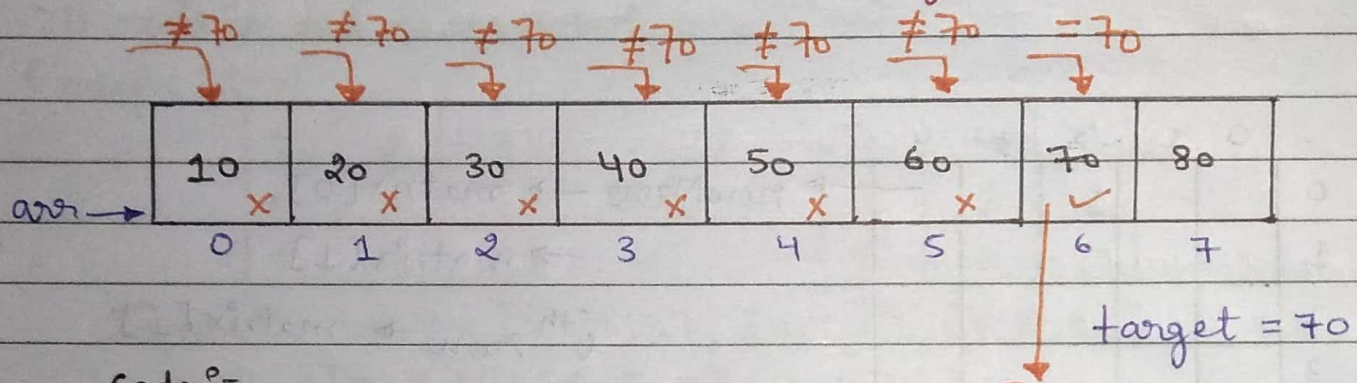
SEARCHING && SORTING
[LEVEL-1]

Date 13/09/23

1. Linear Search :-

It starts searching from one end of a list and goes through each element until the desired target found, otherwise the search continues till the end.

ek ek karke sabko check krdungi



Code :-

```
for ( int i=0; i<n; i++)
{
    if ( arr[i] == target)
        return true;
}
return false;
```

→ Time Complexity → O(n) → Size of array

2. Binary Search :-

Binary search is only applicable on sorted arrays which can either in ascending order or in descending order.
We can apply it on monotonic functions i.e given content must be in either ascending order / increasing order or in decreasing order.

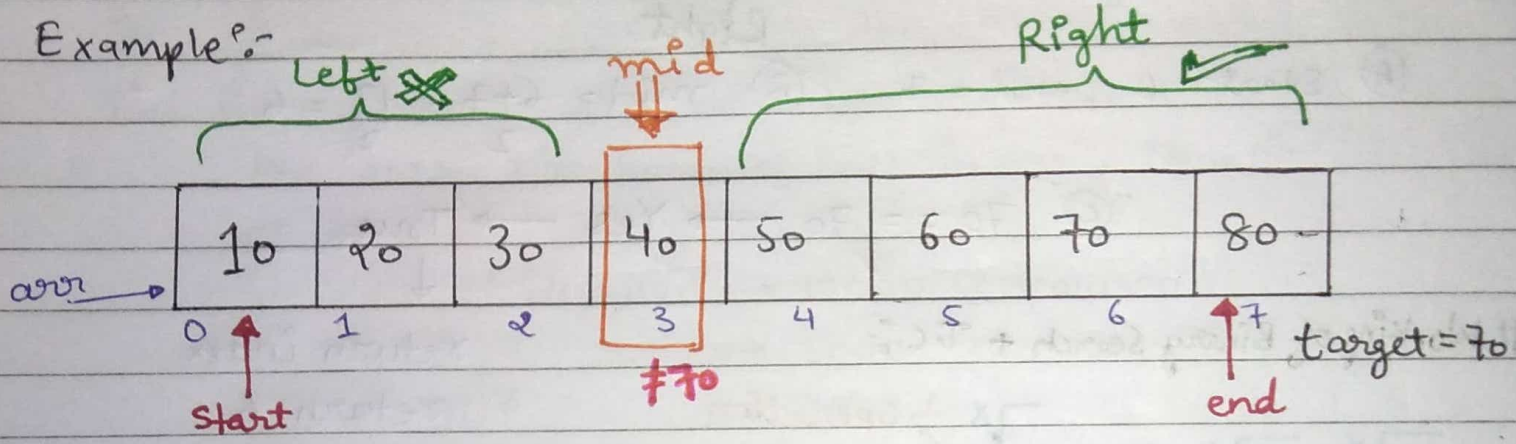
Approach :-

(A) set start and end index with 2 variables
(B) Calculate their mid i.e $mid = \left(\frac{s+e}{2}\right)$
(C) Check the condition and select one part of (Sorted) array and neglect the other.

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

BEST PRACTICE — Use start + (end - start) / 2 because mid = (start + end) / 2 may arise condition of integer overflow so, be on safer side.

Example:-



(A) Set start, end

(B) Calculate mid = (s+e)/2

start = 0, end = 7

mid = (0+7)/2 = 7/2 = 3

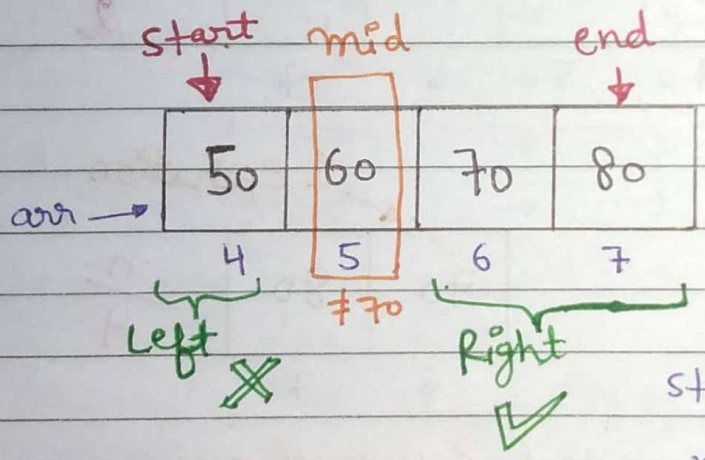
(C) Check the condition

(D) Select one part and neglect the other

40 == 70 → F

70 > 40 → select right side array

↳ Neglect the left side array



start = 4, end = 7

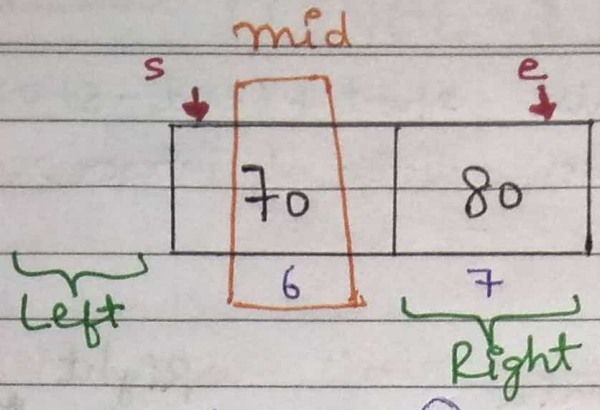
mid = (4+7)/2 = 11/2 = 5

60 == 70 → F

70 > 60 → Select Right

↳ Neglect Left

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



(A) Start = 6 , end = 7 (B) mid = $\frac{6+7}{2} = \frac{13}{2} = 6$

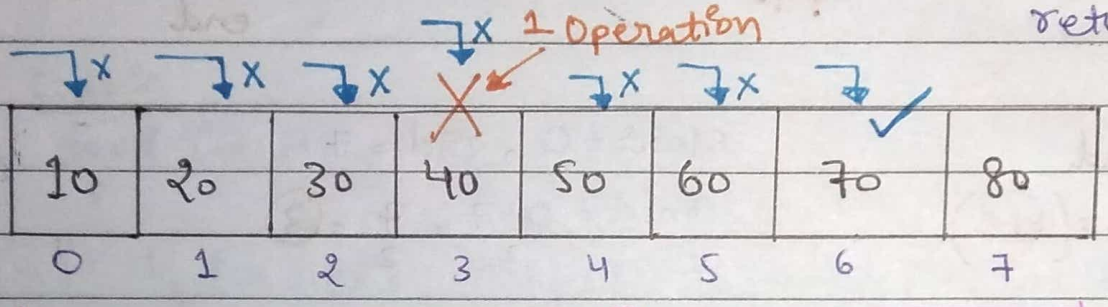
(C) 70 == 70 → Yes → True

↓

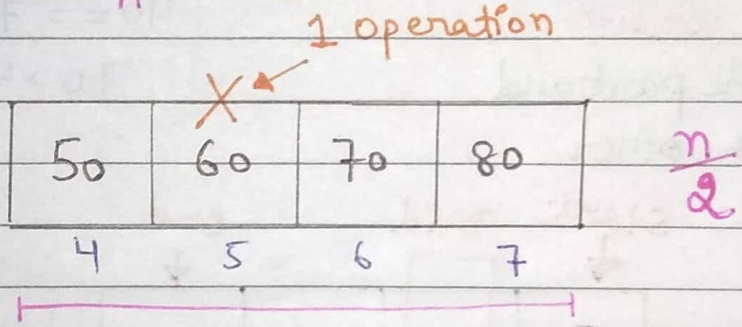
return index

return 6

Working of Binary Search + T.C :-

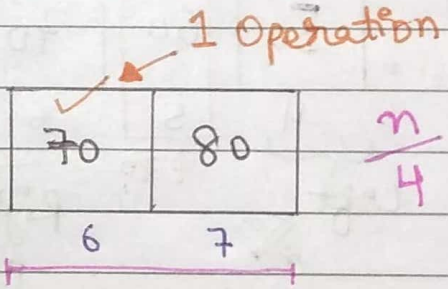


Size = n



★ OBSERVATION :-

With every iteration, array size becomes half in B.S



Binary Search took 3 operations

↳ To tell target found/not found

Linear Search took 7 Search operations, for this specific example.

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

Ex:- If we have an array of 10000 elements and we are finding a target which is not present in this array.

If we use Linear Search, in this case it took 10000 comparisons

But Binary Search took very less comparisons
bcz array size becomes half in every iteration.
So,

I st iteration	→ 10000	comparison
II nd	→ 5000	13 comparisons
III rd	→ 2500	
IV th	→ 1250	
5 th	→ 625	
6 th	→ 312	
7 th	→ 156	
8 th	→ 78	
9 th	→ 39	
10 th	→ 19	
11 th	→ 9	
12 th	→ 4	
13 th	→ 2	
14 th	→ 1	

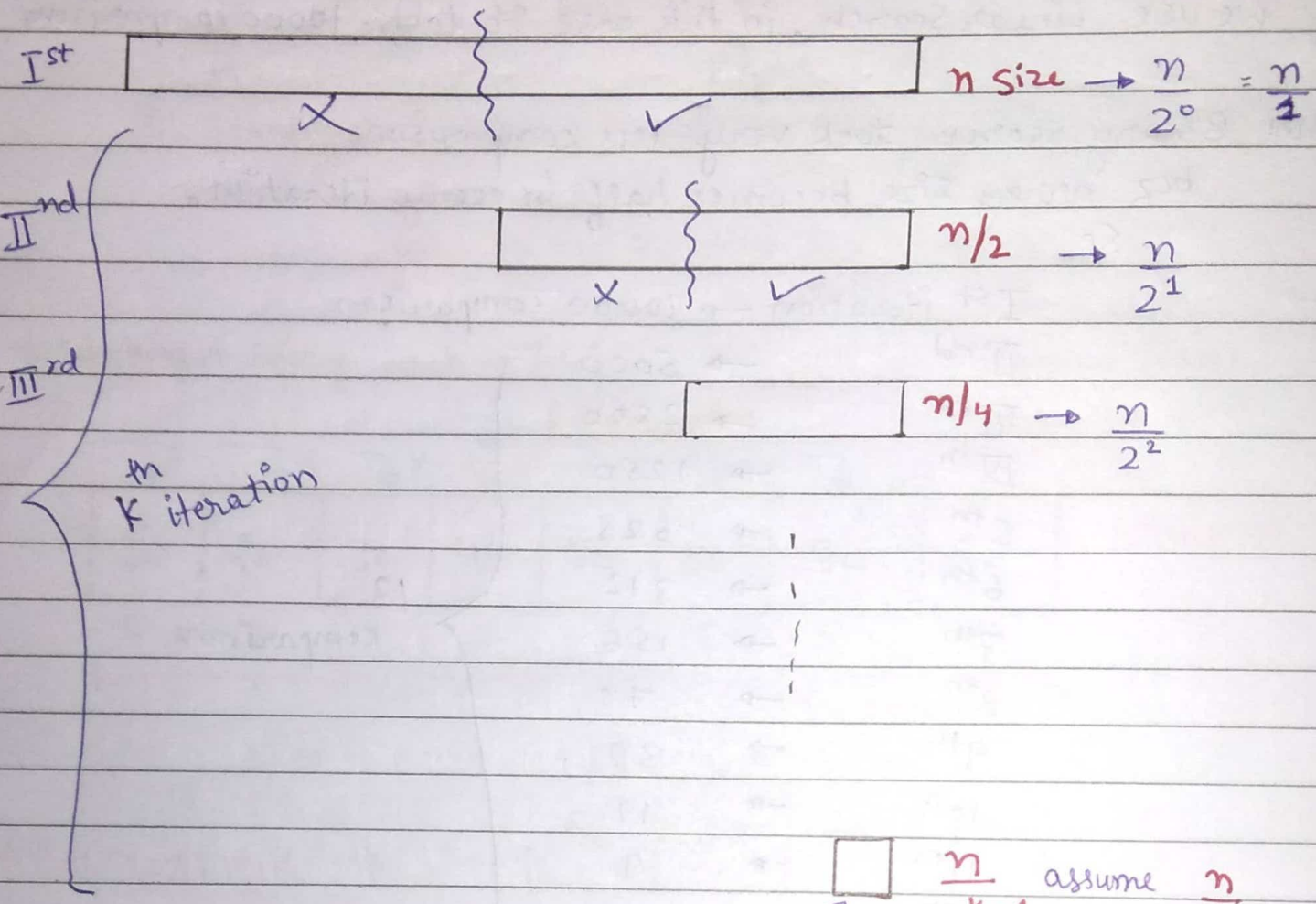
∵ $\log_2(10000)$

Time complexity of Binary Search = $\log_2(n)$

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

Time Complexity of Binary Search :-

↳ In every iteration, size becomes half.



∴ We can say that,

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

$$n = 2^k$$

$$\log_2 n = k \quad \text{(Taking log both sizes)}$$

T.C of B.S → $O(\log_2 n)$

$n = 2^k$

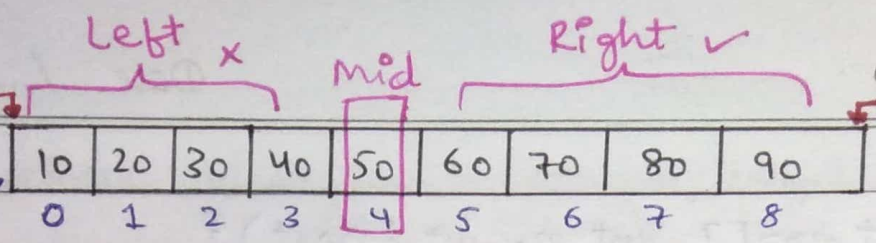
$\log_2 n = \log_2 (2^k)$

$\boxed{\log n = k}$

Spiral

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

Ex:- arr

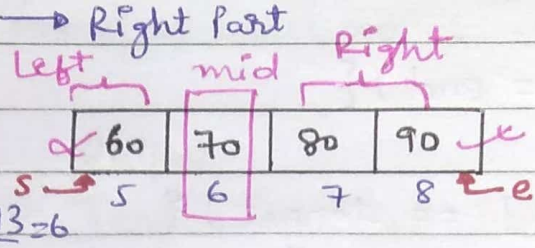


target = 90

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

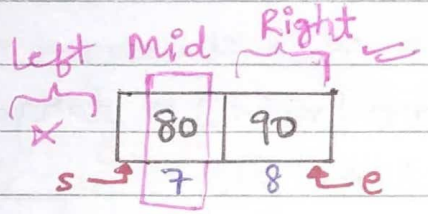
$50 == 90 \rightarrow F$

$90 > 50 \rightarrow$ Right Part



$s = 5, e = 8, mid = \frac{5+8}{2} = \frac{13}{2} = 6$

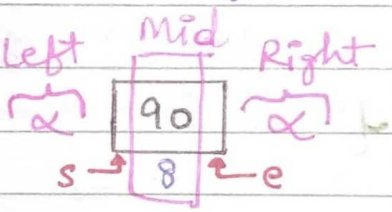
$70 == 90 \rightarrow F \Rightarrow 90 > 70 \rightarrow$ Move towards Right



$s = 7, e = 8, mid = \frac{7+8}{2} = 7$

$80 == 90 \rightarrow F$

$\Rightarrow 90 > 80 \rightarrow$ Move towards Right



$s = 8, e = 8, mid = \frac{8+8}{2} = \frac{16}{2} = 8$

$90 == 90 \rightarrow$ True $\Rightarrow$ Return 8

Rules for applying Binary search:-

① found wala case

```
if (arr[mid] == target)
    return mid;
```

② Not found wale ke case honge

③ ☆ Kabtak apply hoga B.S

while (s <= e)

OR agar (s > e) To bhi Ruk Jao

(i) if (target > arr[mid])

```
s = mid + 1
```

(ii) if (target < arr[mid])

```
e = mid - 1
```

Spiral

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

Code:-

```
int binarySearch (int arr[], int n, int target) {
    int start = 0;
    int end = n-1;
    int mid = start + (end - start)/2;

    while (start <= end) {
        // Found
        if (arr[mid] == target) {
            // return index of found element
            return mid;
        }
        else if (arr[mid] < target) {
            // Right me jao
            start = mid + 1;
        }
        else if (arr[mid] > target) {
            // Left me jao
            end = mid - 1;
        }
        // yaha galti nhi Krni hamesha mid update Krna hai
        mid = start + (end - start)/2;
    }
    // Agar yaha tak phoche matlab target nhi mila so
    // return invalid index
    return -1;
}
int main() {
    int arr[] = {10, 20, 30, 40, 50, 60, 70, 80, 90};
    int n = 9;
    int target = 70;
```

Spiral

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

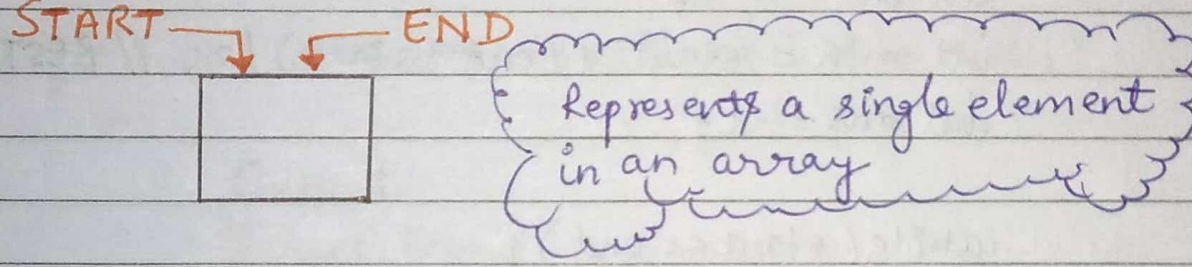
```
int ansIndex = binarySearch (arr, n, target);

if (ansIndex == -1)
        cout << " Target Not found " << endl;
else
        cout << " Target found at index no. : " << ansIndex << endl;
return 0;
}
```

Output

Target found at index no.: 6

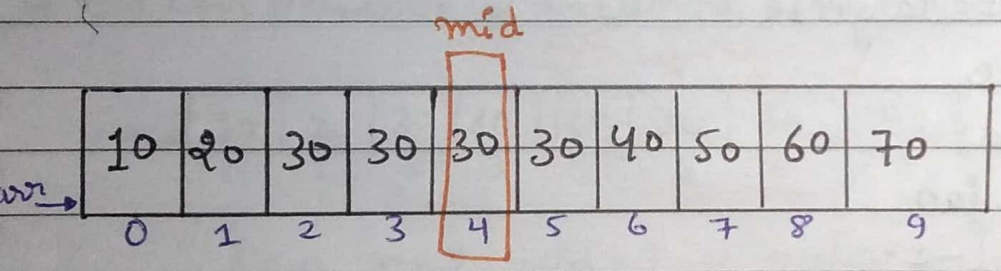
KEEP IN MIND

"START == END" → represents an array whose size is 1.



Q Find first occurence of a number in a sorted array.

Think about B.S if we want to search an item.



target = 30
ans = 2 [Expected Answer]

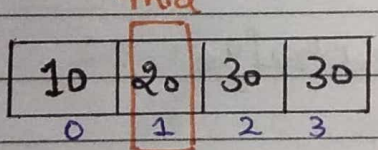
s = 0, e = 9, mid = 4, arr[mid] == target

30 == 30 → True → found

ans = 4

left → e = mid - 1

e = 4 - 1 = 3



s = 0, e = 3, mid = 1

⇒ 20 == 30 → F

30 > 20 → Right → s = mid + 1

s = 1 + 1, s = 2

Rule :-

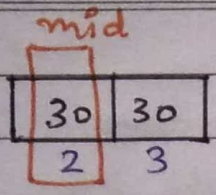
Found kaha

→ ans store then

→ Left me chale jana i.e

end = mid - 1

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



s = 2, e = 3, mid = 2

30 == 30 → Found → ans = 2

Left me jao

e = mid - 1

e = 2 - 1

e = 1

s = 2, e = 1

s > e → Rukjao STOP!!

Code:-

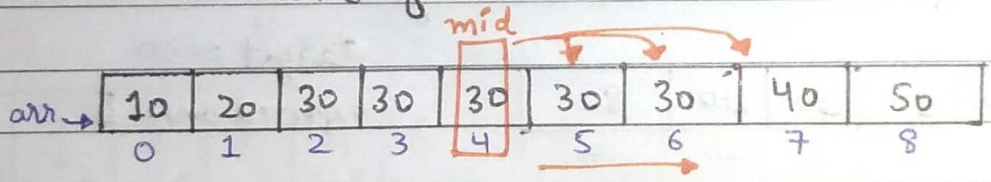
```
int findFirstOccurence(int arr[], int n, int target){
    int start = 0;
    int end = n-1;
    int mid = start + (end - start) / 2; // BESTPRACTICE
    int ans = -1;

    while( start <= end ){
        //found
        if( arr[mid] == target){
            //ans store
            ans = mid;
            // Left me jao
            end = mid - 1;
        }
        else if (arr[mid] > target) {
            // Left me jao bcz 1's occurence chaiye
            end = mid - 1;
        }
        else if (arr[mid] < target) {
            // right me jao
            start = mid + 1;
        }
```

```
    // Galti yaha karte hai hamesha
    mid = start + (end-start)/2;
  }
  return ans;
}
int main(){
  int arr[] = {10, 20, 30, 30, 30, 30, 70, 80, 90};
  int n = 9;
  int target = 30;
  int ansIndex = findFirstOccurence(arr, n, target);
  if(ansIndex == -1)
      cout<<"Target not found"<<endl;
  else
    cout<<"Target found at index no. :"<<ansIndex<<endl;
  return 0;
}
```

Output

Target Found at index no. : 2

Q Find Last occurence of an element in a sorted array.



s = 0, e = 8, mid = 4 => arr[mid] == target

30 == 30 → True → Found → (i) ans store

(ii) Right me jao

↳ s = mid+1

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){
  // Found
  if(arr[mid] == target){
```

```
        // store ans
        ans = mid;
        //right me jao
        s = mid+1;
      }
      else if (arr[mid] < target){
        //right me jao
        s = mid+1;
      }
      else if (arr[mid] > target){
        //Left me jao
        e = mid-1;
      }
      //yaha galtinhi krni yaad rkho update mid
      mid = s + (e-s)/2;
    }
    return ans;
}
```

Q Find total occurence of an element in sorted array.

10	20	30	30	30	30	30	80
0	1	2	3	4	5	6	7

target = 30
Expected ans = 5

First occurence = 2
Last Occurence = 5

Total Occurence = Last Occurence - first Occurence + 1

Code:-

```
int findTotalOccurence (int arr[], int n, int target) {
    int total;
```

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

```
int lastOccurence = findLastOccurence(arr, n, target);
int firstOccurence = findFirstOccurence(arr, n, target);
total = lastOccurence - firstOccurence + 1;
return total;
}
int main(){
int arr[] = {10, 20, 30, 30, 30, 30, 30, 40};
int n = 8;
int target = 30;
int total = findTotalOccurence(arr, n, target);
cout << "Total occurence is : " << total << endl;
return 0;
}
```

Output :-
Total Occurence is : 5

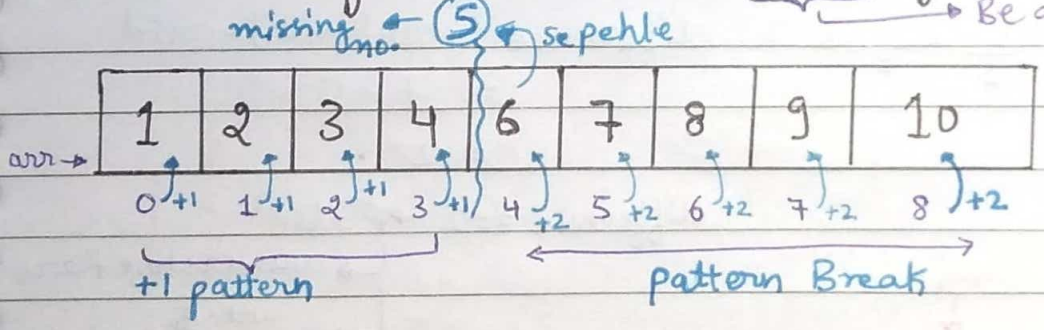
★ Binary Search ke Advance Questions are based upon understanding of firstOccurence and lastOccurence questions. Ki agar FOUND wali condition hai to RIGHT me jao, answer store karo, etc.

Q Find missing element in a sorted array.

→ Be attentive to think about B.S.

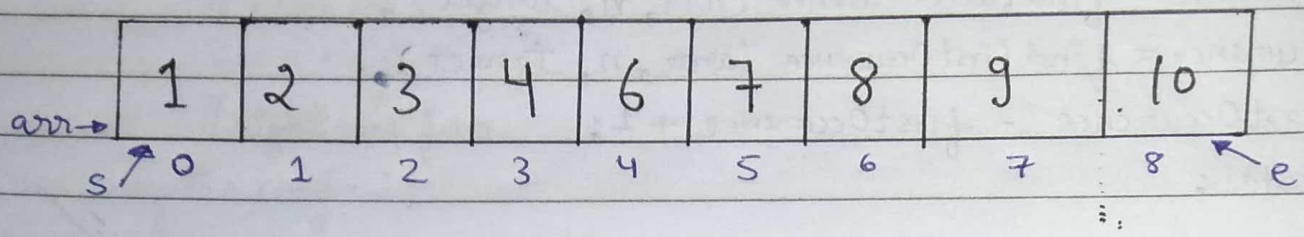
given → 1 to n no.'s

ans = 5 (missing no.)

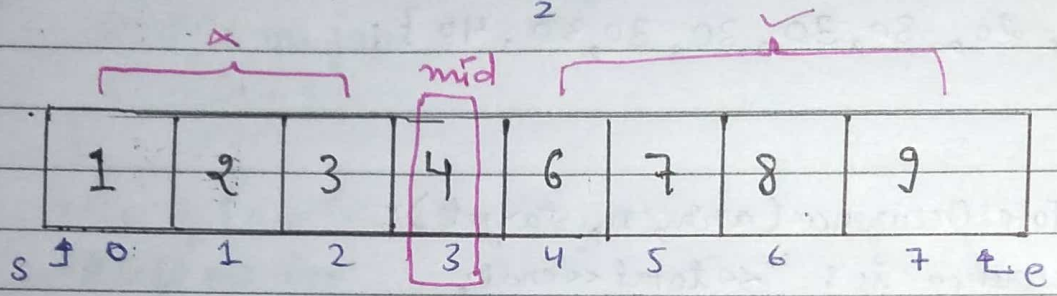


★ Jabtak missing element nhi ayga tabtak +1 pattern chalega, Or jese hi pattern change hoga toh jaha se pattern change hoga uske just pehle wala element hi missing element hoga.

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



s=0, e=08, mid = $\frac{0+8}{2}$ =



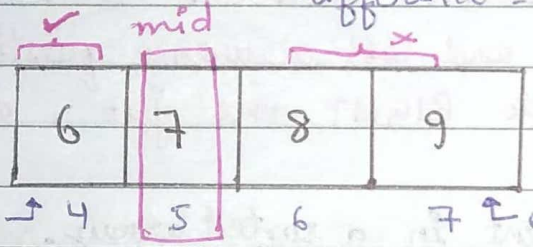
s=0, e=7, mid = $\frac{0+7}{2}$ = 3 difference = arr[mid] - mid

= 4 - 3

= 1

difference == 1 → Right me jao

s = mid + 1



s=4, e=7, mid = $\frac{4+7}{2}$ = $\frac{11}{2}$ = 5 difference = arr[mid] - mid

= 7 - 5

= 2

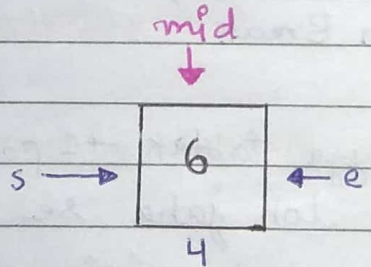
difference == 1 → False

↳ answer store Karo

ans = 5

↳ Left me jao

e = mid - 1



s=4, e=4, mid = $\frac{4+4}{2}$ = 4 difference = arr[mid] - mid

= 6 - 4

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

difference = 2

difference == 1 → No → ANS store , ans = 4

↳ LEFT me jao, e = mid-1

s=4, e=3 ⟶ s>e ⟶ Rukjao

Final Ans = ans+1

Code:-

```
int findMissingElement( 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 kro
       ans = mid;
       // Left me jao
       e = mid - 1;
    }
    mid = s+(e-s)/2;
 }
 if(ans +1 == 0){
    return n+1; // for the corner case when missing number is the last no
 }
return ans + 1;
}
```

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

Q Peak Index in a Mountain Array. 852 (Leetcode)

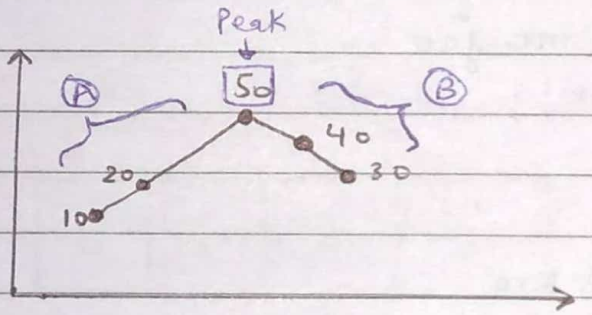
An array 'arr' is a mountain if the following properties hold:

- '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] > ... > arr[arr.length - 1].

You must solve it in O(log(arr.length)) time complexity.

if array → array[] = {10, 20, 50, 40, 30}

o/p → ans = 50



(A)

arr[i] < arr[i+1]

(B)

arr[i] > arr[i+1]

Peak Point

arr[i-1] < arr[i] > arr[i+1]

∴ If arr[i] < arr[i+1] → Exist in (A)

if Not then it must be Exist in (B) Line or a Peak element

Observation

(A) Line

arr[i] < arr[i+1]

(B) Line

arr[i] > arr[i+1]

Peak

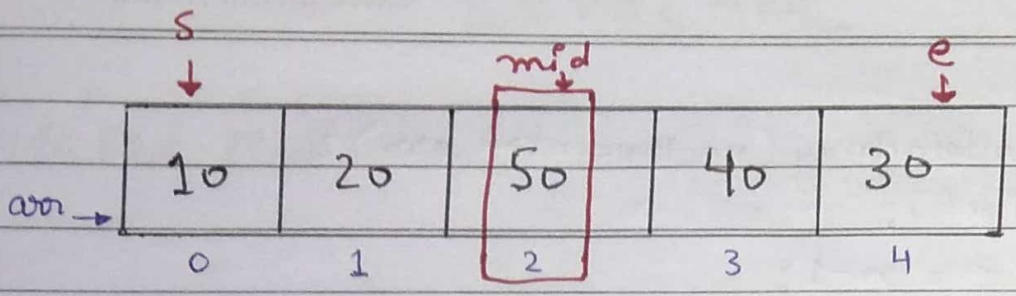
arr[i] > arr[i+1]

arr[i] > arr[i-1]

we can club

Spiral

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



s=0, e=4, mid = $\frac{0+4}{2}$ = 2, arr[mid] < arr[mid+1]

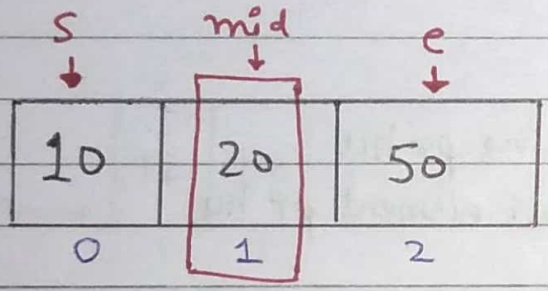
50 < 40 → False

↳ Either (B) or Peak

↳ Left me jao

e = mid - 1 ✗

(If we use this, we lost our peak)



e = mid ✓

(Benefit :- Left bhi chale gaye or peak bhi lost nhi hui)

s=0, e=2, mid = $\frac{0+2}{2}$ = 1, arr[mid] < arr[mid+1]

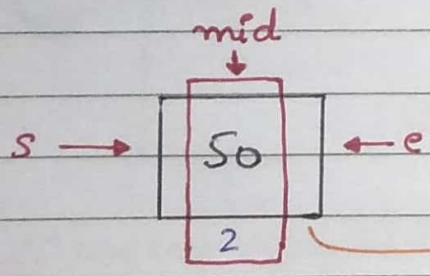
20 < 50 → True → (A) Line Pr hai

↳ Right me jao

s = mid + 1

s = 1 + 1

s = 2



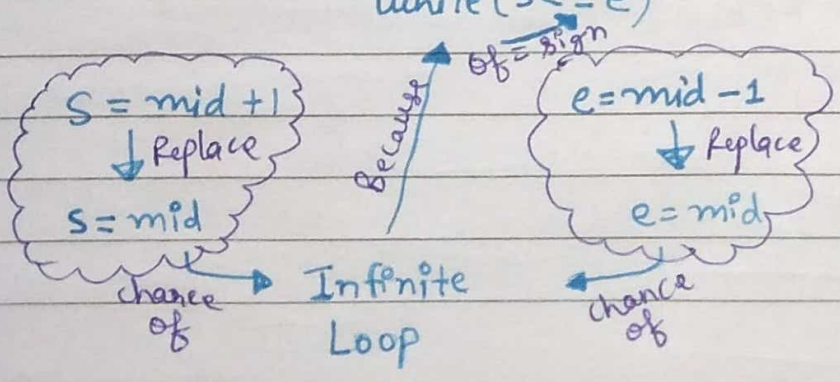
Single Element

↳ its the Peak

return s

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

KEEP IN MIND:-



∴ Remove '=' sign while changing these forward | Backword Conditions

Spiral

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

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) {  // Infinite Loop se bachne ke liye s<e
        if(arr[mid] < arr[mid+1]) {
            // A wali line me hu
            // Peak right me exist krti hai
            s = mid+1;
        }
        else {
            // Yaa toh main B line pr hu
            // Yaa toh main peak element pr hu
            e = mid;
        }
        // yaha galti nhi krni so update the mid
        mid = s+(e-s)/2;
    }
    return s;
}
```

SEARCHING && SORTING

LEVEL-2

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

Q Find Pivot element. [It's a sub part of search in a rotated & sorted array (33 Leetcode)]

Sorted

2	4	6	8	10	12	14	16

Pivot [max. no] in this case

Rotated & Sorted

12	14	16	2	4	6	8	10

Increasing ORDER — INCREASING ORDER

∴ Some elements are rotated here

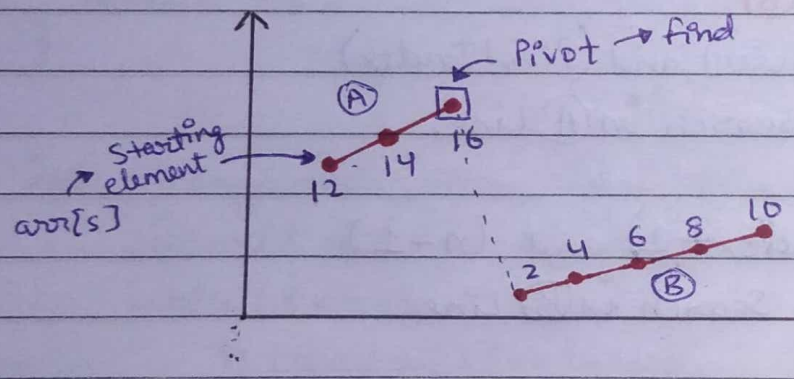
∵ Definition of pivit element is different in different resources.

Ex:-

Increasing, Decrease, Increasing

arr →

12	14	16	2	4	6	8	10
0	1	2	3	4	5	6	7



Approaches

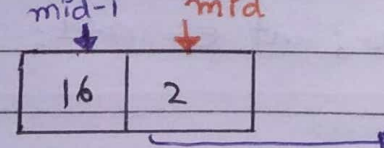
#1 → Linear Search → O(n)

#2 → Sort (Asc/Desc) → O(n log n)

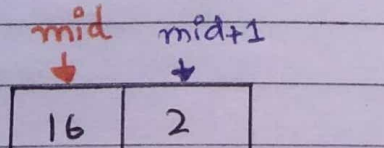
#3 → BS → O(log n)

∵ We can apply Binary search in (A) and (B) but we have to handle 16 and 2 seperately.

** [mid-1] >= 0 love Bhaiya cond^n lagana bhul gye the so, remember that [mid+1] < n



(I) arr[mid] < arr[mid-1] → ans → (mid-1)



(II) arr[mid] > arr[mid+1] → ans = (mid)

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

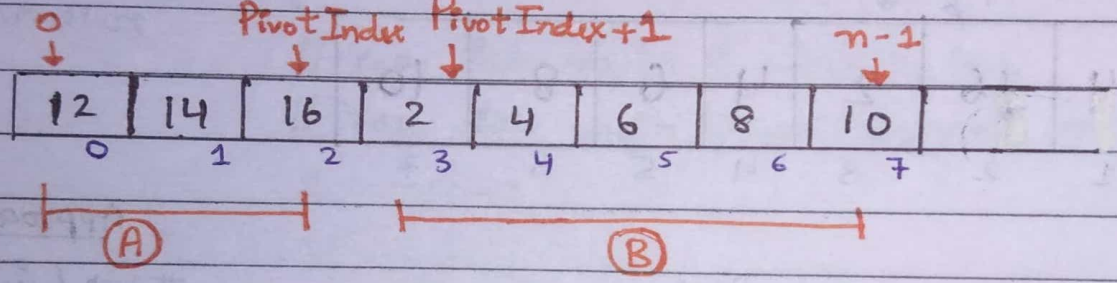
∴ Logic for (A) and (B) Line elements

(III) if arr[s] > arr[mid] → (B) Line
↳ left me jao

(IV) else → (A) Line
↳ right me jao

∴ Corner case s==e i.e when array have single element.
↳ return e bcz its the pivot element here.

↳ where to apply Binary search? in (A) Line or in (B) Line.



∴ If target is between (start index) and (PivotIndex) then apply Binary search in (A) Line.

If target is between (PivotIndex+1) and (n-1) then apply Binary search in (B) Line

Code:-

```
int findPivotIndex (vector<int> & arr) {
    int n = arr.size(); int s=0; int e = n-1;
    int mid = s+(e-s)/2;
    while ( s<=e) {
    // Corner Case
    if ( s ==e ) {
            // single element
            return e;
    }
```

Spir

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

```
    else if (mid-1 >=0 && arr[mid] < arr[mid-1]){
            return mid-1;
    }
    elseif (mid+1 <n && arr[mid] > arr[mid+1]) {
            return mid;
    }
    else if (arr[s] > arr[mid]){  // On (B) Line
            e = mid - 1;          // Left me jao
    }
    else {                        // On (A) Line
        s = mid +1;               // Right me jao
    }
    // yaha galti ni Karni so, update mid
    mid = s + (e- s)/2;
}
  return -1;
}

int binarySearch (vector<int>& arr, int s, int e , int target ){
    int mid = s+(e-s)/2;  int ans = -1;
    while (s<=e){
        if(arr[mid]==target)
            return mid;
        else if( arr[mid] > target)
            e= mid - 1;
        else
            s= mid+1;
        mid= s+(e-s)/2;
    }
    return -1;
}
```

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

```
int search(vector<int>& nums, int target) {
    int n = nums.size();
    int pivotIndex = findPivotIndex(nums);
    int ans = -1;

    // search in (A)
    if (target >= nums[0] && target <= nums[pivotIndex]) {
        ans = binarySearch(nums, 0, pivotIndex, target);
    }
    else {
        ans = binarysearch(nums, pivotIndex+1, n-1, target);
    }
    return ans;
}
```

3) Sqrt(x), (69 leetcode)

i/p → Number = x

o/p → $\sqrt{x}$ → ans

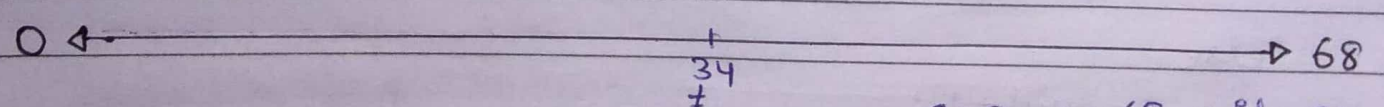
Ex:- i/p → 25 , 36 [Find it in O(log n) T.C]

o/p → $\sqrt{25} = 5$, $\sqrt{36} = 6$

Search space minimisation :-

Possibility of answers within a range.

Ex:- x = 68 , possible answers of $\sqrt{68}$ are its search space.



s = 0, e = 68, mid = 34

Predicate func^n

34 x 34 > 68 → left me jao

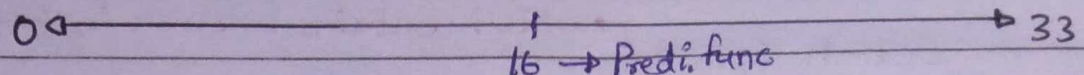
e = mid - 1

e = 34 - 1 = 33

Step I :- Search space choose

Step II :- Predicate function :- Check any condition and return T or F.

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

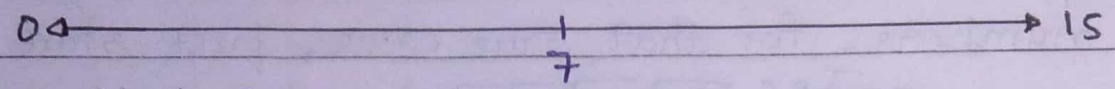


S=0, e=33, mid = 16

16 × 16 → 256 == 68 → F

256 > 68 → Left me jao → e = mid - 1

e = 16 - 1 = 15



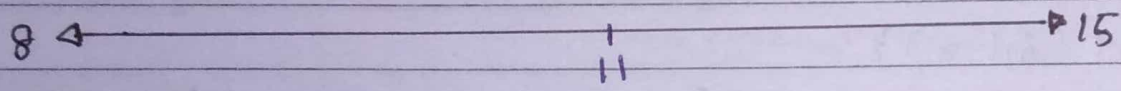
S=0, e=15, mid = $\frac{0+15}{2}$ = 7

7 × 7 = 49 == 68 → F

49 < 68 → Right me jao

Store ans ↙ ans = 7

s = mid + 1 = 7 + 1 = 8

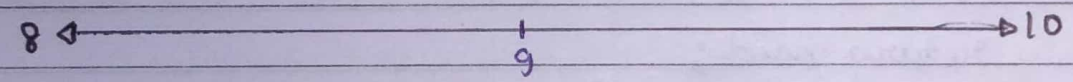


S=8, e = 15, mid = $\frac{8+15}{2} = \frac{23}{2}$ = 11

11 × 11 == 68 → F

121 > 68 → Left me jao → e = mid - 1

e = 11 - 1 = 10

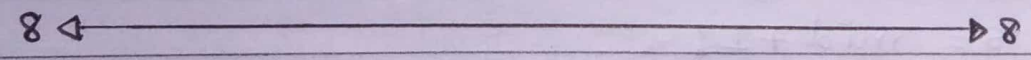


S=8, e=10, mid = 9

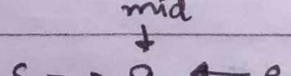
9 × 9 == 68 → F

81 > 68 → Left me jao → e = mid - 1

e = 9 - 1 = 8



S=8, e=8, mid=8



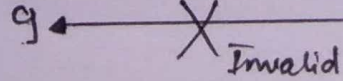
8 × 8 == 68 → F

8 × 8 > 68 → F

64 < 68 → store the ans in the case of '<' less than

ans = 8

Right me jao, s = mid = 1 ⇒ 8 + 1 = 9

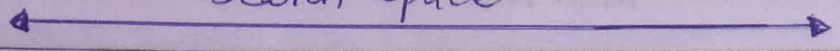


→ Rukjao

Spiral

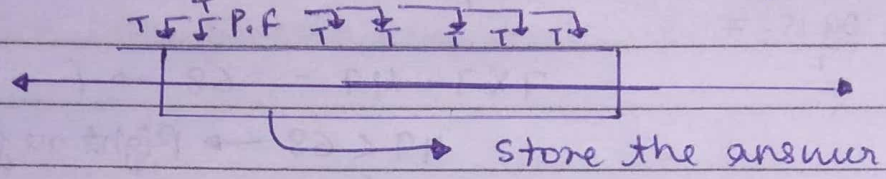
Date ..../..../.....

(A) Search Space → which contains answer in between



(B) Predicate function → To check any condⁿ and return True or false

(C) It may happen that, Predicate function return True for a range of numbers. for that true cases, first store the answer.



Code :-

```
int mySqrt (int x) {
    int s=0; int e=x ; int mid = s +(e - s)/2;
    int ans = -1;
    while (s <= e) {
    // Kya mid hi toh answer nahi
    if ( mid * mid == x)
            return mid;
    else if ( mid*mid < x )
            // store the ans
            //and move towards right
            ans = mid;
            s = mid +1;
    else
            //move towards Left
            e = mid - 1;
    //update mid
     mid = s + (e -s)/2;
    }
    return ans;
}
```

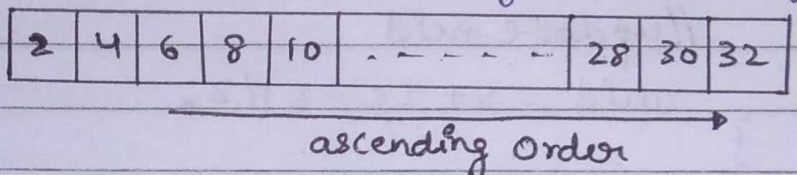
Date ..../..../.....

Binary Search in 2-D Array:-

	0	1	2	3
0	2	4	6	8
1	10	12	14	16
2	18	20	22	24
3	26	28	30	32

∴ (i) Every row is sorted in increasing order
(ii) Starting element of each row is greater than the last element of previous row.

→ It is stored in the memory as 1'D array.



↳ formula for 2D to 1D conversion — $(* i + j$
No. of Columns ← row index → col index

* ↳ formula for 1D to 2D conversion — $i = \frac{mid}{c}$, $j = mid \% c$
row Index ← No. of columns → column Index

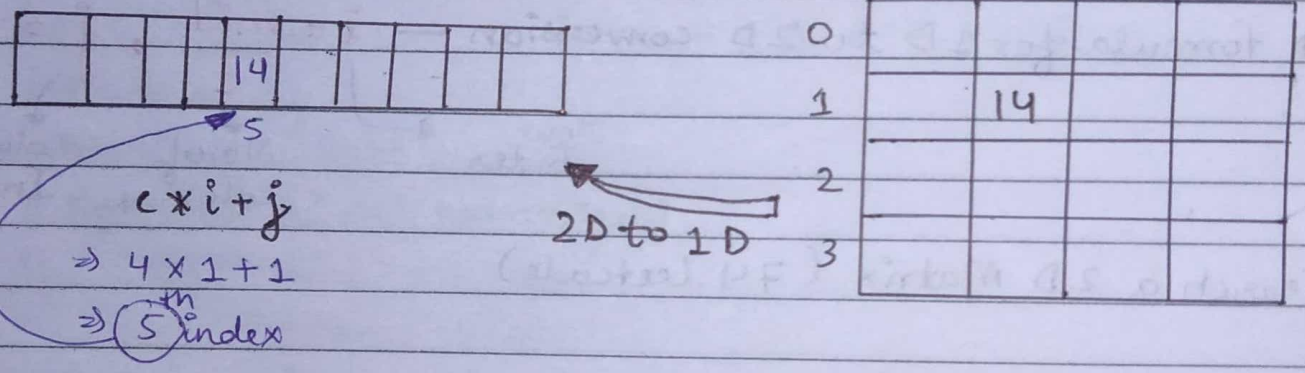
Q Search a 2D Matrix (74. Leetcode)

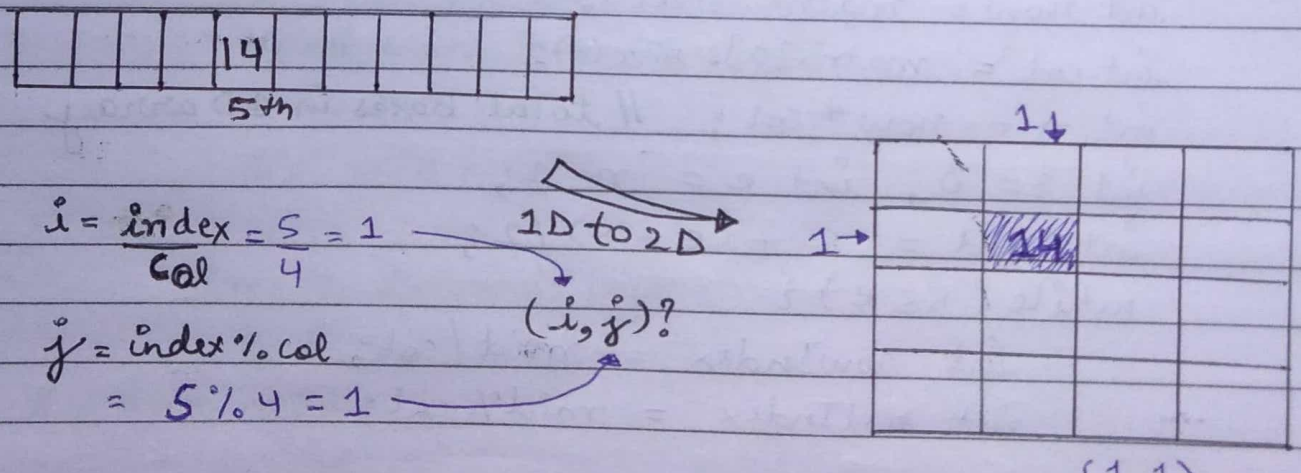
```
bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int row = matrix.size();
    int col = matrix[0].size();
    int n = row * col;  // total boxes in 1D array
    int s = 0, int e = n-1;
    int mid = s + (e-s)/2;
    while(s<=e){
        int rowIndex = mid/col;
        int colIndex = mid%col;

        int currentNumber = matrix[rowIndex][colIndex]
```

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

```
if (currentNumber == target)
        return true;
else if (currentNumber > target)
        // Left
        e = mid - 1;
else
        // right
        s = mid + 1;
//update mid
 mid = s + (e - s)/2;
}
return false;
}
```

Total column = 4





Spiral

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

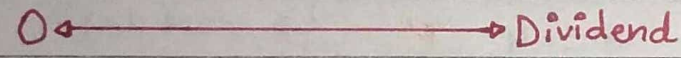
Divisor) Dividend (Quotient

Remainder

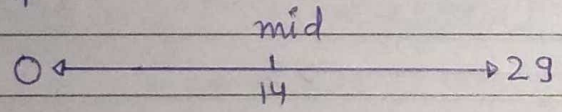
↳ Quotient * Divisor + Remainder = Dividend

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?

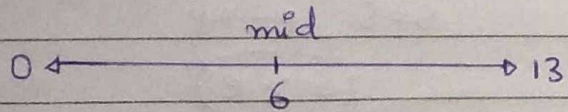
Quotient x divisor <= Dividend

14 x 7 = 98 ≠ 29

98 > 29 → false

↳ Left me jao → e = mid-1

e = 14-1 = 13



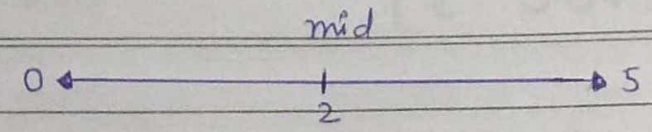
s=0, e=13, mid = $\frac{0+13}{2}$ = 6

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

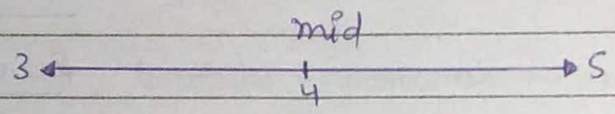
ans = 2

↳ 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

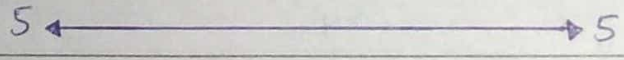
28 ≤ 29 → Valid Ans

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;
}
```

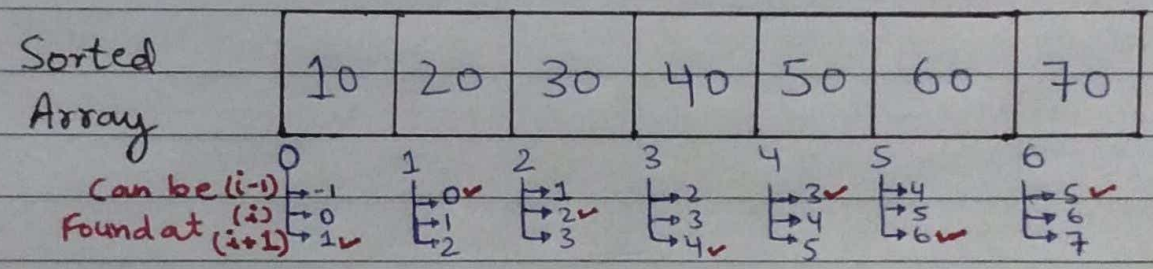
Date ..../..../.....

```
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

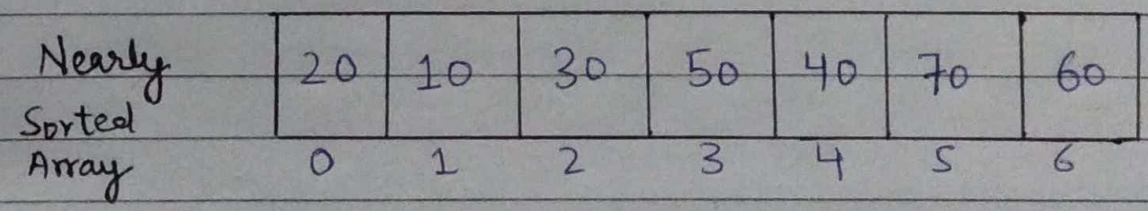
Q2. Binary Search on nearly sorted array.



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

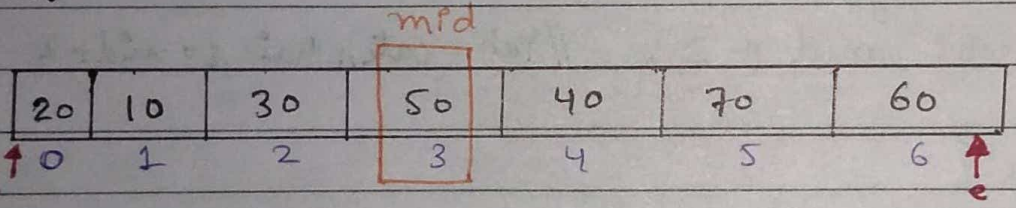
But here ans can be at
(mid-1)
(mid)
(mid+1)

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:-

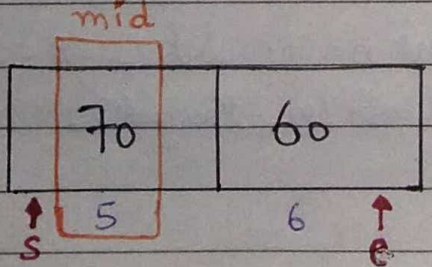


target = 70

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

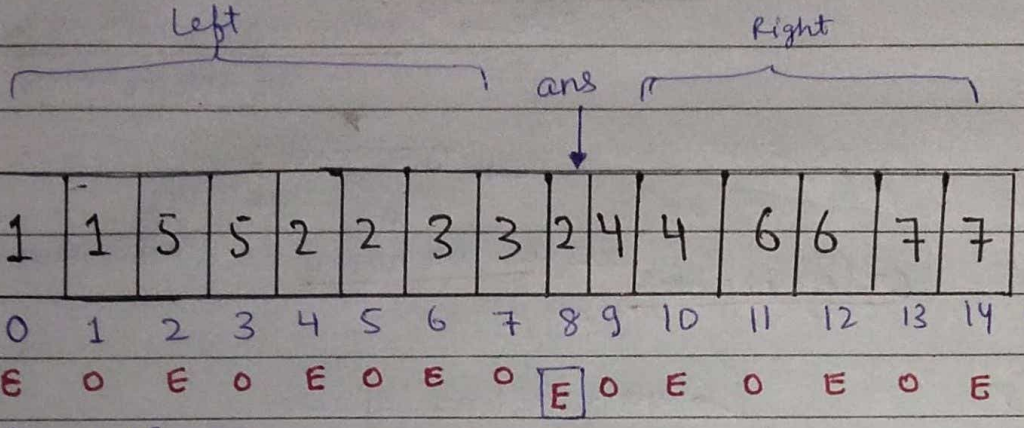
Q Find the odd occuring element.

↳ 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

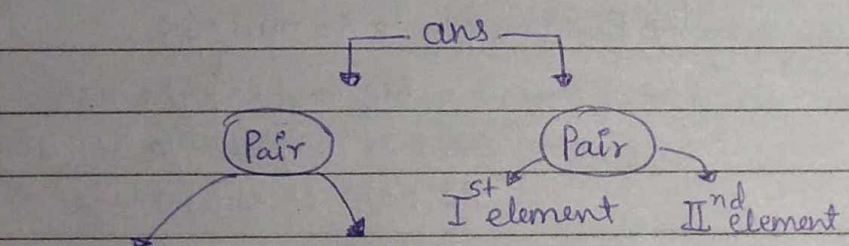
① Classical

② Search space

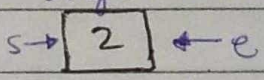
③ Predicate function

④ Index pe observation karke logic bnana

Observation :-



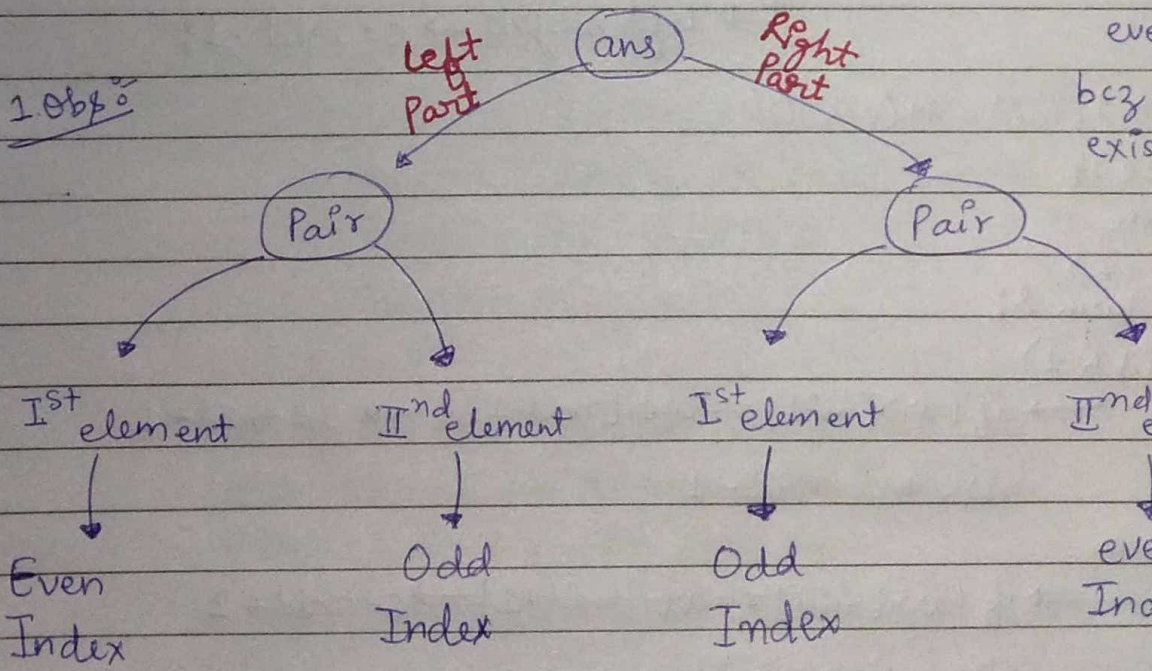
2 obs :- single element



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