PAGE NO.
DATE : / /

Binary Search

18 ques

1. Binary Search
2. Order agnostic BS
3. 1st & last occurrence of an element
4. Count of element in a sorted array
5. # of times - array is rotated
6. Find an element in rotated sorted
7. Searching in nearly sorted array
8. Floor / Ceil of an element
9. Next letter
10. Index of last 1 in sorted array
11. Find the position of an element inf. sorted array
12. Minimum diff element in a sorted array
13. Bitonic array max element
14. Search in Bitonic array
15. Search in row wise + col wise sorted array
16. Find element in sorted array that appears once
17. Allocate min # of pages

sorted **

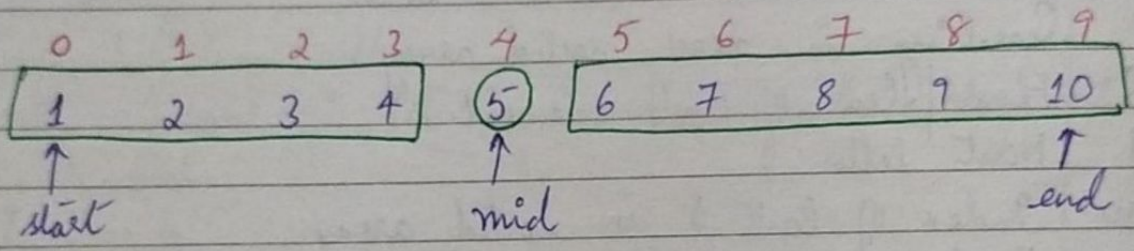
PAGE NO.
DATE : / /

Binary Search

arr[] : 1 2 3 4 5 6 7 8 9 10
ele : 2

Ans = 1 → o/p

SORTED

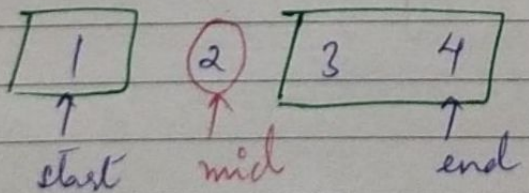


I mid = $\frac{0+9}{2}$ = (4)

→ mid element

→ 2 < 5

→ Hence, go to left of mid to search for element



II mid = $\frac{0+3}{2}$ = 1

2 == 2 ✓

Hence arr[mid] found

return index (1) Ans//

PAGE NO.
DATE : / /

→ start end ek dusre ki taraf move kar rahe h
→ kab tak loop chalega
↓
start <= end

Code :

```
int start = 0;
int end = n-1;

while (start <= end) {

    int mid = (start + end) / 2;

    if (ele == arr[mid])
        return mid;

    else if (ele < arr[mid])
        end = mid - 1;

    else
        start mid+1;

}

return -1;
```

to avoid int overflow

start + end - start
2

→ agar element mila hi nhi

Descending Sorted Array

```
if (ele < arr[mid])
    start = mid + 1;

else
    end = mid - 1
```

Order not known Search - Order Agnostic Search

check —

arr[0] < arr[1] ⇒ Ascending order
arr[0] > arr[1] ⇒ Descending order

1st and Last Occurrence of an Element

Given sorted array

⇒ **1st occurrence**

```
int start = 0, end = n-1;
int res = -1;

while (start <= end) {
    int mid = start + (end - start)/2;

    if (ele == arr[mid]) {
        res = mid;
        end = mid - 1;
    }
}
```

→ search continue rakhni hai or we are not sure ki ye 1st occurrence hai

PAGE NO. icon
DATE : / /

```
else if (ele < arr[mid])
        end = mid - 1;

else
        start = mid + 1;
```

```
}
return res;
```

⇒ last occurrence

```
start = mid + 1;
```

Count of an element in a sorted array

2 4 | 10 10 10 | 18 20

↑ ↑

4 - 2 + 1 = (3) Ans //

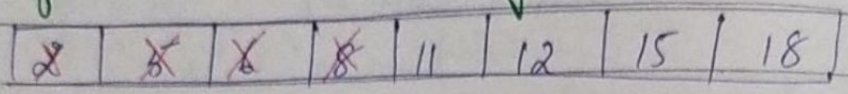
first & last occurrence last video

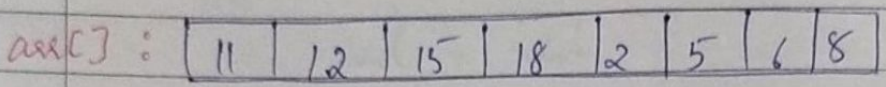
→ usage //

PAGE NO. icon
DATE : / /

Imp

of times a sorted array is rotated





4 times rotated

↓

O/P

2 5 6 8 11 12 15 18

↓

11 12 15 18 2 5 6 8

# of times rotated = Index of min^m element

Linear Search → O(n)

↓

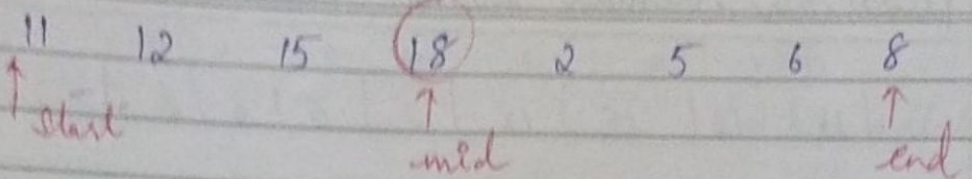
Better

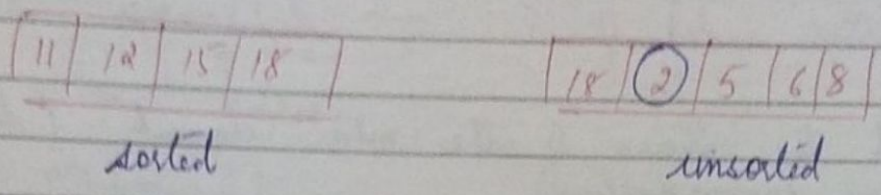
Given sorted array → BS → $O(\log n)$

11 12 15 18 (2) 5 6 8

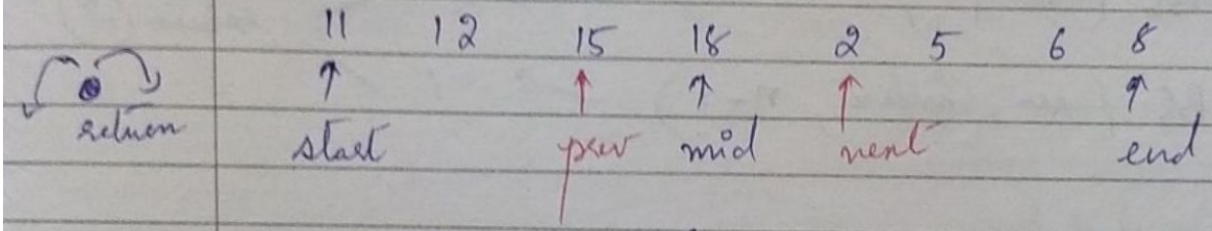
↓ MID

If it is a min^m element → then it will be smaller than both of its neighbours





So, basically we have to move in that direction, where the array part is unsorted




```
while (start <= end){
    mid = (start + end) / 2;

    next = (mid+1) % N
    prev = (mid + N-1) % N

    if ( arr[mid] <= arr[next] &&
         arr[mid] <= arr[prev] )
        return mid;

    if ( arr[start] <= arr[mid] ){
        start = mid + 1;
    }
    else if ( arr[mid] <= arr[end] ){
        end = mid - 1;
    }
}
```

mid → toh return kardenge
small → bada, bada

start = mid + 1; → we'll be searching right array part

end = mid - 1; → we'll be searching left array part

PAGE NO.
DATE

Print an Element in a Sorted Rotated Array

0	1	2	3	4	5	6	7
11	12	15	18	2	5	6	8

Print index of 15 → (2) → dp

(1) Find the index of the min^{m} element (done in previous problem)

(2) index

BS (arr, 0, index − 1) → 2

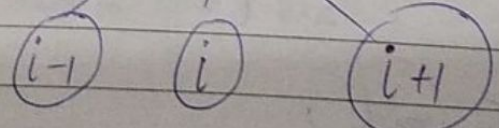
BS (arr, index, n − 1) → −1

→ return (2)

Searching in a Nearly Sorted Array

arr[]: | 5 | 10 | 30 | 20 | 40 |

Nearly Sorted ⇒ i^{th} index

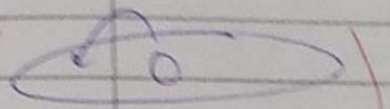


PAGE NO.
DATE : / /

BS
(sorted)

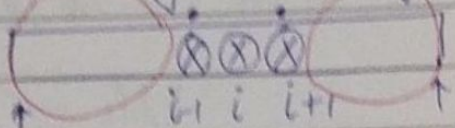
start end

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



end = mid - 1

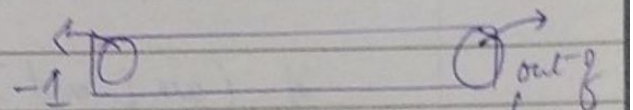
BS
(nearly sorted array)



start end

```
if(ele == arr[mid])    -①
   ele == arr[mid-1]   -②
   ele == arr[mid+1]   -③
```

if small
end = mid - 2

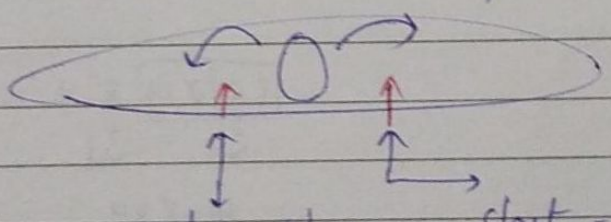


needs to be checked for both

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

if (mid-1 >= start && arr[mid-1] == ele)
    return mid-1;

if (mid+1 <= end && arr[mid+1] == ele)
    return mid+1;
```



end = mid - 2 start = mid + 2

PAGE NO. icon
DATE : / /

Find floor of an element in a sorted array

arr[] :	1	2	3	4	8	10	10	12	19

ele = 5

Floor of 5 : → Greatest element smaller than 5

1
2
3
4 → o/p

```
if (arr[mid] < ele){
        res = arr[mid];
        start = mid + 1;
}
else if (arr[mid] > ele) {
        end = mid - 1;
}

return res;
}
```

~~RIGHT wali me store kiya hai~~

1	2	3	4	8	10	10	12	19

mid ↓ mid ↓

res = 2 3 4 → o/p

PAGE NO. icon
DATE : / /

Find ceil of an element in a sorted array

Ceil of 5 : Smallest element greater than 5

o/p → 8

res = 8

Candidates

8 → Ans.
10
10
12
19

```
int res = 0;
if (arr[mid] < ele) {
        start = mid + 1;
}
else if (arr[mid] > ele) {
        res = arr[mid];
        end = mid - 1;
}
return res;
```

~~LEFT side me store karna hai~~

PAGE NO.
DATE : / /

Next Alphabetical Element

Given: [a c f h] key = f

o/p : h

Prob. St.

① similar to ceil problem

② If f is present, don't return f, return next alphabet

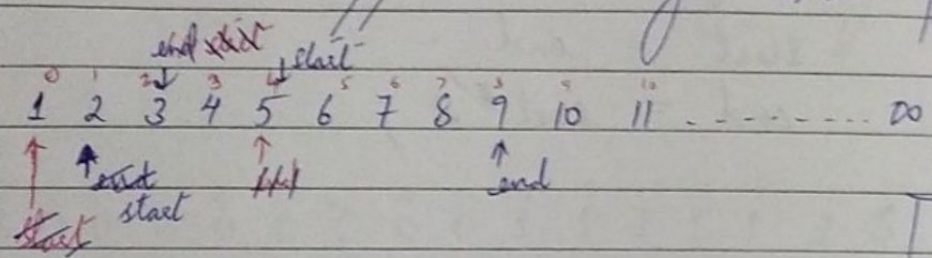
```
if (arr[mid] == key) {
        start = mid+1;        ⟶ search continue rakho
}
else if (arr[mid] < key) {
        low = mid+1;
}
else if (arr[mid] > key) {
        res = mid;
        high = mid-1;
}
```

PAGE NO. icon
DATE : / /

Find position of an element in an ∞ sorted array

Imp for F2F Interviews

(Cause ∞ array can't be given as an i/p)



Key = 7

key = 7, arr[end] = 2.

2 < 7.

⇒ end → end * 2;

&

⇒ Start ko purane wale end me le aayenge

⋮

Ye kate kate

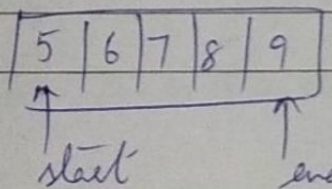
We reach a point where

start → index 4

end → index 9

& we saw that arr[end] 7 ~~element~~ key

Hence the reduced array is :-



Now perform Normal BS

```
int start = 0
int high = 1
while ( key > arr[end] ) {
    start = end;
    end = end * 2;
}
BS ( arr, start, end )
```

Find position of an element in an ∞ sorted array

0 0 0 0 0 . . . . . ∞

F2F interview ques.

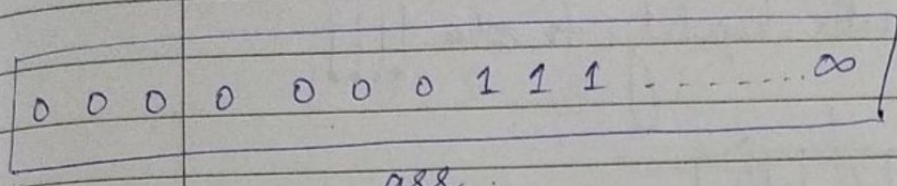
0	1	2	3	4		
0	0	0	0	0	. . .	0
↑ start	↑ end					

Logic.

high mark kone me dikkat aa rhi thi, hence like in prev. pgm end to index 1 pe le liya

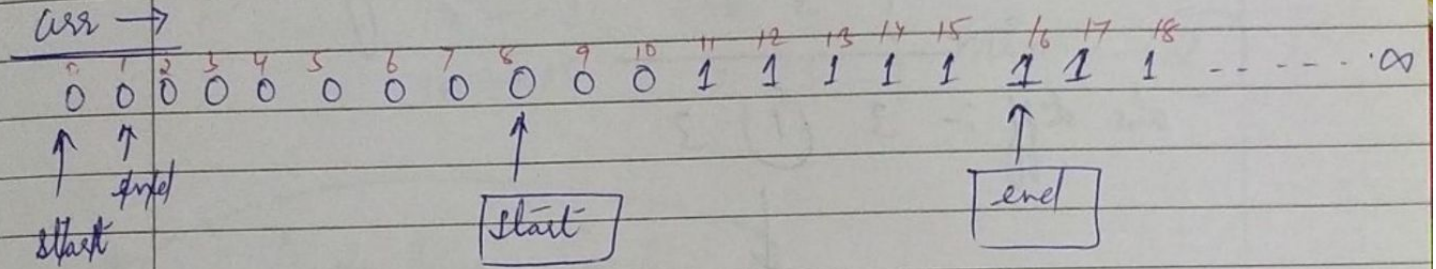
F2F interview //

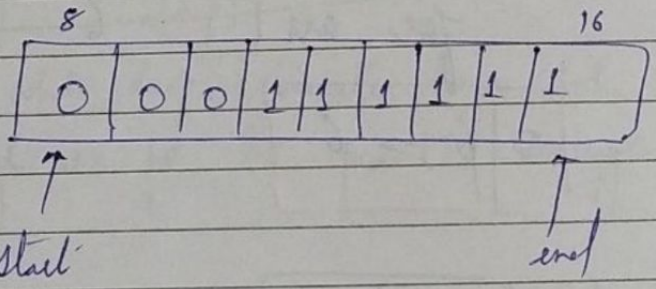
Find the index of first 1 in a Binary Sorted ∞ array



0, 1 in array

00...111...∞





$$mid = \frac{8+16}{2} = \frac{24}{2} = (12)$$

```
if (arr[mid] == key) {
    res = mid;
    end = mid - 1;
```

Combination of 2 ques ⇒ 1st occurrence of element + Search in ∞ array sorted

Interesting problem !

Minimum d/f element in a sorted array

* does not require any extra knowledge to solve !!!!

eg

	0	1	2
arr[] :	4	6	10

key = 7

7 7 7

---

abs. diff :- 3 (1) 3

↓

for arr[i] = 6

0	p = 6

eg

1	3	8	10	15

key = 12

12 12 12 12 12

---

11 9 4 (2) 3

↓ for arr[i] = 10

minimum value

0	p = 10

PAGE NO. icon
DATE : / /

if key present in array → return key → BS

1 3 8 (10) (15) [12]

minimum diff in 12's neighbours

10 − 12 = 2
10 − 15 = 3
} min = (2)

* Simple Binary Search

jab while loop terminate hoga toh low & high 12 ke neighbours pe hi point karenge

1 3 8 10 15
↑ ↑
high low

yeh koi coincidence nhi hai
Hamesha yehi hota hai

abs (arr[low] − key)
abs (arr[high] − key)

~~10 − 12~~ 15 − 12 = 3
~~10 − 15~~ 10 − 12 = 2
→ min (2)
Ans

Fav ***

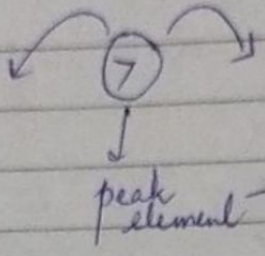
Peak Element

	0	1	2	3
arr[] :	5	10	20	15

→ unsorted array

↓

Toh bhi Binary Search !!!



| o/p → 2 | → 2nd ele

Eg. | 10 | 20 | 15 | 2 | 23 | 90 | 67 |

(20 > 10, 20 > 15) peak — peak (90 > 23, 90 > 67)

| o/p → 1 | 5 |

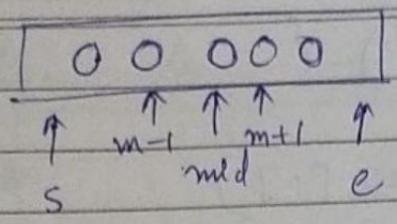
Eg. 10 3 20 40 50

peak — peak

(10 > 3) (50 > 40)

agar mid+1 greater hua to uske dir me move karenge

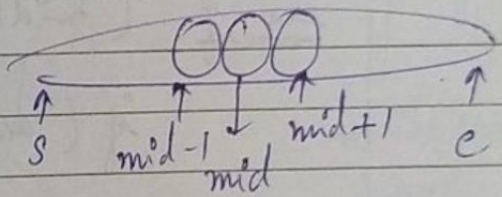
agar mid-1 greater hua toh uske dir me move karenge



mid - 1 agar bada hua → mid se → left
mid + 1 agar bada hua → mid se → right
(promising sides)

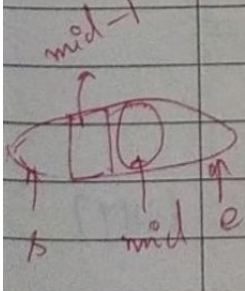
CODE :—

start
int ~~low~~ = 0
int end = size - 1



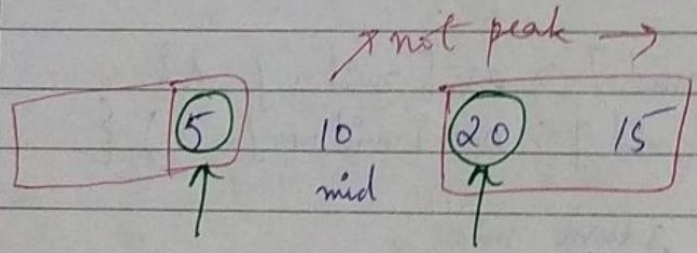

```
while (start <= end) {
    int mid = start + (end - start) / 2;

    if (mid > 0 && mid < n-1) {
        if (arr[mid] > arr[mid-1] &&
            arr[mid] > arr[mid+1]) {
            return mid;
        }
        else if (arr[mid-1] > arr[mid]) {
            end = mid - 1;
        }
        else {
            start = mid + 1;
        }
    }
```



```
else if (mid == 0) {
    if (arr[0] > arr[1])
        return 0
    else
        return 1
}
else if (mid == n-1){
    if (arr[n-1] > arr [n-2])
        return n-1
    else
        return n-2
}
```

3



→ Agar left jaane ki sochte hain → toh 5 compare hoga < 10 se / left ke element se

But we realize ki 5 toh peak ho hi nhi sakta WHY?

kyunki 5 < 10

→ So right side jaana is more promising

→ Hence, we will go right

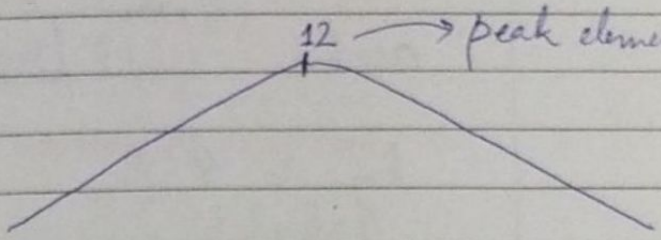
CRITERIA

PAGE NO.
DATE : / /

Bitonic Array - Find max element

↓

Array monotonically ↑
then monotonically ↓



Also, obv in this problem → there will only be 1 peak element

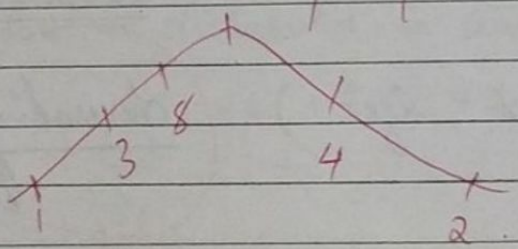
↓

unlike prev. ques

arr: | 1 | 3 | 8 | 2 | 4 | 2 |

↓

12 peak/max element



o/p = 12

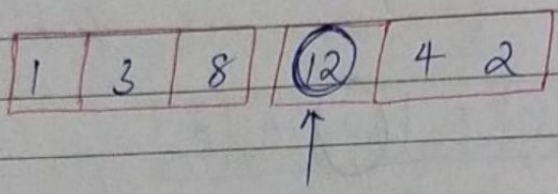
Max Element in Bitonic Array == Finding Peak Element

Search in Bitonic array

	0	1	2	3	4	5
arr[] :	1	3	8	12	4	2

key = 4

o/p = 4 → index



BS (arr, 0, index - 1) ascending order

BS (arr, index, size - 1) descending order

return -1

return 4

Print 4 Ans//

Make sure you write code for binary search in case of descending order sorted array

Best Question

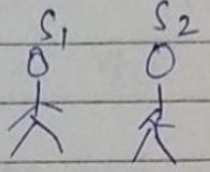
PAGE NO. icon
DATE : / /

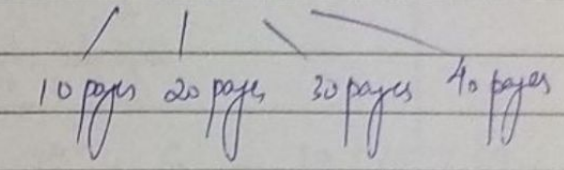
Allocate Minimum Number of Pages

	0	1	2	3
arr[] :	10	20	30	40

k : 2 → No. of students

→ not necessary ki array sorted ho

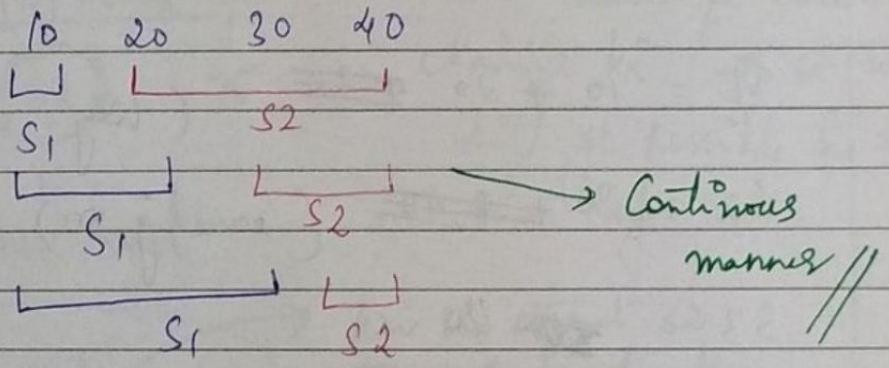


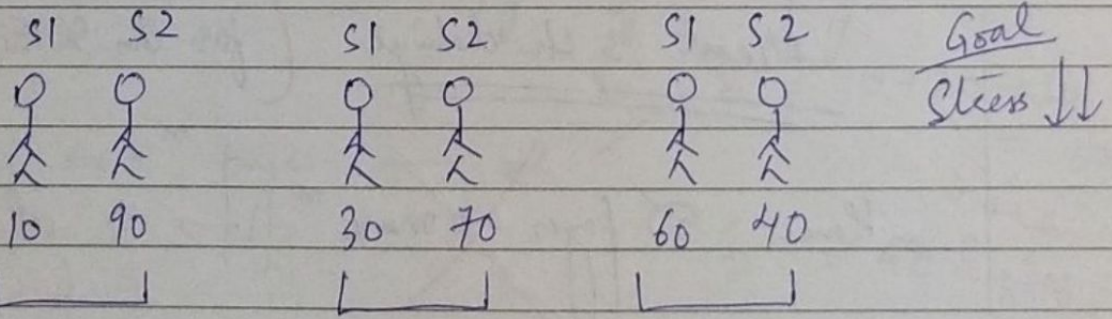


1 → Every student will receive atleast 1 book

2 → Pura pura book will be given to the student → aise nhi ki kuch page S1 ko de diye and kuch page S2 ko de diye

3 → Books have to distributed in continous manner



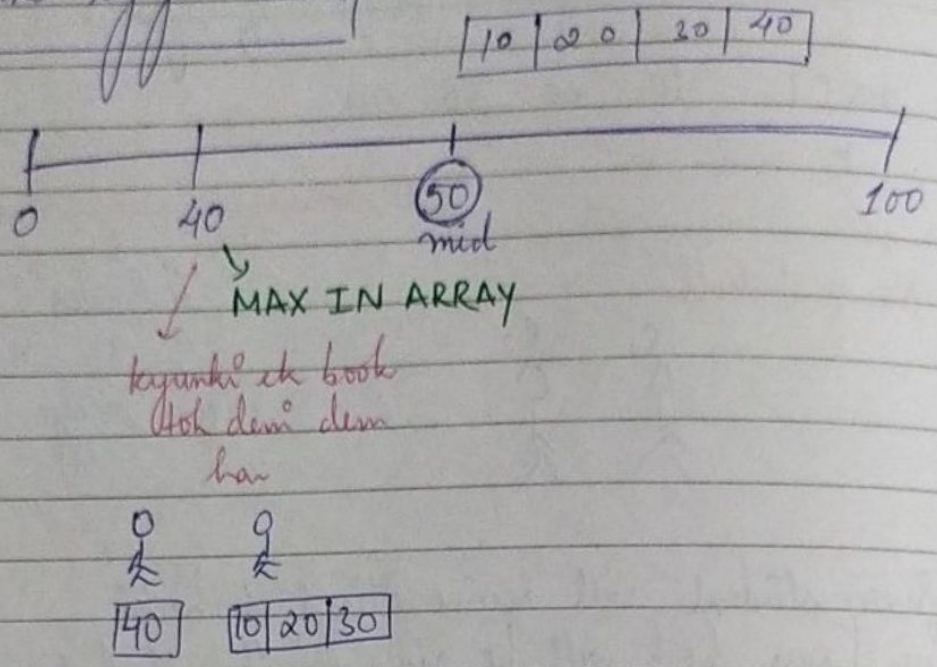


min^m = 60

o/p → 60

PAGE NO. icon
DATE : / /

Same layegi BS!!



mid = 50 ⟶ max^m no. of pages (koi bhi student 50 pages se jyada nhi padhega)

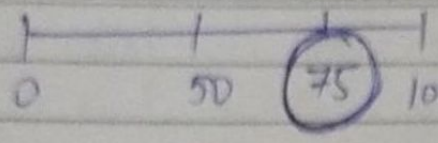
S1 = 10 + 20 ~~+ 30~~ (Why? exceeding 50)

S2 = 30 ~~+ 40~~ (exceeding 50)

S3 = ~~20~~ 40

Atleast 3 stu chahiye (for this SCHEME)

We have only 2 stu Hence 50 pages → max^m → Ye scheme nahi bana sakte



koi bhi student 75 pages se zyada nahi padhya

10	20	30	40

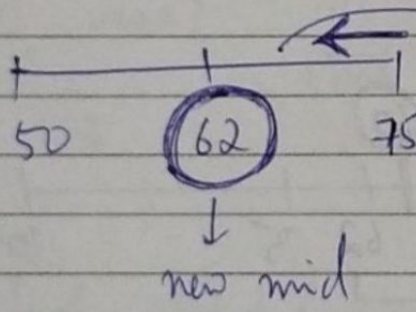
new mid = 75

S1 = 10 + 20 + 30 ~~+ 40~~

S2 = 40

If max^m pages → 75 < $S_1 = 10+20+30 = 60$, $S_2 = 40$

Res = 75 ✓ SCHEME SUCCESSFUL



75 se kam pe check krna chahiye ki 75 se kam pe bhi ye possible ho rha hai kya

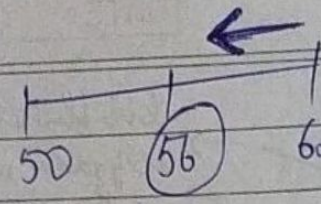
mid = 62 → koi bhi student 62 pages se zyada nahi padhya

If max^m pages → 62 < $S_1 = 10+20+30 = 60$, $S_2 = 40$

✓ SCHEME SUCCESSFUL

Res = ~~75~~ 62.

(update result)



new mid = 56

$S_1 = 10 + 20 + \cancel{30}$

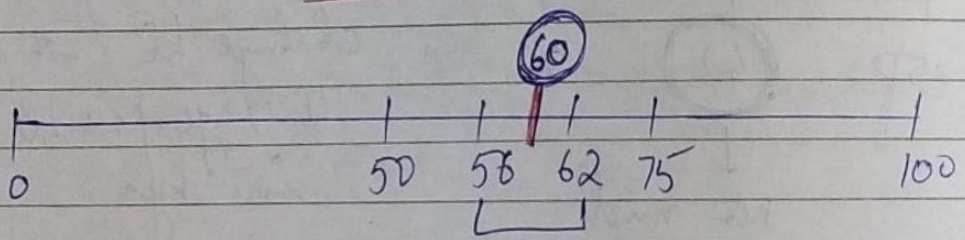
$S_2 = 30 + \cancel{40}$

$S_3 = 40$

But we have only 2

(3) students are required.

✗ Scheme Unsuccessful



$S1 = 10 + 20 + 30$

$S2 = 40$

Students = 2 ✓

* Better start the number line with 40 (cause 0 - 40 me we don't have to go)

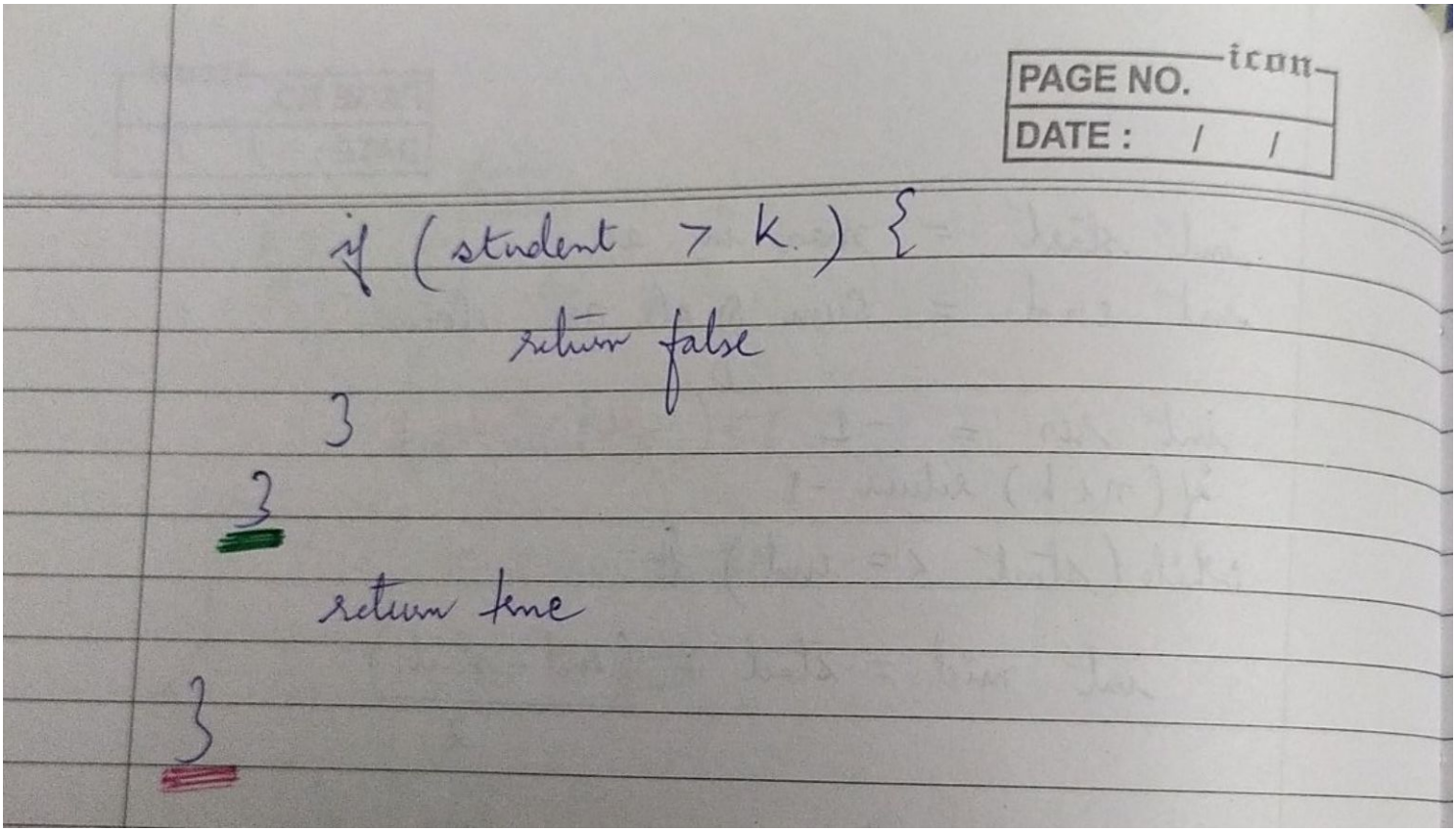
PAGE NO.
DATE : / /

int start = max in arr
int end = Sum of all arr elements

int res = -1 (for NP case)
if (n < k) return -1
while (start <= end) {
int mid = start + (end - start) / 2

if (isValid(arr, n, k, mid) == true) {
res = mid
end = mid - 1
}
else {
start = mid + 1
}
}

bool isValid (arr, n, k, max) {
int student = 1
int sum = 0
for (int i = 0; i < n; i++) {
sum += arr[i]
if (sum > max) {
student ++
sum = arr[i]
}


PAGE NO.
DATE : / /
if (student > k) {
return false
}
}
return true
}