Binary search

01 Recursive Binary search.

Eg.

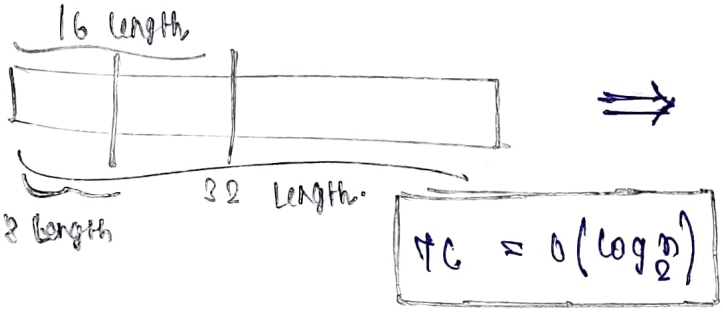
0	1	2	3	4	5	6	7
3	4	6	7	9	12	16	17

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.

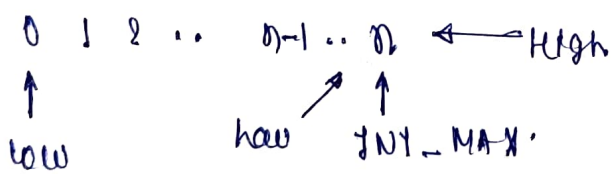


$\Rightarrow$ $32 = 2^5$

$64 = 2^6$

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

Now **overflow cases.**



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

low, high → INT_MAX

So,

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

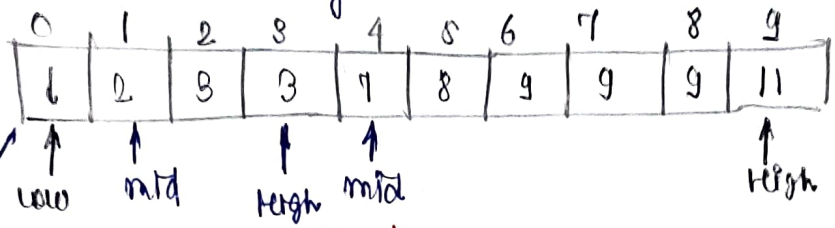
02 lower bound → smallest index such that arr[ind] >= n

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

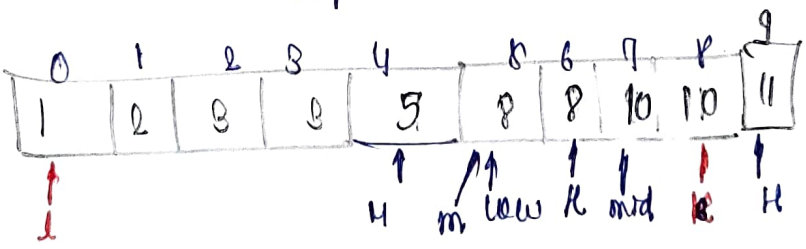
a[0] = 1 = 1 yes possible.

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.

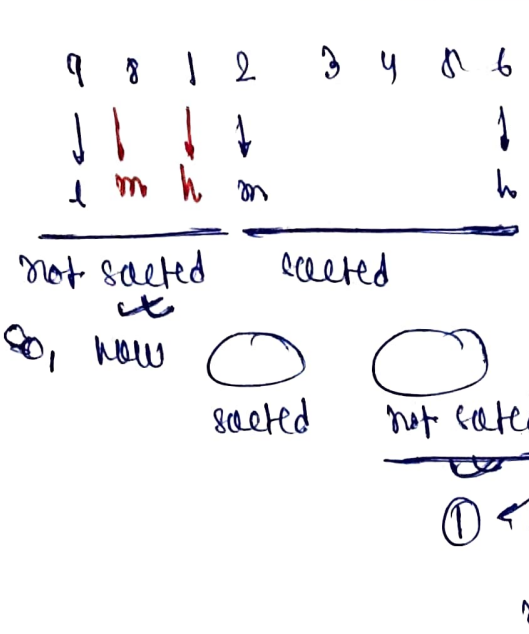
→ Identify sorted half.

Eg [4 5 1 2 3] — l, m, h

Here right half contains ans.

Left answer is here.

eg.



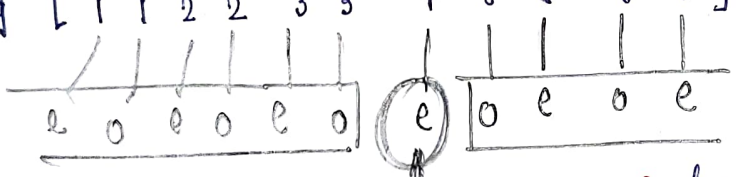
07 How many times has an array been rotated

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

S-I: low = 1, high = 28 | mid = 14
$14 \times 14 > 28$ ✗

S-II: l = 1, h = 13 | mid = 7
$7 \times 7 > 28$ ✗

S-III: l = 1, h = 6 | mid = 3
$3 \times 3 < 28$ ✓

ans = ~~3~~ 5

So, **ans = 5**

S-IV: l = 4, h = 6 | mid = 5
$5 \times 5 = 25 < 28$ ✓

S-V: l = 6, h = 6 | m = 6
$6 \times 6 > 28$ ✗

11. Find Nth root of an integer

N = 4, m = 69 | 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

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

So l = 1, m = 34 | ✗ m = $\cancel{8}$ 7

So, l = 1, h = 16 | m = 8
✗

So, l = 1, h = 7 | m = 4
✗

ans = 3

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

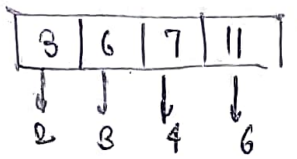
So l = 3, h = 3 | m = 3
$3 \times 3 \times 3 \times 3 > 69$ ✗

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

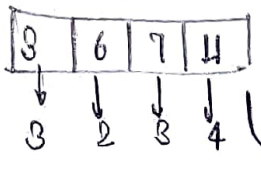
Here, k = banana / hr

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

If take 2 banana / hr.

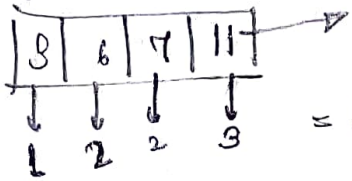
So, 

→ 2
= 18 hr

So 

= 12 hr.

Eg. 2 banana / hr.

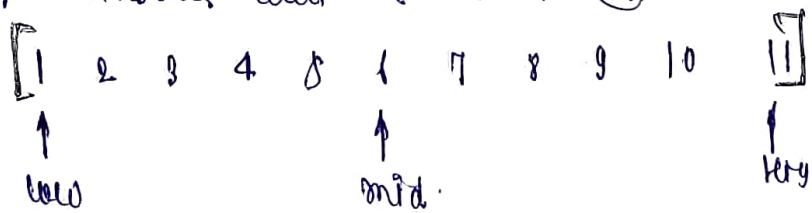
So, 

→ 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

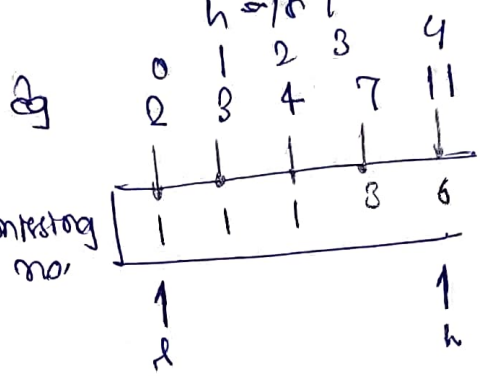
So, l = 18, m = 18, h = 18 | Yes

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)

3 1 1

0	1	2
~~1~~ ~~2~~ 3	1	1

ans[] = [1 (4) (7) (10) 13 (18) 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.