Today's Content :-

→ Searching in Sorted Array

→ Sqrt

→ A^{th} Magical Number.

→ Median of Two arrays.

Today's Quote.

You have to pick something big to work upon,
because its hard to commit your life to something small.

Ques Search in a sorted but rotated array. → elements are distinct

9 4 8 10 15

k = 2

k=1 => 15 2 4 8 10

k=2 => 10 15 2 4 8

1 2 3 4 5 6

if array is rotated

(A[0] > A[n-1]) {

yes

} else {

B.S. → NO

}

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

K

Brute force :- Do linear Search

T.C → O(N)

S.C → O(1)

If largest element idx is given, P,

apply B.S (0, P), (P+1, n-1).

Twist :- Largest elent idx is not given.

4 5 8 10 1 2 3

find local maxima using B.S,

then, → B.S.

the apply B.S in both parts.

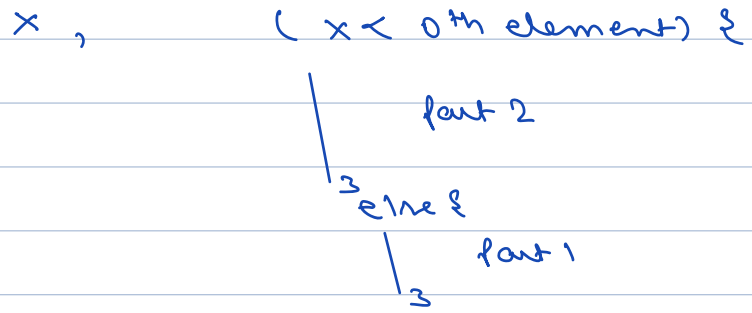
Twist :- Do it in One B.S.

1 2 3 4 5 8 10

I II

4 5 8 10 1 2 3

Part 1 > Part 2



Get mid, find in which part our middle is and in which part, our target is,
if both are in diff part's, we'll move mid towards the target, else apply normal B.S.

0	1	2	3	4	5	6	7	8	9	10	11
10	20	30	1	2	3	4	5	6	7	8	9

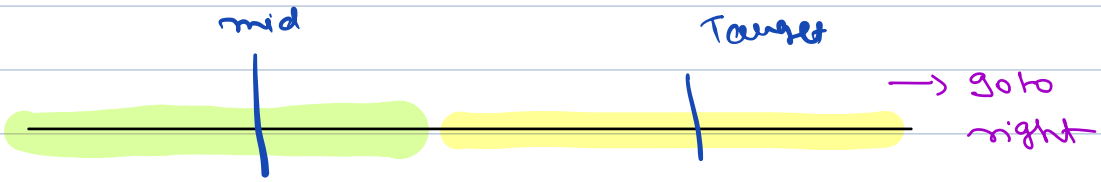
k = 20 → I

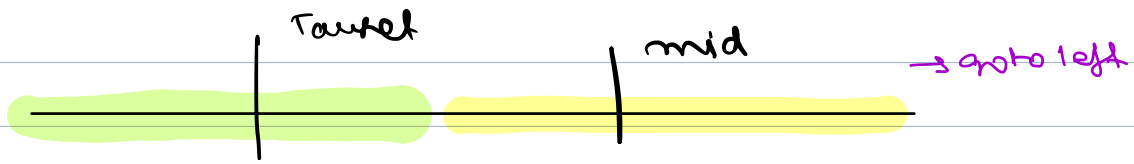
l	h	mid	mid Area	Target Area.	
0	11	5	2	1	goto left
0	4	2	1	1	goto left
0	1	0	1	1	goto right
1	1	1	reached.		

k = 60 → 2.

0	1	2	3	4	5	6
70	80	90	100	40	50	60

l	h	mid	mid Area	Target Area.	
0	6	3	1	2	goto right
4	6	5	2	2	goto right
6	6	reached.			





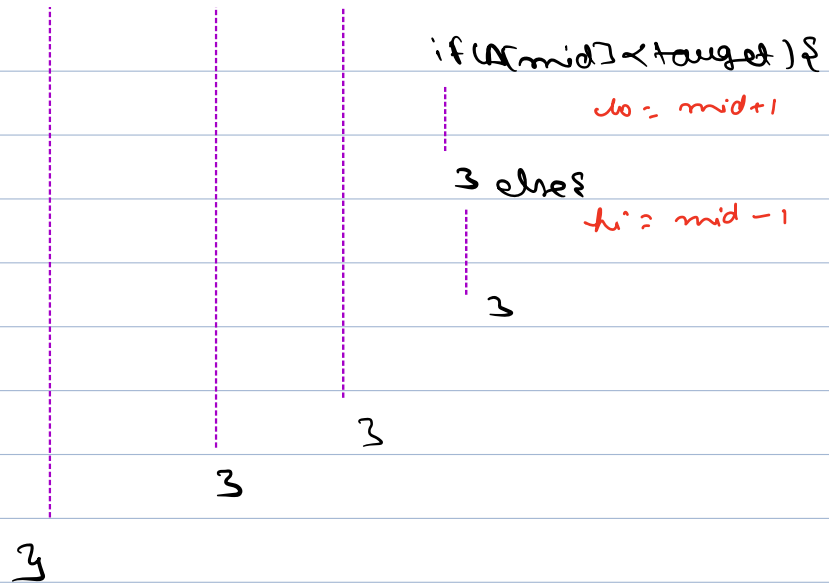
apply B.S.

```
lo = 0,   hi = n-1;
while ( lo <= hi) {
    mid =   lo + (hi-lo)/2 ;

    if (A[mid] == target) {
        return mid.
    }

    if ( target < A[0] ) {   // target Part -> 2

        if (A[mid] >= A[0]) {  // mid part -> 1
            lo = mid+1 ;
        } else {   //mid -> part 2
            if ( A[mid] < target) {
                lo = mid+1
            } else {
                hi = mid-1
            }
        }
    } else {   // target part - 1
        if(A[mid] < A[0]) {   // mid part 2
            hi = mid-1
        }
        else {  // mid part 1
```



T.C → O(log n)

S.C → O(1)

1 2 3 4 5 6 7 8 9 10

Ques Given +ve, N, find sqrt(N);

↳ floor sqrt(N).

sqrt (25) = 5
sqrt (20) = 4
sqrt (10) = 3

T.C → O(√N), S.C → O(1)

```
i = 1; ans;
while (i*i <= N) {
    ans = i;
    i++;
}
```

B.S :-

Search Space → 1 to N

Target → floor (sqrt(N))

Case-1 mid * mid = N
return mid;

Case 2: mid * mid > N
go to left

Case 3 :- mid * mid < N
ans = mid;
go to right

N = 20

1 2 3 4 5 6 7 8 9 10

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

N = 50 ; ans = 6

l	h	m		
1	50	25	25×25 > 50	goto left
1	24	12	12×12 > 50	goto left
1	11	6	6×6 < 50	ans = 6, goto right
7	11	9	9×9 > 50	goto left
7	8	7	7×7 < 50	ans = 7, goto right
8	8	8	8×8 > 50	goto left
8	7	break ;		

```
int sqrt(int n) {
    ans = -1;
    l = 1, h = n;
    while (l <= h) {
        m = l + (h-l)/2
        // logic
    }
}
```

1 3

T.C → log N

S.C → O(1)

8:06 am - 8:16 am

Ques why not ternary search? doubt session.

A^{th} Magical Number

multiples of 3 → [1 100]

$\frac{100}{3} \Rightarrow 33$

4 from → [1 100] $\Rightarrow \frac{100}{4}$

6 from [1 to 100] $\Rightarrow \frac{100}{6} = 16$

4 or 6 from [1 to 100] $\Rightarrow$

$\frac{100}{4} + \frac{100}{6} - \frac{100}{12} = \ldots$ → LCM of 4 & 6.

4 8 12 16 20 24 ...

6 12 18 24 30 36 ...,

> no. of multiples of A or B, from $1 \to x$, $\Rightarrow \frac{x}{A} + \frac{x}{B} - \frac{x}{LCM(A,B)}$

Ques. A, B & C → A^{th} Magical No.

↳ A no. is said magical if it is divisible by B or C.

e.g 1) 3 C A

2 3 8

↳ 2, 3, 4, 6, 8, 9, 10, 12, 14

ex 2)

B	C	A
4	6	5

4, 6, 8, 12, 16, 18, →

1) Brute force

```
d = 0;
for ( i=1;  ;  i++ ) {        // a * min(b,c);
    if ( i%B == 0 || i%C == 0 ) {
        d++;
    }
    if (d == N) {
        return i;
    }
}
```

→ [min(b,c), a * min(b,c)]

Search space

Target → Nth magical no.

Ex,

B	C	A
4	6	10

Is 36 my 10th magical no'.

$\frac{36}{4} + \frac{36}{6} - \frac{36}{12} \Rightarrow 12$

36 37 38 39
x x x x

goto left.

is 24 your 10th magical No.

$\frac{24}{4} + \frac{24}{6} - \frac{24}{12} = 8$

22 23 24 goto right
x x x

Tracing.

B	C	A
5	7	4

lo	hi	mid	
5	20	12	$\frac{12}{5} + \frac{12}{7} - \frac{12}{35} \Rightarrow 3$ goto right.
13	20	16	$\frac{16}{5} + \frac{16}{7} - \frac{16}{35} \Rightarrow 5$. goto left
13	15	14	$\frac{14}{5} + \frac{14}{7} - \frac{14}{35} \Rightarrow 4$. Ans = 4, goto left
13	13	13	$\frac{13}{5} + \frac{13}{7} - \frac{13}{35} = 3$ goto right
14	13	Break	

B = 5, C = 7, A = 3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	1	1	2	2	2	3	3	3	3	4	5

Pseudo Code :-

$LCM(x,y) = \frac{x*y}{gcd(x,y)}$

```
Nth Magical No. (A, B, C) {
    lo = min(B, C);
    hi = A * min(B, C);
    lcm = lcm(B, C)
    while (lo <= hi) {
        m = lo + (hi - lo)/2
        int co = m/B + m/C - m/lcm(B,C)
        if (co < A) {
            lo = m+1
        } else if (co > A)
            hi = m-1
        }
        else {
            ans = m;
            hi = m-1
        }
    }
    return ans;
}
```

Median of Two Sorted Array

Median.

e.g 1)

A[] = 1, 4, 5
B[] = 2, 3
] → 3

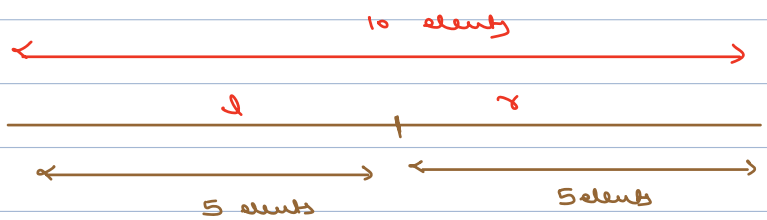
1, 2, 3, 4, 5 → 3

1, 2, 3, 4, 5, 6
↳ $\frac{3+4}{2}$ => 3.5

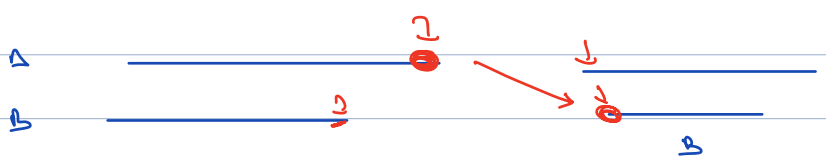
e.g 2) A[] = 1, 2, 3
B[] = 4
] → 2.5

e.g 3).

A[] → 1, 3, 4, 7, 10, 12
B[] → 2, 3, 6, 15
] → 10 elements.



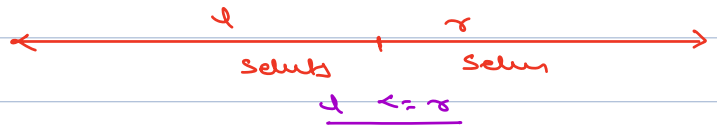
l <= r



Break 9:05pm - 9:01pm :-

A[] → 1, 3, 4, 7, 10, 12

B[] → 2, 3, 6, 15



Case-1 , A we choose 4.

l R ×

A[] → 1, 3, 4, 7 10, 12

B[] → 2 3, 6, 15

Case-2 , A we choose 2,

l, r,

A[] → 1, 3, 4, 7, 10, 12 ×

B[] → 2, 3, 6, 15

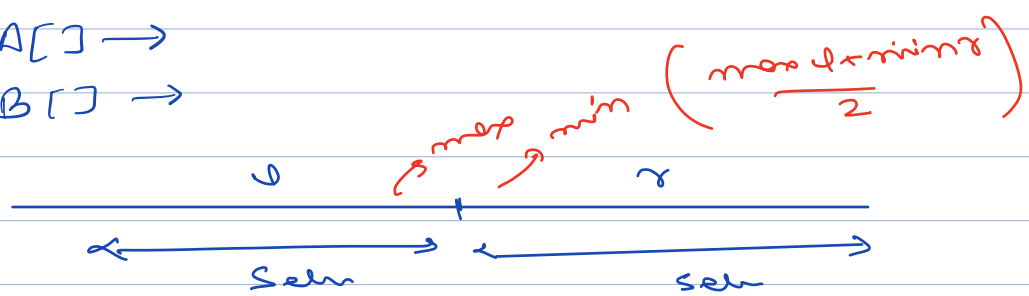
Case-3 :- (we choose 3 elements from A)

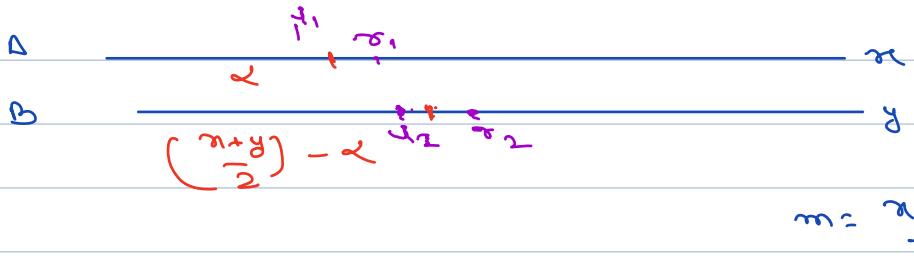
1, 3, 4 7, 10, 12

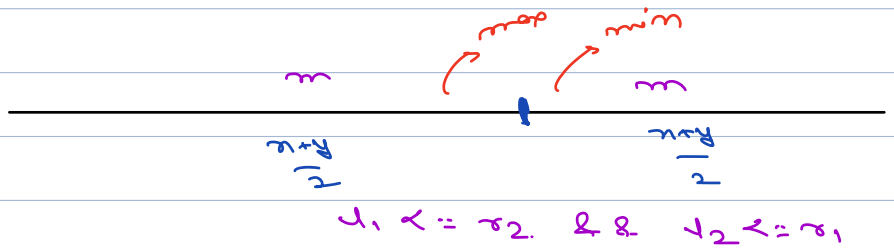
2, 3, 6, 15

A[] →

B[] →







cut 1

arr 1 → 7, | 12, 14, 15] 10 elmts

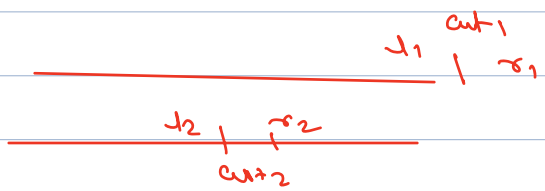
arr 2 → 1, 2, 3, 4, | 9, 11

cut 2

$\frac{7+9}{2} \Rightarrow 8$.

B.S → no. of elmts of array 1 on left side.

l	hi	mid	
0	4	2	goto left
0	1	0	goto right,
1	1	1	



nums1 <= nums2

```
findMedian ( int [] nums1, int [] nums2) {
    if ( nums2.size < nums1.size() ) {
        return findMedian (nums2, nums1);
    }
    int n1 = nums1.length;
    int n2 = nums2.length;
    lo = 0;
    hi = n1
    halfreq = (n1+n2+1)/2;
    while (lo <= hi) {
        mid = lo + (hi - lo)/2
        cut1 = mid;
        cut2 = (n1+n2+1)/2 - cut1;
        l1 = arr1[cut1-1];
        l2 = arr2[cut2-1];
        r1 = arr1[cut1];
        r2 = arr2[cut2];
        if ( l1 <= r2 && l2 <= r1 ) {
            return (max(l1,l2) + min(r1,r2))/2
        }
        else if (l1 > r2) {
            go left;
            hi = mid-1
        }
        else {
            lo = mid+1
        }
    }
    if ((n1+n2)%2 == 0) {
        return (max(l1,l2) + min(r1,r2))/2
    }
    else {
        return max(l1,l2);
    }
}
```

cut1 = 0 & l1 = -∞

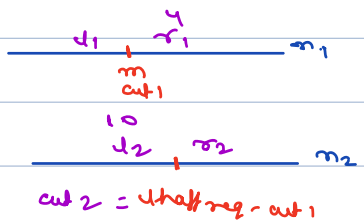
cut2 = 0; l2 = -∞

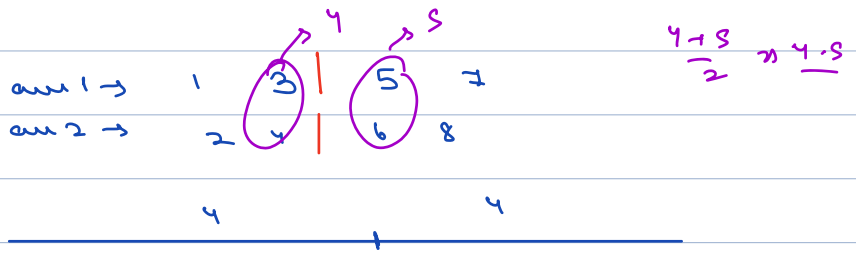
if cut1 == n1, r1 = +∞

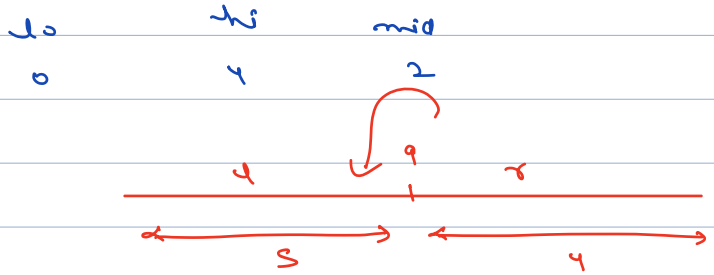
if cut2 == n2, r2 = +∞

halfreq = (n1+n2)/2

cut2 = halfreq - cut1

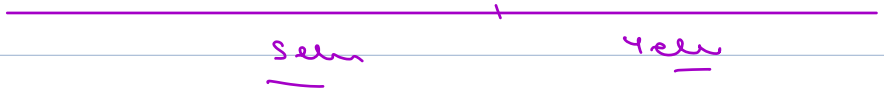




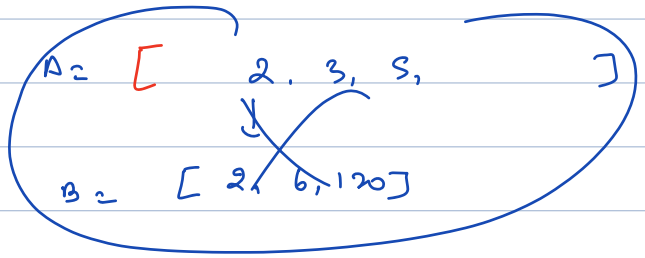


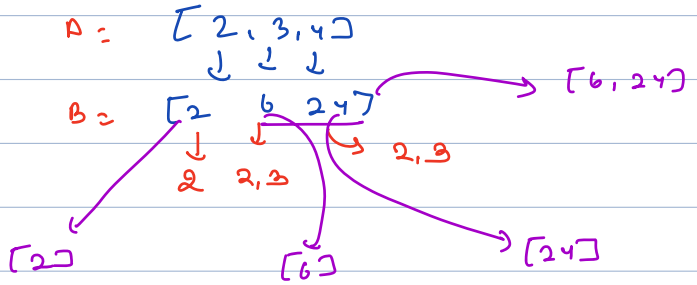
arr 1 → 7, 12, 14, 15 } 9 elmts

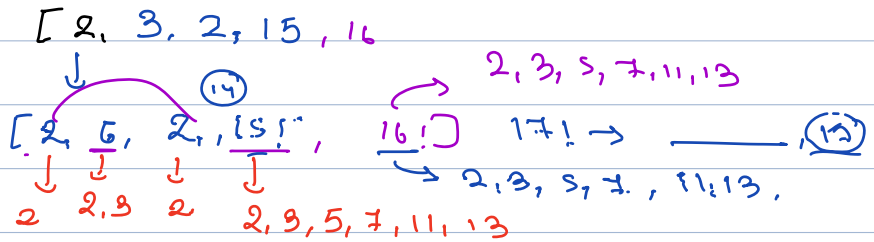
arr 2 → 1, 2, 3, 4, 9

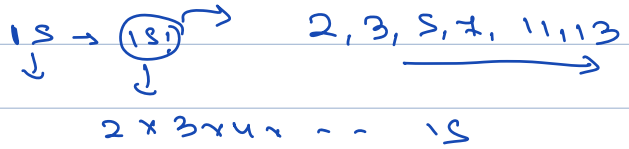


lo	hi	mid
0	4	2
0	1	0
1	1	1









13 14 15 16 17, 19, 20, 21, 22, 23