left by mistake. (searching & sorting level - 2)

→ Binary search on 2D Array.

	0	1	2	3
0	2	4	6	8
1	10	12	14	16
2	18	20	22	24
3	28	34	40	50

→ Sotered 1D in actually.

2D → 1D

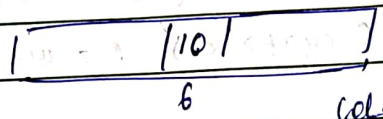
↳ c * i + j.

1D → 2D

$i = \frac{mid}{c}$ $j = mid \% c$

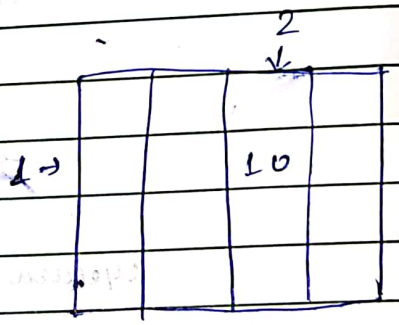
↳ Binary search same lagegi But 2D ↔ 1D ka conversion aana challenge.

1D → 2D



cols = 4

$i = \frac{index}{col} = \frac{6}{4} = 1$



~~j = index % col~~

$= 6 \% 4 = 2.$

$(i, j) = (1, 2) \rightarrow 10$

2D → 1D

c * i + j

1D → 2D

$i = \frac{index}{col}$, $j = index \% col.$

Code ~~bool searchMatrix(vector<int~~

```
bool searchMatrix(vector<vector<int>>& matrix,
                  int target) {
    int row = matrix.size();
    int col = matrix[0].size();
    int total = row * col;

    int s = 0;
    int e = total - 1;
    while(s <= e) {
        int mid = (s+e)/2;
        int rowIndex = mid / col;
        int colIndex = mid % col;
        int currElement = matrix[rowIndex][colIndex];

        if(currElement == target) return true;
        else if(currElement > target) e = mid - 1;
        else s = mid + 1;
    }
    return false;
}.
```

Time Complexity :- O(log n)
Space Complexity :- O(1).

Date
Page 213

Searching & Sorting - Level 3.

Q:- find quotient of a no.

I/P :- ① Divisor → a
② Dividend → b
} Divide Binary search.

$\frac{a}{b}$ → quotient

$\frac{24}{6} = 4$ $\qquad$ $\frac{28}{4} = 7$

divisor) dividend (Quotient.

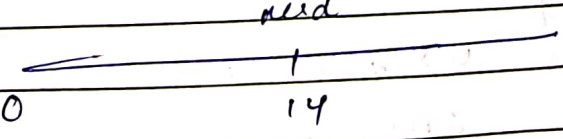
Remainder.

→ Quotient * divisor + Remainder = Dividend.

(Quotient * divisor <= Dividend.

Let's take eg :- Divisor = 7, Dividend = 29.

$\frac{29}{7}$.



s : 0
e : 29
→ mid = 14 → It can be possible ans so let's cross check.

Quotient x divisor <= Dividend.

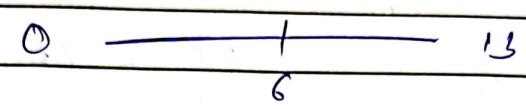
14 x 7 = 98

98 > 29

e = mid - 1 ;

e = 14 - 1

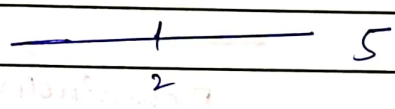
e = 13



mid = 6

6 x 7 = 42 > 29

e = 5

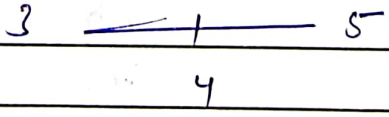


2 x 7 = 14 < 29

ans = 2

s = mid + 1 = 2 + 1 = 3

Move kr lenge.



mid = 4

4 x 7 <= 28

ans = 4

s = 5

mid = 5

5 × 7 > 35

e = 6

Here

e > s → loop breaks.

Code

```
int getQuotient (int divisor, int divident) {
    int s = 0;
    int e = divident;
    int ans = -1;
    while (s <= e) {
        int mid = s + (e-s)/2;
        if (mid * divisor == divident) return mid;
        else if (mid * divisor > divident)
            e = mid-1;
        else {
            ans = mid;
            s = mid+1;
        }
    }
    return ans.
main () {
int divisor = -7, divident = 28;
int res = getQuotient (abs(divisor), abs(divident));
if ((divident<0 && divisor>0) || (divident>0 &&
                                  divisor<0))
    res = 0 - res;
cout << "The quotient is: " << res << endl;
```

Questions on Binary search.

→ classical
→ search space.
→ Predicate function.
→ Index → logic.

Q. Binary search on nearly sorted Array.

Sorted Array.

10	20	30	40	50	60	70
0	1	2	3	4	5	6

Nearly Sorted Array.

20	10	30	50	40	70	60
0	1	2	3	4	5	6

↳ value may at
↳ i
↳ i-1
↳ i+1.

Normal Sorted Array	Nearly Sorted Array.

→ if (arr[mid] == target) return mid;
→ if (target > arr[mid]) ↳ Right
else ↳ Left.
	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
	else ↳ Left.

Dry Run

0	1	2	3	4	5	6
20	10	30	50	40	70	60
↑ s			↓ mid.			↑ e

s = 0
e = 6

target = 70.

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

if (target > arr[mid])
right jana hai
[s = mid + 2] → catch.

4	5	6
40	70	60
s ↓		e ↓

s = 4
e = 6

→ Agar hum s = mid + 1 karte toh.

5	6
70	60
s ↓	e ↓
mid	

s = 5
e = 6

arr[mid] == target ✓

return mid

5 → Ans

another target = 20

s		mid		e		
20	10	30	50	40	70	60
0	1	2	3	4	5	6

s = 0
e = 6 → mid = 3

arr[mid-1] == target ✗
arr[mid] == target ✗
arr[mid+1] == target ✗

if (target > arr[mid])
    s = mid + 2;
else
    e = mid - 2 → catch

s		e
20	10	
0	1	

mid = 1/2 = 0 → arr[0] == target ✓

found.
return mid

0 → Any

Code

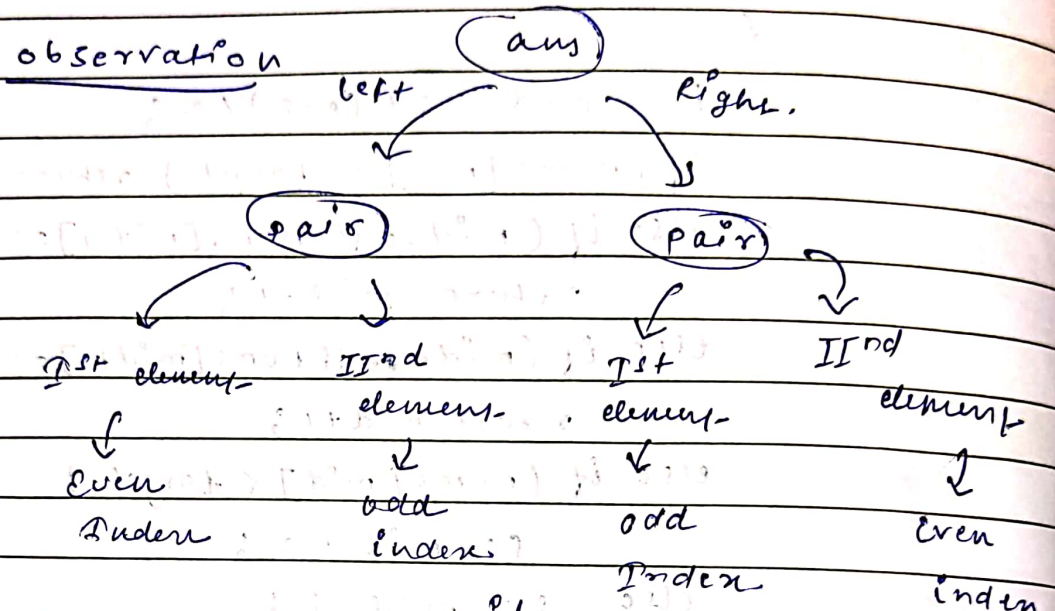
```
SearchNearlySortedArray (vector<int> &nums,
{                                    int &target)
    int n = nums.size() - 1;
    int s = 0;
    int e = n;
    while (s <= e)
    {
        int mid = s + (e-s) / 2;
        if (nums[mid] == target) return mid;
        else if (mid > 0 && nums[mid-1] == target)
            return mid-1;
        else if (mid < n && nums[mid+1] == target)
            return mid+1;
        else if (nums[mid] < target)
            s = mid + 2;
        else  e = mid - 2;
    }
    return -1;
}
```

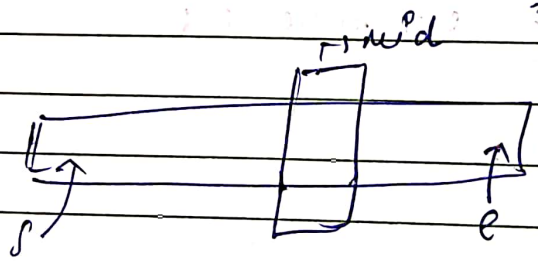
Time complexity → O(log n)
Space Complexity → O(1)

Q. find the odd occurring elements.

1	1	5	5	2	2	3	3	2	4	4
0	1	2	3	4	5	6	7	8	9	10

observation






```
if (mid % 2 == 0)
    if (arr[mid] == arr[mid+1])
        s = mid + 2
    else  e = m
else
    if (arr[mid] == arr[mid-1])
        s = mid + 1
    else  e = mid - 1
```

Catch

Code:

```
OddOccurringElement (vector<int> nums)
{
    int n = nums.size();
    int s = 0
    int e = nums.size() - 1;
    while ( s<=e ){
        if (s==e) return s;
        int mid = s+(e-s)/2;
        if (mid &1)
        {
            if (mid>0 && nums[mid] == nums[mid-1])
                s = mid+1;
            else e = mid-1;
        }
        else
        {
            if (mid+1<n && nums[mid]==nums[mid+
                s=mid+2;
            else e = mid;
        }
    }
}
```

Time complexity :- O(logn)
Space complexity :- O(1)