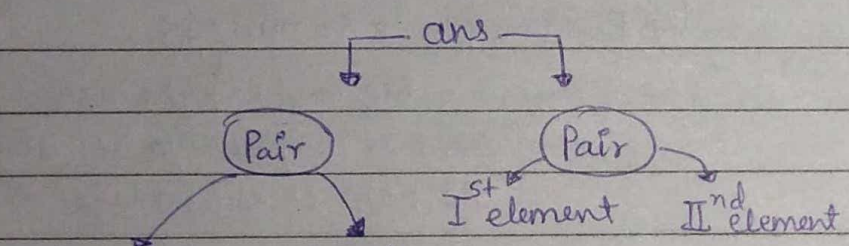
① Classical

② Search space

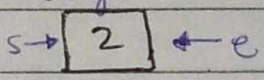
③ Predicate function

④ Index pe observation karke logic bnana

Observation :-



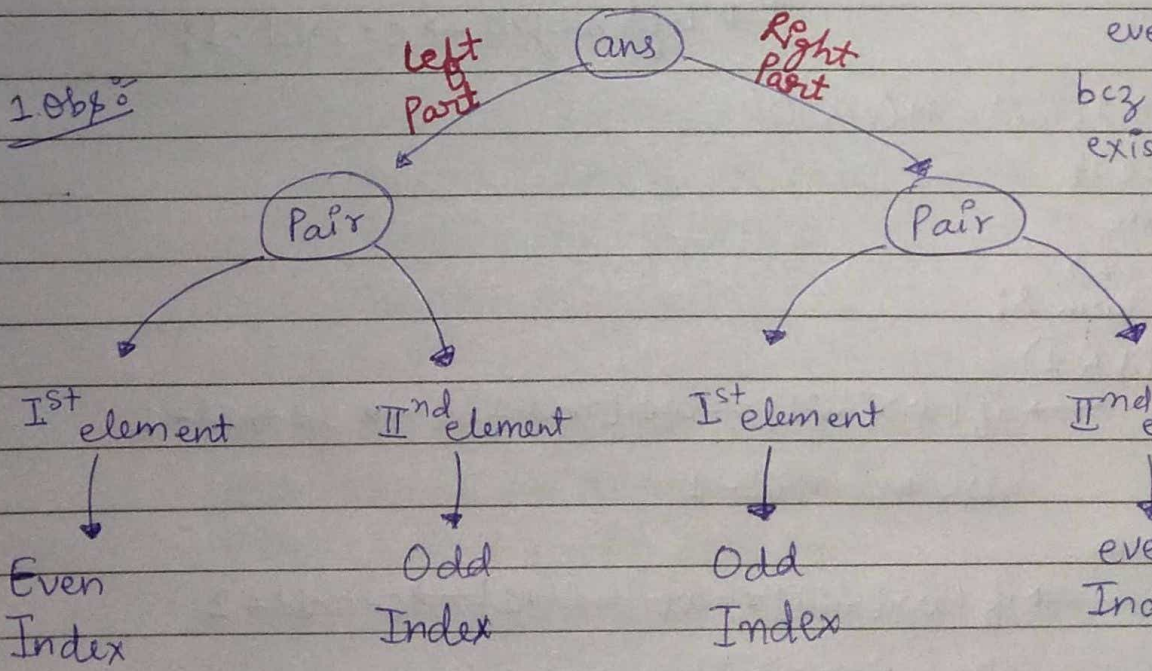

2 obs :- single element




if (s==e)
return s

3 obs :- We get ans at even index always bcz previous elements exist in pair of 2

1 Obs :-



Date ..../..../.....

logic

if (mid % 2 == 0) ⇒ Even index

→ if (arr[mid] == arr[mid+1])

abhi ans K left me hai

↳ Right me jao

↳ s = mid + 2 ★

already +1 check kr liya thats why +2

else → Ya toh ans K right me hai / ya ans pehle hai

↳ e = mid; ★

↳ why?

ans lost na hojaye

Or left bhi chhlejaye thats why.

{Peak in mountain}

if (mid % 2 == 1)

→ ans K left me hai abhi

→ if (arr[mid] == arr[mid-1]

↳ Right me jao → s = mid+1;

Bcz mid K piche wale element se compare kiya hai isliye +1 kr ke ek element aage jayenge.

else

↳ left me jao → e = mid - 1;

```
int s=0, e=n-1, mid = s+(e-s)/2
while (s<=e){
  // single case
  if (s==e)
      return s;
  if (mid & 1)
      → if (arr[mid] == arr[mid-1]) → s = mid+1
        else → e = mid - 1
  else → Even
      → if (arr[mid] == arr[mid+1] → s = mid + 2
        else → e = mid; → ans lost na ho
```

Spiral

Date ..../..../.....

Code:-

```
int findOddOccuringElement ( int arr[], int n){
    int s=0; int e=n-1; int mid = s+(e-s)/2;
    while (s<=e){
        if ( s==e) {
            return s;
        }
        //check mid → even or odd
        if(mid & 1) {  // mid&1 → True → odd no.
            if ( mid +1 < n  && arr[mid] == arr[mid+1]){
                    // Left me jao
                    e= mid - 1;
            }else {
                    // ans Ke left side me hai to
                    // right me jao
                    s = mid+1;
            }
    }
    else {
        // even no
        if (mid +1 < n  && arr[mid] == arr[mid +1]) {
                // mid+1 already checked so move towards
                // right using mid + 2
                s= mid + 2;
        }else {
            // Ya toh mai right part par Khda hu
            // Ya toh mai ans Ke upper Khda hu
            // that's why e = mid Krra hu
            // Kyuki e = mid - 1 se ans lost ho skta hai
            e= mid;
        }
    }
}
```

Spiral

Date ..../..../.....

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

    return -1;
}

int main(){
    int arr[] = {20, 20, 5, 5, 3, 3, 1};
    int n = 7;
    int ans = findOddOccuringElement(arr, n);
    cout << "Final ans is: " << arr[ans];
}
```

Output:-

Final ans is 1