Binary search

01 Recursive Binary search.

Eg.

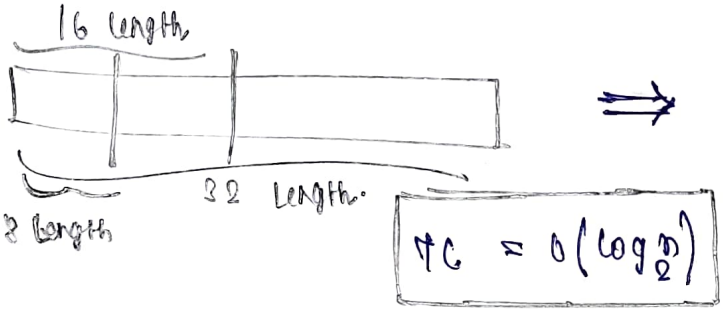
0	1	2	3	4	5	6	7
3	4	6	7	9	12	16	17

n = 8

target = 18.

```
R(arr, low, high) {
    if (low > high)        ] Base
    return -1;             ] case
    mid = (low + high) / 2;   ⟶ mid way
    if (a[mid] == target)
    return mid;
    else if (target > a[mid])
    return R(arr, mid+1, high, target);
    else
     return R(arr, low, mid-1, target);
```

Binary search corresp.

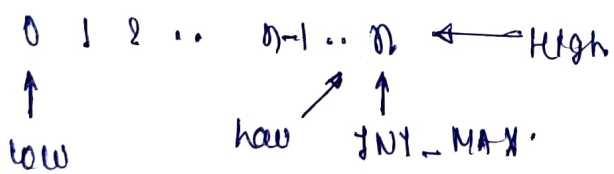


$\Rightarrow$ $32 = 2^5$

$64 = 2^6$

$$TC = O(\log_2 n)$$

Now **overflow cases.**



$$mid = (low + high)/2;$$

low, high → INT_MAX

So,

$$mid = low + \frac{(high - low)}{2}$$

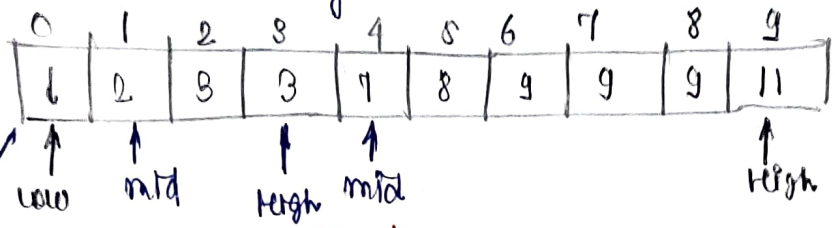
02 lower bound → smallest index such that arr[ind] >= n

arr[] = [3 5 8 15 19] n = 5

x = 8	x = 9	x = 19	x = 16	x = 20
lb = 2	lb = 3	lb = 4	lb = 4	lb = -1

- If no. is repeating so we return lower index.

Eg. 1



ans = ~~10 4 3~~ 0 If no element possible

Eg - 1

mid = 0+9/2 = 4

a[mid] = 7 > 4 yes possible.

mid = 0+3/2 = 1

a[1] = 2 > 1 yes possible.

mid = 0+0/2 = 0

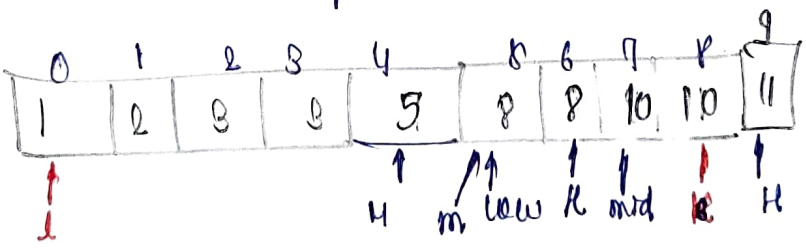
a[0] = 1 = 1 yes possible.

low, low = 0 high = -ve

so, loop end so ans = 0 Ans.

x = 9

Eg - 2



ans = ~~10~~ 7

mid = 0+9/2 = 4 a[4] = 5 < 9 No.

mid = 5+9/2 = 7 a[7] = 10 > 9 yes.

mid = 5+6/2 = 5 a[5] = 8 < 9 No.

mid = 6+6/2 = 6 a[6] = 8 < 9 No.

low, search space ended so, ans = 7 Ans

→ Upper bound → smallest index s.t arr[mid] > x

	0	1	2	3	4	5	6	7	8	9
arr[] =	2	3	6	7	8	8	11	11	11	12

x = 6	x = 11	x = 12	x = 13	x = 0
UB = 3	UB = 9	UB = 10	UB = 10	UB = 0

→ Search insert position.

arr[] = [1 2 4 7]

same as lower bound.

→ Floor and ceil in sorted array

largest no in array <= x

smallest no in array >= x

- arr[] = [10 20 30 40 50]

  x = 25

  floor = 20

  ceil = 30.

- arr[] = [10, 20 25 30 40]

  x = 25

  floor = 25

  ceil = 25.

```
Floor (arr, n) {
    ans = -1
    low = 0
    high = n-1
    while (low <= high) {
        mid = (low + high)/2;
        if (a[mid] <= mid) {
            ans = arr[mid];
            low = mid + 1;
        }
        else {
            high = mid - 1;
        }
    }
    return ans;
```

33 Find 1st and last occurrences of x

Eg [2 4 6 8 8 8 11 13]
0 1 2 3 4 5 6 7

x = 8	x = 11
First = 3	First = 6
last = 5	last = 6

use lower bound → a[ind] >= x
and upper bound → a[ind] > x

so LB(8) = so index = 3
UB(8) = so index = 6 - 1 = 5

---

but it will not work for all.

so, LB(10) = index = 6 ✗

so, LB(14) = index = 8 ✗

so, if (LB(x) == n || arr[LB] != x)
so, no. not present.

TC = O(log2 n)
SC = O(1).

Hind way

[2 8 8 8 8 8 11 13]

Here we use simple BS for 1st and last occurrences and return it.

34 search in rotated sorted array I

arr[] = [7 8 9 1 2 3 4 5 6] target = 1
0 1 2 3 4 5 6 7 8

use of Binary search (because of search in sorted).

Eg 1 2 3 4 5
low mid high

so, (1 <= 3)

- just checking on one half it makes decision.

so, 0 + 8/2 = 4 a[4] = 2 ≠ target

so a[low] = a[0] = 7
a[mid] = a[4] = 2
not sorted.

→ so, we eliminate right because it is sorted.

so, low = 0
high = mid - 1 = 3.

so, 1 > 1 left me hoga

↳ yaha pe dono half ko check kro sirf ek ko check karne se nhi hoga.

again do these things.

15 Search in rotated sorted array II

arr[] = [7 8 1 2 3 3 3 4 5 6] [Identify sorted half]

target = 3

Eg. [6 7 1 2 3 4 4 5] — l, m, h — Yes sorted

Eg [4 5 1 2 3 3 4 4] — l, m, h — m sorted, not sorted

Eg. [3 1 2 3 3 3 3] — low, mid, high

Edge case

Here clashes will be done.

try to trim down this condition.

arr[low] = arr[mid] = arr[high]

↳ problem.

trim search space

TC = O(log n/2)

↳ [3 3 1 3 3 3 3] — low, l, mid, high, h

So, [8 1 2 3 3 3 3] — low, mid, high.

again we shrink.

Here, yaha pe search space half ho jayega.

16 Minimum in rotated sorted array (unique).

Eg. [4 5 6 7 0 1 2] — 0 1 2 3 4 5 6

return 0. — l, mid, high

sorted | not sorted

→ eliminate left / right.

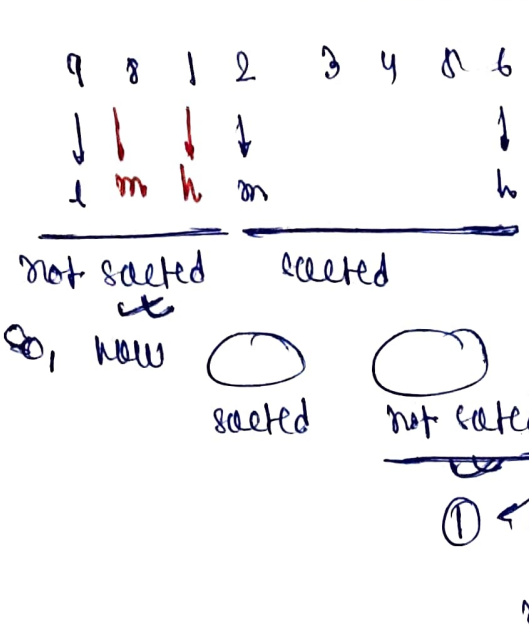
→ Identify sorted half.

Eg [4 5 1 2 3] — l, m, h

Here right half contains ans.

Left answer is here.

eg.



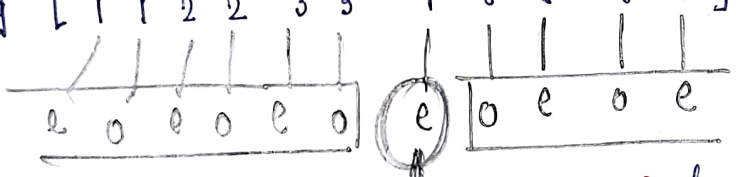
07 How many times has an array been rotated

Same slight variation in Find Min code and manipulated with index

Keep track of index only (Min ans).

08 single element in sorted array

eg [1 1 2 2 3 3 4 5 5 6 6]. → Elimination will be done.



$TC = O(\log \frac{n}{2})$

- (0,0) → element in on right half
- (0,1) → element in on left half.

09 Find peak element.

arr[] = [1 2 8 4 5 6 7 8 5 1]

ans = (8)

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

eg. [1 2 1 3 5 6 4]

ans = (2)

eg. -∞ [1 2 8 4 5] -∞

ans = (5)

eg. -∞ [5 4 3 2 1] -∞

ans = 5

Assume array has 1 peak.

	0	1	2	3	4	5	6	7	8	9
arr =	1	2	3	4	5	6	7	8	5	1

ans = 8. Iteration 1 — l = 0, h = 9 | m = 4, arr[4] = 5 | check kr lo mil gya peak.

so left me chale jao

It-2 — l = 5, h = 9 | m = 7, arr[7] = 8 | yes peak. done.

Eg.

	0	1	2	3	4	5	6	7	8	9
	1	10	13	7	6	5	4	2	1	0
	↑ l									↑ high

It-1 — l = 0, h = 9 | m = 4, arr[4] = 6 | not peak, 7 > 6 > 5 decreasing | so, peak on the left.

It-11 — l = 0, h = 3 | m = 1, arr[1] = 10 | not peak. 1 < 10 < 13 increasing | so, peak on right

It-111 — l = 2, h = 3 | m = 2, arr[2] = 13 | yes peak done.

What if first and last will be peak?

Eg. [1 2 3 4 5] Here peak = 5.

Eg [5 4 3 2 1] Here peak = 5.

How to deal with that.

arr[mid] > arr[mid+1]

and arr[mid] > arr[mid-1]

- Simple manually write code for 1st and last elements.

mid = 0

mid = n-1

- It happens directly returns don't check for others.

→ Binary search on answers.

10 Find sqrt(x)

Eg $n = 28$

$ans = 5$

$ans = \cancel{8}\ 5$

So, **ans = sub**

So, low = 1, high = 28 | mid = 14 | $14 \times 14 > 28$ ✗

So, l = 1, h = 13 | mid = 7 | $7 \times 7 > 28$ ✗

So, l = 1, h = 6 | mid = 3 | $3 \times 3 < 28$ ✓

So, l = 4, h = 6 | mid = 5 | $5 \times 5 = 25 < 28$ ✓

So, l = 6, h = 6 | mid = 6 | $6 \times 6 > 28$ ✗

11 Find Nth root of an integer

N = 3, M = 27 → So, l = 1, h = 27 | m = 14 | $14 \times 14 \times 14 > 27$ ✗

So, l = 1, h = 13 | m = 7 | $7 \times 7 \times 7 > 27$ ✗

So, l = 1, h = 6 | m = 3 | $3 \times 3 \times 3 = 27$ ✓

ans = 3

N = 9, m = 69

So, l = 1, h = 69 | m = 35 | $35 \times 35 \times 35 \times 35 \times 35 > 69$ ✗

So, l = 1, h = 34 | m = 17 | ✗

So, l = 1, h = 16 | m = 8 | ✗

So, l = 1, h = 7 | m = 4 | ✗

ans = 2

So, l = 1, h = 3 | m = 2 | $2 \times 2 \times 2 \times 2 = 16 < 69$ ✓

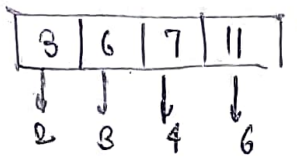
So, l = 3, h = 3 | m = 3 | $3 \times 3 \times 3 \times 3 > 69$ ✗

Koko eating Bananas. → return the minimum integer k such that koko can eat all bananas within h hours.

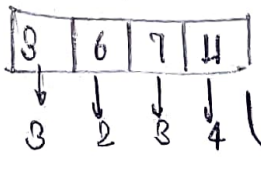
Here, k = banana / hr

Eg. piles[] = [3 6 7 11]
h = 8

If take 2 banana / hr.

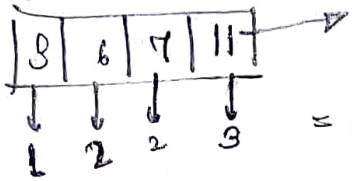
So, 

→ 2
= 18 hr

So 

= 12 hr.

Eg. 2 banana / hr.

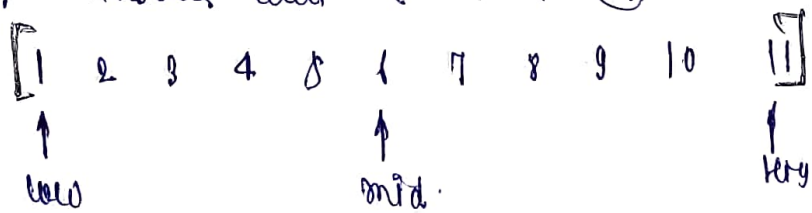
So, 

→ 4 banana/hr
= 8 hrs

So (ans = 4) Ans

Here we use binary search on answers.

So, answers will be 1 to (11) → max of array.



So, ans = 6.

So, l = 1 h = 11 m = 6	do, 1+1+2+2 = 6 < 8	Yes possible
So, l = 1 h = 5 m = 3	do 1+2+3+4 = 10 > 8	No
So, l = 4 h = 5 m = 4	do 1+2+2+3 = 8 = 8	Yes done.

TC = O(N) * log max el 2

14 minimum no. of days to make m bouquets.

bloom day = [7 7 7 7 13 11 12 7] m = 2 (no. of bouquets) k = 3 (adjacent flowers required)

18th day: 7 7 7 | 11 12 7

17th day: ✗ not possible.

So, ans = ~~18~~ 12.

12th day: [✓ ✓ ✓ ✓ ✗ ✓ ✓ ✓] yes.

Yes done.

11th day: [✓ ✓ ✓ ✓ ✗ ✗ ✗ ✓] No ✗

Eg: [1 10 3 10 2] m = 3 k = 2

10th day: [✓ ✓ ✓ ✓ ✓] No 3 bouquets were possible

So, ans = -1.

Note:

7	7	7	7	13	11	12	7

So, BS on answers: [7 8 9 10 11 12 13] (l = 7, mid = 10, h = 13)

l = 7, h = 13, m = 10 | No not possible

l = 11, h = 13, m = 12 | yes

l = 11, h = 11, m = 11 | No. ~~yes~~

ans = 12

TC = O(N × $\log_2$(max − min + 1))

15 Find the smallest divisor given a threshold

arr[] = [1 2 5 9] Threshold = 6.

Q-1. 1/1 2/1 5/1 9/1

1 + 1 + 2 + 3 = 7 ≯ 6

(ans = 5)

Q-2 1/5 2/5 5/5 9/5

1 1 1 2 = 5 < 6 (yes)

16 Capacity to ship packages within D days

arr[] = [1 2 3 4 5 6 7 8 9 10] days = 5.

Ship will run once per day.

Capacity 10

day 1 →	1 2 3 4 = 10.	→ 1
day 2 →	5 = 5	→ 2
day 3 →	6 = 6	→ 3
day 4 →	7	
day 5 →	8	total 7 days
day 6 →	9	
day 7 →	10	

(ans = 15)

Now Capacity 15.

So, day-1 = 1 2 3 4 5
day-2 = 6 7
day-3 = 8
day-4 = 9
day-5 = 10

Yes 5 days completed

So, min Capacity = * max-element of array

max capacity of ship to be hold be sum of all array.

So, Range = [10, 55]

So, l = 10, h = 58, m = 32 | Yes

So, l = 10, h = 81, m = 90 | Yes

one = ~~32~~ 90 ~~14~~ 18

So, l = 10, h = 19, m = 14 | ✗

So, l = 18, h = 19, m = 18 | yes

So, l = 18, h = 16, m = 18 | yes

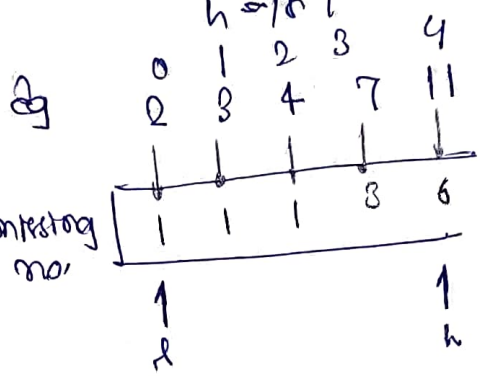
So, l = 18, m = 18, h = 18 | Yes

17 Find the missing number

arr[] = [2 3 4 7 11] k = 8.

So, 1 2 3 4 5 6 7 8 9 10 11

return 9



arr[] = [4 7 9]

k = 8.

So, high is pointing to -ve index

So, not good way.

always not good.

arr[high] + more.

arr[high] + (k - missing)

arr[high] + k - (arr[high] - high - 1)

~~(arr[high] + k - arr[high] + high + 1)~~

[k + high + 1] Ans

So, l = 0, h = 4, m = 2 | m[2] = 1 ✗

So, l = 3, h = 4, m = 3 | m[3] = 3 ✗

So, l = 4, h = 4, m = 4 | m[4] = 6. ✓

Now l = 4, h = 3 | find no.

arr[high] = 7

and missing no = 3.

diff = 2.

So, 7 → 8 → 9.

one

high + how many more you need.

18 Aggressive cows (min dist b/w cows) is maximum

arr[] = [0 3 4 7 10 9]

cows = 4.

↳ 0 3 4 7 9 10.

C1 C2 C3 C4 = 1

C1 ~~+~~ C2 ~~+~~ C3 ~~+~~ C4 = 1

C1 ~~+~~ C2 ~~+~~ C3 ~~+~~ C4 = 3

↳ same as LC 1552

Magnetic force b/w two balls.

So ans = 3.

- We iterative put cows and how much cows we put is selected by Binary search.

19 Allocate Books.

arr[] = [25 46 28 49 24]

Students = 4.

25 | 46 | 28 | 49 + 24. → max = (77)

25 | 46 | 28 + 49 | 24 → max = (77)

25 | 46 + 28 | 49 | 24 → max = (74)

We should start from (49) → max element. (Low)

high = sum of array.

[49 . . . . . 172]

So, l = 49, h = 172, m = 110

1 → 25 46 28 ✗ only 2 students.

2 → 49 24

So l = 49, h = 109, m = 78

1 → 25 46 ✗ only 3 students

2 → 28 49

3 → 24

So, l = 49
h = 77
m = 68

1 → 10
2 →
3 →
4 →
5 →

students

So l = 70
h = 77
m = 73

4 students

So, one = 73

So, l = 62
h = 77
m = 69

5 students

So, l = 70
h = 72
m = 71

4 students

So, l = 70
h = 70
m = 70

5 students not possible

20 Painters Partition

and, split array largest sum,

now, l = 71
h = 70

arr[] = [10 20 30 40]

k = 2

So, [10] [20 30 40] So, 90 P2

[10 20] [30 40] So, 70 P2

[10 20 30] [40] So, 60 P1.

min (all max)

ans.

min 60 print

Now split array largest sum

split array in k subarrays such each one has one element and max subarray sum is minimum.

same

as Painter's Partition.

21 Minimise maximum distance b/w gas stations. 08.

Eg. arr[] = [1 2 3 4 5]
k = 4
→ ans = [1, 2, 3, 4, 5, 6, 7, 8, 9].

- ans within 10^6 of actual answer would be accepted.

Output 6 decimal places.

Eg. arr[] = [1 7]
k = 2.

Now, [1 2 4 7] = 3 (gaps 1, 2, 3)

[1 3 5 7] = (2) is better. (gaps 2, 2, 2)

Eg. [1 13 17 23] k = 5 (gaps 12, 4, 6)

0 1 2
| 1 | 1 | 1 |
2
3

3 1 1

ans[] = [1 (4) (7) (10) 13 (15) 17 (20) 23]

22 Median of two sorted arrays

arr1[] = [1 3 4 7 10 12]
arr2[] = [2 3 6 15]

→ arr[] = [1 2 3 3 4 6 7 10 12 15]
(n = 10)
4 + 6/2 = (5)

Eg. arr1[] = [2 3 4]
arr2[] = [1 3]
[1 2 3 3 4]
↑
ans

Now, How we use binary search?

arr[] = [1 2 3 3 4 | 6 7 10 12 15] in hypothetical scenarios.

pick 1 element: 1 | 2 3 6 15 | 3 4 7 10 12] not valid

pick 2 element: 1 3 | 2 3 6 | 4 7 10 12 15] not valid.

Pick 8 element | 1 3 4 11 | 11 9 10 12 | Yes valid → 8 element from arr 1 and 2 from another

| 2 3 3 4 | 6 7 10 12 18

Symmetry

Pick 4 element | 1 3 4 7 | 10 12 | 4 < 6, 8 < 7 yes

2 | 8 6 18.

So, 0 1 2 (3) 4 5 6 — we check only cross guys.

we Yes done.

Just for validation.

Now greater in (11, 12) + ~~greater~~ small (9, 2) / 2

So, (4 + 6)/2 = 10/2 = (5) Ans

Now [0 1 2 3 4 5 6] do BS on that array.

we do BS on shorter array too.

23 kth element of two sorted arrays.

arr1[] = [2 3 6 7 9]

arr2[] = [1 4 8 10]

k = 4

So,

4	n
2 3	6 7 9
1 4	8 10
(k)	(n-k)

24 Row with maximum 1's.

0	0	0	1	1	1	= 3
1	0	0	0	0	0	= 0
2	0	1	1	1	1	= (4)
3	0	0	0	0	0	= 0
4	0	1	1	1	1	= (4)

return (2) min in row

- Brute way we done in O(n×m)
- now use binary search.
- we found only first occurrence of one's
- or LB of 1
- and (UB of 0)

Now, mat[0] = [0 0 1 1 1]

25 Search in 2D matrix

mat[][] = $\begin{bmatrix} 3 & 4 & 7 & 9 \\ 12 & 13 & 16 & 18 \\ 20 & 21 & 28 & 29 \end{bmatrix}$ target = 28

yes

Simply flatten this in a single array.

So, [3 4 7 9 12 13 16 18 20 21 28 29]

How? Formula 1D → 2D.

mid = 10 $\xrightarrow{10/4}$ 10 y. 4 (mat[2,2])

So, (mid / mat[0].size() , mid % mat[0].size()).

↳ yehi coordinate hai yeh formula hai.

26 search in 2D matrix - II

$\begin{bmatrix} 1 & 4 & 7 & 11 & 18 \\ 2 & 5 & 8 & 12 & 19 \\ 3 & 6 & 9 & 16 & 22 \\ 10 & 13 & 14 & 17 & 24 \\ 18 & 21 & 23 & 26 & 30 \end{bmatrix}$

Here every row and column are sorted.

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

These all sorted,

target = 14.

row = 0 ~~1~~ ~~2~~ 3

col = ~~4~~ ~~3~~ 2

row (3, 2) = 14

yes.

time = O(n+m)

27 Find peak element ↔ 2 adjacent cells are not same.

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

- There are multiple peak element but you have to return only single one {row, column}.

1st low = 0 high = 5 | max = 6 but not peak.

mid = 2

now delete right half

2nd low = 0 high = 1 | max = 4 but it's peak — so return this.

mid = 0

TC = O(log$_2^m$ × n).

SC = O(1).

28 Median OR row wise sorted matrix.

$$\begin{bmatrix} 1 & 5 & 7 & 9 & 11 \\ 2 & 3 & 4 & 8 & 10 \\ 9 & 10 & 12 & 14 & 16 \end{bmatrix}$$

[1 2 2 4 5 5 7 9 9 10 10 11 12 14 16]

↓ 2 3 4 6 6 7 7

→ A bit tough Question.

BINARY SEARCH QUESTIONS

01.) IMPLEMENT LOWER BOUND

CODE

```
int lowerBound(vector<int> arr, int n, int x) {
    int low=0;
    int high=n-1;
    int ans=n;

    while(low<=high){
        int mid=(low+high)/2;
        if(arr[mid]>=x){
            ans=mid;
            high=mid-1;
        }
        else{
            low=mid+1;
        }
    }
    return ans;
}
```

02.) IMPLEMENT UPPER BOUND

CODE

```
int upperBound(vector<int> &arr, int x, int n){
    int low=0;
    int high=n-1;
    int ans=n;

    while(low<=high){
        int mid=(low+high)/2;
        if(arr[mid]>x){
            ans=mid;
            high=mid-1;
        }
        else
        low=mid+1;
    }
    return ans;
}
```

BINARY SEARCH QUESTIONS

03.) SEARCH INSERT POSITION

CODE

Same as lower bound question that is done above

```
int searchInsert(vector<int>& arr, int x)
{
    int n=arr.size();
    int low=0;
    int high=n-1;
    int ans=n;
    while(low<=high){
        int mid=(low+high)/2;
        if(arr[mid]>=x){
            ans=mid;
            high=mid-1;
        }
        else{
            low=mid+1;
        }
    }
    return ans;
}
```

BINARY SEARCH QUESTIONS

04.) FLOOR AND CEIL

CODE

```
int floor(vector<int> &a, int n, int x){
    int ans=-1;
    int low=0;
    int high=n-1;
    while(low<=high){
        int mid=(low+high)/2;
        if(a[mid]<=x){
            ans=a[mid];
            low=mid+1;
        }
        else{
            high=mid-1;
        }
    }
    return ans;
}

int ceil(vector<int> a, int n, int x) {
    int low=0;
    int high=n-1;
    int ans=-1;

    while(low<=high){
        int mid=(low+high)/2;
        if(a[mid]>=x){
            ans=a[mid];
            high=mid-1;
        }
        else{
            low=mid+1;
        }
    }
    return ans;
}

pair<int, int> getFloorAndCeil(vector<int> &a, int n, int x) {
    int ans1=floor(a, n, x);
    int ans2=ceil(a, n, x);
    pair<int,int> p=make_pair(ans1, ans2);
    return p;
}
```

BINARY SEARCH QUESTIONS

05.) FIND FIRST AND LAST OCCURENCES IN AN ARRAY

CODE

```
class Solution {
public:
    int lowerBound(vector<int> arr, int n, int x) {
        int low=0;
        int high=n-1;
        int ans=-1;

        while(low<=high){
            int mid=(low+high)/2;
            if(arr[mid]==x){
                ans=mid;
                high=mid-1;
            }
            else if(arr[mid]<x){
                low=mid+1;
            }
            else{
                high=mid-1;
            }
        }
        return ans;
    }

    int upperBound(vector<int> &arr, int n, int x){
        int low=0;
        int high=n-1;
        int ans=n;

        while(low<=high){
            int mid=(low+high)/2;
            if(arr[mid]>x){
                ans=mid;
                high=mid-1;
            }
            else
            low=mid+1;
        }
        return ans;
    }

    Vector<int> searchRange(vector<int>& nums, int target) {
        int n=nums.size();
        vector<int> ans;
        int lb=lowerBound(nums, n, target);
        ans.push_back(lb);
```

```
        if(lb==-1)
        ans.push_back(-1);
        else
        ans.push_back(upperBound(nums, n, target)-1);
        return ans;
    }
};
```

06.) SEARCH IN ROTATED SORTED ARRAY

CODE

```
class Solution {
public:
    int search(vector<int>& nums, int target) {
        int low=0;
        int high=nums.size()-1;

        while(low<=high){
            int mid=(low+high)/2;
            if(nums[mid]==target)
            return mid;

            if(nums[low]<=nums[mid]){
                if(nums[low]<=target && target<=nums[mid]){
                    high=mid-1;
                }
                else{
                    low=mid+1;
                }
            }
            else{
                if(nums[mid]<=target && target<=nums[high]){
                    low=mid+1;
                }
                else{
                    high=mid-1;
                }
            }
        }
        return -1;
    }
};
```

BINARY SEARCH QUESTIONS

07.) SEARCH IN ROTATED SORTED ARRAY II

CODE

```cpp
class Solution {
public:
    bool search(vector<int>& nums, int target) {
        int low=0;
        int high=nums.size()-1;

        while(low<=high){
            int mid=(low+high)/2;
            if(nums[mid]==target)
            return true;

            // EXTRA CASE ONLY
            if(nums[low]==nums[mid] && nums[mid]==nums[high]){
                low++;
                high--;
                continue;
            }
            if(nums[low]<=nums[mid]){
                if(nums[low]<=target && target<=nums[mid]){
                    high=mid-1;
                }
                else{
                    low=mid+1;
                }
            }
            else{
                if(nums[mid]<=target && target<=nums[high]){
                    low=mid+1;
                }
                else{
                    high=mid-1;
                }
            }
        }
        return false;
    }
};
```

BINARY SEARCH QUESTIONS

8.) FIND MINIMUM IN ROTATED SORTED ARRAY

CODE

```
class Solution {
public:
    int findMin(vector<int>& nums) {
        int low=0;
        int high=nums.size()-1;
        int ans=INT_MAX;

        while(low<=high){
            int mid=(low+high)/2;
            if(nums[low]<=nums[mid]){
                ans=min(ans, nums[low]);
                low=mid+1;
            }
            else{
                high=mid-1;
                ans=min(ans, nums[mid]);
            }
        }
        return ans;
    }
};
```

BINARY SEARCH QUESTIONS

9.) FIND MINIMUM IN ROTATED SORTED ARRAY II

CODE

```
class Solution {
public:
    int findMin(vector<int>& nums) {
        int low=0;
        int high=nums.size()-1;
        int ans=INT_MAX;

        while(low<=high){
            int mid=(low+high)/2;
            // EXTRA CASE ONLY
            if(nums[low]==nums[mid] && nums[mid]==nums[high]){
                ans=min(ans, nums[low]);
                low++;
                high--;
                continue;
            }
            if(nums[low]<=nums[mid]){
                ans=min(ans, nums[low]);
                low=mid+1;
            }
            else{
                high=mid-1;
                ans=min(ans, nums[mid]);
            }
        }
        return ans;
    }
};
```

BINARY SEARCH QUESTIONS

10.) HOW MANY TIMES ARRAY IS ROTATED

CODE

Slight variation in find minimum in sorted array code

```
#include <bits/stdc++.h>
int findMin(vector<int>& nums) {
    int low=0;
    int high=nums.size()-1;
    int ans=INT_MAX;
    int index=-1;
    while(low<=high){
        int mid=(low+high)/2;
        if(nums[low]<=nums[high]){
            if(nums[low]<ans){
                index=low;
                ans=nums[low];
            }
            break;
        }
        if(nums[low]<=nums[mid]){
            if(nums[low]<ans){
                index=low;
                ans=nums[low];
            }
            low=mid+1;
        }
        else{
            if(nums[mid]<ans){
                index=mid;
                ans=nums[mid];
            }
            high=mid-1;
        }
    }
    return index;
}

int findKRotation(vector<int> &arr){
    int ans=findMin(arr);
    return ans;
}
```

BINARY SEARCH QUESTIONS

11.) SINGLE ELEMENT IN SORTED ARRAY

CODE

```
class Solution {
public:
    int singleNonDuplicate(vector<int>& nums) {
        if(nums.size()==1)
        return nums[0];

        if(nums[0]!=nums[1])
        return nums[0];

        if(nums[nums.size()-1]!=nums[nums.size()-2])
        return nums[nums.size()-1];

        int low=1;
        int high=nums.size()-1;

        while(low<=high){
            int mid=(low+high)/2;
            if(nums[mid]!=nums[mid+1] && nums[mid]!=nums[mid-1])
            return nums[mid];
            if((mid%2==1) && nums[mid-1]==nums[mid]
                || (mid%2==0) && nums[mid]==nums[mid+1]){
                low=mid+1;
            }
            else{
                high=mid-1;
            }
        }
        return -1;
    }
};
```

BINARY SEARCH QUESTIONS

12.) FIND PEAK ELEMENT

CODE

```
class Solution {
public:
    int findPeakElement(vector<int>& nums) {
        int n=nums.size();
        if(n==1)
        return 0;
        if(nums[0]>nums[1])
        return 0;
        if(nums[n-1]>nums[n-2])
        return n-1;

        int low=1;
        int high=n-2;

        while(low<=high){
            int mid=(low+high)/2;
            if(nums[mid]>nums[mid-1] && nums[mid]>nums[mid+1]){
                return mid;
            }
            else if(nums[mid]>nums[mid-1]){
                low=mid+1;
            }
            else{
                high=mid-1;
            }
        }
        return -1;
    }
};
```

BINARY SEARCH QUESTIONS

13.) PEAK INDEX IN A MOUNTAIN ARRAY

CODE

```
class Solution {
public:
    int peakIndexInMountainArray(vector<int>& nums) {
        int n=nums.size();
        if(n==1)
        return 0;
        if(nums[0]>nums[1])
        return 0;
        if(nums[n-1]>nums[n-2])
        return n-1;

        int low=1;
        int high=n-2;

        while(low<=high){
            int mid=(low+high)/2;
            if(nums[mid]>nums[mid-1] && nums[mid]>nums[mid+1]){
                return mid;
            }
            else if(nums[mid]>nums[mid-1]){
                low=mid+1;
            }
            else{
                high=mid-1;
            }
        }
        return -1;
    }
};
```

BINARY SEARCH QUESTIONS

14.) FIND SQRT

CODE

```
class Solution {
public:
    int mySqrt(int x) {
        if(x==0)
        return 0;

        int low=1;
        int high=x;
        int ans=INT_MIN;

        while(low<=high){
            int mid=low+((high-low)/2);
            if(mid>x/mid){
                high=mid-1;
            }
            else if(mid<=x/mid){
                ans=max(ans, mid);
                low=mid+1;
            }
        }
        return ans;
    }
};
```

BINARY SEARCH QUESTIONS

15.) FIND THE Nth ROOT OF AN INTEGER

CODE

```cpp
#include <bits/stdc++.h>
using namespace std;

int func(int mid, int n, int m) {
  long long ans = 1;
  for (int i = 1; i <= n; i++) {
      ans = ans * mid;
      if (ans > m) return 2;
  }
  if (ans == m) return 1;
  return 0;
}

int NthRoot(int n, int m) {
  int low = 1, high = m;
  while (low <= high) {
      int mid = (low + high) / 2;
      int midN = func(mid, n, m);
      if (midN == 1) {
          return mid;
      } else if (midN == 0) {
          low = mid + 1;
      } else {
          high = mid - 1;
      }
  }
  return -1;
}
```

BINARY SEARCH QUESTIONS

16.) KOKO EATING BANANAS

CODE

```
class Solution {
public:
    long long func(vector<int> &p, int mid){
        long long sum=0;
        for(int i=0;i<p.size();i++){
            sum=sum+ceil((double)p[i]/(double)mid);
        }
        return sum;
    }

    int minEatingSpeed(vector<int>& p, int h) {
        long long int low=1;
        long long high=*max_element(p.begin(), p.end());
        int ans=-1;

        while(low<=high){
            long long int mid=low+((high-low)/2);
            long long int tot_hr=func(p, mid);
            if(tot_hr<=h){
                ans=mid;
                high=mid-1;
            }
            else{
                low=mid+1;
            }
        }
        return ans;
    }
};
```

BINARY SEARCH QUESTIONS

17.) MINIMUM NUMBER OF DAYS TO MAKE M BOUQUETS

CODE

```
class Solution {

public:
    bool func(vector<int> &b, int day, int m, int k){
        int count=0;
        int num=0;
        for(int i=0;i<b.size();i++){
            if(b[i]<=day){
                count++;
            }
            else{
                num=num+(count/k);
                count=0;
            }
        }
        num=num+(count/k);
        return num>=m;
    }

    int minDays(vector<int>& b, int m, int k) {
        if(b.size()/k<m)
        return -1;

        int low=*min_element(b.begin(), b.end());
        int high=*max_element(b.begin(), b.end());
        int ans=INT_MAX;

        while(low<=high){
            int mid=(low+high)/2;
            if(func(b, mid, m, k)==true){
                ans=mid;
                high=mid-1;
            }
            else{
                low=mid+1;
            }
        }
        return ans;
    }
};
```

BINARY SEARCH QUESTIONS

18.) MAXIMUM CANDIES ALLOCATED TO K CHILDREN

CODE

```
class Solution {
public:
    bool func(vector<int> &c, int mid, long long k){
        long long int count=0;
        for(int i=0;i<c.size();i++){
            count=count+(c[i]/mid);
        }
        return count>=k;
    }

    int maximumCandies(vector<int>& c, long long k) {
        int low=1;
        int high=*max_element(c.begin(), c.end());

        while(low<=high){
            int mid=low+((high-low)/2);
            if(func(c, mid, k)){
                low=mid+1;
            }
            else{
                high=mid-1;
            }
        }
        return high;
    }
};
```

BINARY SEARCH QUESTIONS

19.) FIND THE SMALLEST DIVISOR GIVEN A THRESHOLD VALUES

CODE

```
class Solution {
public:
    int func(vector<int> &nums, int mid){
        int count=0;
        for(int i=0;i<nums.size();i++){
            count=count+ceil((double)nums[i]/(double)mid);
        }
        return count;
    }

    int smallestDivisor(vector<int>& nums, int th) {
        int low=1;
        int high=*max_element(nums.begin(), nums.end());

        while(low<=high){
            int mid=(low+high)/2;
            if(func(nums, mid)<=th){
                high=mid-1;
            }
            else{
                low=mid+1;
            }
        }
        return low;
    }
};
```

BINARY SEARCH QUESTIONS

20.) CAPACITY TO SHIP PACKAGES WITHIN D DAYS

CODE

```
class Solution {
public:
    int func(vector<int> &w, int cap){
        int day=1;
        int load=0;
        for(int i=0;i<w.size();i++){
            if(w[i]+load>cap){
                day=day+1;
                load=w[i];
            }
            else{
                load=load+w[i];
            }
        }
        return day;
    }

    int shipWithinDays(vector<int>& w, int days) {
        int low=*max_element(w.begin(), w.end());
        int high=accumulate(w.begin(), w.end(), 0);
        while(low<=high){
            int mid=(low+high)/2;
            int day=func(w, mid);
            if(day<=days)
            high=mid-1;
            else
            low=mid+1;
        }
        return low;
    }
};
```

BINARY SEARCH QUESTIONS

21.) Kth MISSING POSITIVE NUMBERS

CODE

```
class Solution {
public:
    int findKthPositive(vector<int>& arr, int k) {
        int low=0;
        int high=arr.size()-1;

        while(low<=high){
            int mid=(low+high)/2;
            int miss=arr[mid]-(mid+1);
            if(miss<k)
            low=mid+1;
            else
            high=mid-1;
        }
        return k+high+1;
    }
};
```

BINARY SEARCH QUESTIONS

22.) AGGRESSIVE COWS

CODE

```
bool ispossible(vector<int> &stalls, int k, int mid) {
    int cowcount = 1;
    int lastpos = stalls[0];

    for (int i = 0; i < stalls.size(); i++) {
        if (stalls[i] - lastpos >= mid) {
            cowcount++;
            if (cowcount == k) {
                return true;
            }
            lastpos = stalls[i];
        }
    }
    return false;
}

int aggressiveCows(vector<int> &stalls, int k) {
    sort(stalls.begin(), stalls.end());
    int s = 0;
    int e = *max_element(stalls.begin(), stalls.end());
    int ans = -1;

    while (s <= e) {
        int mid = s + (e - s) / 2;
        if (ispossible(stalls, k, mid)) {
            ans = mid;
            s = mid + 1;
        } else {
            e = mid - 1;
        }
    }
    return ans;
}
```

BINARY SEARCH QUESTIONS

23.) MAGNETIC FORCE BETWEEN TWO BALLS

CODE

Same as aggressive cow questions

```cpp
class Solution {
public:
    bool ispossible(vector<int> &stalls, int k, int mid) {
        int cowcount = 1;
        int lastpos = stalls[0];

        for (int i = 0; i < stalls.size(); i++) {
            if (stalls[i] - lastpos >= mid) {
                cowcount++;
                if (cowcount == k) {
                    return true;
                }
                lastpos = stalls[i];
            }
        }
        return false;
    }

    int maxDistance(vector<int>& stalls, int k) {
        sort(stalls.begin(), stalls.end());
        int s = 0;
        int e = *max_element(stalls.begin(), stalls.end());
        int ans = -1;

        while (s <= e) {
            int mid = s + (e - s) / 2;
            if (ispossible(stalls, k, mid)) {
                ans = mid;
                s = mid + 1;
            } else {
                e = mid - 1;
            }
        }
        return ans;
    }
};
```

BINARY SEARCH QUESTIONS

24.) ALLOCATE BOOKS

CODE

```
#include <bits/stdc++.h>
using namespace std;

int func(vector<int> &arr, int pages){
    int stud=1;
    long long pageStud=0;

    for(int i=0;i<arr.size();i++){
        if(pageStud+arr[i]<=pages){
            pageStud=pageStud+arr[i];
        }
        else{
            stud++;
            pageStud=arr[i];
        }
    }
    return stud;
}

int findPages(vector<int>& arr, int n, int m) {
    if(m>n)
    return -1;

    int low=*max_element(arr.begin(), arr.end());
    int high=accumulate(arr.begin(), arr.end(), 0);

    while(low<=high){
        int mid=(low+high)/2;
        int students=func(arr, mid);
        if(students>m)
        low=mid+1;
        else
        high=mid-1;
    }
    return low;
}
```

BINARY SEARCH QUESTIONS

25.) SPLIT ARRAY LARGEST SUM

CODE

Same as ALLOCATE BOOKS

```
class Solution {
public:
    int func(vector<int> &arr, int pages){
        int stud=1;
        long long pageStud=0;

        for(int i=0;i<arr.size();i++){
            if(pageStud+arr[i]<=pages){
                pageStud=pageStud+arr[i];
            }
            else{
                stud++;
                pageStud=arr[i];
            }
        }
        return stud;
    }

    int splitArray(vector<int>& arr, int m) {
        int n=arr.size();
        if(m>n)
        return -1;

        int low=*max_element(arr.begin(), arr.end());
        int high=accumulate(arr.begin(), arr.end(), 0);

        while(low<=high){
            int mid=(low+high)/2;
            int students=func(arr, mid);
            if(students>m)
            low=mid+1;
            else
            high=mid-1;
        }
        return low;
    }
};
```

BINARY SEARCH QUESTIONS

26.) PAINTER'S PARTITION PROBLEM

CODE

Same as ALLOCATED BOOKS

```cpp
#include <bits/stdc++.h>
using namespace std;

int func(vector<int> &arr, int pages){
    int stud=1;
    long long pageStud=0;

    for(int i=0;i<arr.size();i++){
        if(pageStud+arr[i]<=pages){
            pageStud=pageStud+arr[i];
        }
        else{
            stud++;
            pageStud=arr[i];
        }
    }
    return stud;
}

int findLargestMinDistance(vector<int> &arr, int m)
{
    int n=arr.size();
    if(m>n)
    return -1;

    int low=*max_element(arr.begin(), arr.end());
    int high=accumulate(arr.begin(), arr.end(), 0);

    while(low<=high){
        int mid=(low+high)/2;
        int students=func(arr, mid);
        if(students>m)
        low=mid+1;
        else
        high=mid-1;
    }
    return low;

}
```

BINARY SEARCH QUESTIONS

27.) FIND Kth SMALLEST PAIR DISTANCE

CODE

```
class Solution {
public:
    bool isPossible(vector<int>& nums, int k, int mid) {
        int count = 0;
        int left = 0;

        for (int right = 0; right < nums.size(); right++) {
            while (nums[right] - nums[left] > mid) {
                left++;
            }
            count += right - left;
        }
        return count >= k;
    }

    int smallestDistancePair(vector<int>& nums, int k) {
        sort(nums.begin(), nums.end());
        int low = 0, high = nums.back() - nums.front();

        while (low < high) {
            int mid = low + (high - low) / 2;
            if (isPossible(nums, k, mid)) {
                high = mid;
            } else {
                low = mid + 1;
            }
        }
        return low;
    }
};
```

BINARY SEARCH QUESTIONS

28.) MEDIAN OF TWO SORTED ARRAYS

CODE 01

```
class Solution {
public:
    double findMedianSortedArrays(vector<int>& a, vector<int>& b) {
        int n1=a.size();
        int n2=b.size();

        int i=0;
        int j=0;
        int n=(n1+n2);

        int ind2=n/2;
        int ind1=ind2-1;

        int count=0;
        int ind1el=-1;
        int ind2el=-1;

        while(i<n1 && j<n2){
            if(a[i]<b[j]){
                if(count==ind1)
                ind1el=a[i];
                if(count==ind2)
                ind2el=a[i];
                count++;
                i++;
            }
            else{
                if(count==ind1)
                ind1el=b[j];
                if(count==ind2)
                ind2el=b[j];
                count++;
                j++;
            }
        }
        while(i<n1){
            if(count==ind1)
            ind1el=a[i];
            if(count==ind2)
            ind2el=a[i];
            count++;
            i++;
        }
```

```
        while(j<n2){
            if(count==ind1)
            ind1el=b[j];
            if(count==ind2)
            ind2el=b[j];
            count++;
            j++;
        }

        if(n%2==1)
        return ind2el;
        return (double)((double)(ind1el+ind2el)/2.0);
    }
};
```

CODE 02

```
class Solution {
public:
    double findMedianSortedArrays(vector<int>& a, vector<int>& b) {
        int n1=a.size();
        int n2=b.size();

        if(n1>n2)
        return findMedianSortedArrays(b, a);

        int low=0;
        int high=n1;

        int left=(n1+n2+1)/2;
        int n=n1+n2;

        while(low<=high){
            int mid1=(low+high)>>1;
            int mid2=left-mid1;

            int l1=INT_MIN;
            int l2=INT_MIN;
            int r1=INT_MAX;
            int r2=INT_MAX;

            if(mid1<n1)
            r1=a[mid1];
            if(mid2<n2)
            r2=b[mid2];
            if(mid1-1>=0)
            l1=a[mid1-1];
```

```
            if(mid2-1>=0)
            l2=b[mid2-1];

            if(l1<=r2 && l2<=r1){
                if(n%2==1)
                return max(l1, l2);
                return ((double)(max(l1, l2)+min(r1, r2)))/2.0;
            }
            else if(l1>r2)
            high=mid1-1;
            else
            low=mid1+1;
        }
        return 0;
    }
};
```

BINARY SEARCH QUESTIONS

29.) Kth ELEMENT OF TWO SORTED ARRAYS

CODE

```
#include <bits/stdc++.h>
int kthElement(vector<int> &a, vector<int> &b, int n1, int n2, int k){
    if(n1>n2)
    return kthElement(b, a, n2, n1, k);

    int low=max(k-n2, 0);
    int high=min(k, n1);

    int left=k;
    int n=n1+n2;

    while(low<=high){
        int mid1=(low+high)>>1;
        int mid2=left-mid1;

        int l1=INT_MIN;
        int l2=INT_MIN;
        int r1=INT_MAX;
        int r2=INT_MAX;

        if(mid1<n1)
        r1=a[mid1];
        if(mid2<n2)
        r2=b[mid2];
        if(mid1-1>=0)
        l1=a[mid1-1];
        if(mid2-1>=0)
        l2=b[mid2-1];

        if(l1<=r2 && l2<=r1){
            return max(l1, l2);
        }
        else if(l1>r2)
        high=mid1-1;
        else
        low=mid1+1;
    }
    return 0;

}
```

BINARY SEARCH QUESTIONS

30.) ROW WITH MAXIMUM 1's

CODE

```
class Solution {
public:
    int lowerBound(vector<int> &arr, int n, int x) {
        int low=0;
        int high=n-1;
        int ans=n;
        sort(arr.begin(), arr.end());

        while(low<=high){
            int mid=(low+high)/2;
            if(arr[mid]>=x){
                ans=mid;
                high=mid-1;
            }
            else{
                low=mid+1;
            }
        }
        return ans;
    }

    vector<int> rowAndMaximumOnes(vector<vector<int>>& mat) {
        int count_max=0;
        int index=0;

        for(int i=0;i<mat.size();i++){
            int m=mat[i].size();
            int count=m-lowerBound(mat[i], m, 1);
            if(count>count_max){
                count_max=count;
                index=i;
            }
        }
        return {index, count_max};
    }
};
```

BINARY SEARCH QUESTIONS

31.) SEARCH IN 2D MATRIX I

CODE

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& mat, int target) {
        int n=mat.size();
        int m=mat[0].size();

        int low=0;
        int high=(n*m)-1;

        while(low<=high){
            int mid=(low+high)/2;
            int row=mid/m;
            int col=mid%m;

            if(mat[row][col]==target)
            return true;
            else if(mat[row][col]<target)
            low=mid+1;
            else
            high=mid-1;
        }
        return false;
    }
};
```

BINARY SEARCH QUESTIONS

32.) SEARCH IN 2D MATRIX II

CODE

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& mat, int target) {
        int n=mat.size();
        int m=mat[0].size();

        int row=0;
        int col=m-1;

        while(row<n && col>=0){
            if(mat[row][col]==target)
            return true;
            else if(mat[row][col]<target)
            row++;
            else
            col--;
        }
        return false;
    }
};
```

BINARY SEARCH QUESTIONS

33.) FIND PEAK ELEMENT II

CODE

```
class Solution {
public:
    vector<int> findPeakGrid(vector<vector<int>>& mat) {
        int m = mat.size();
        int n = mat[0].size();
        int low = 0;
        int high = n - 1;

        while (low <= high) {
            int mid = low + (high - low) / 2;
            int maxRow = 0;

            for (int i = 0; i < m; i++) {
                if (mat[i][mid] > mat[maxRow][mid]) {
                    maxRow = i;
                }
            }

            bool isLeftValid = false;
            bool isRightValid = false;

            if (mid - 1 >= 0 && mat[maxRow][mid] < mat[maxRow][mid - 1]) {
                isLeftValid = true;
            }

            if (mid + 1 < n && mat[maxRow][mid] < mat[maxRow][mid + 1]) {
                isRightValid = true;
            }

            if (!isLeftValid && !isRightValid) {
                return {maxRow, mid};
            }

            if (isLeftValid) {
                high = mid - 1;
            }
            else {
                low = mid + 1;
            }
        }
        return {-1, -1};
    }
};
```

BINARY SEARCH QUESTIONS

34.) MEDIAN OF A ROW WISE SORTED MATRIX

CODE

```
class Solution {
public:
    int countLessEqual(vector<vector<int>>& matrix, int mid) {
        int count = 0;
        int rows = matrix.size();
        int cols = matrix[0].size();

        for (int i = 0; i < rows; ++i) {
            count += upper_bound(matrix[i].begin(), matrix[i].end(), mid)
                                - matrix[i].begin();
        }
        return count;
    }

    int findMedian(vector<vector<int>>& matrix) {
        int rows = matrix.size();
        int cols = matrix[0].size();

        int low = INT_MAX, high = INT_MIN;
        for (int i = 0; i < rows; ++i) {
            low = min(low, matrix[i][0]);
            high = max(high, matrix[i][cols - 1]);
        }

        int desired = (rows * cols + 1) / 2;

        while (low < high) {
            int mid = low + (high - low) / 2;
            if (countLessEqual(matrix, mid) < desired) {
                low = mid + 1;
            } else {
                high = mid;
            }
        }

        return low;
    }
};
```

THANK YOU !