Today's content

① Searching an element in sorted. rotated array

② Sqrt

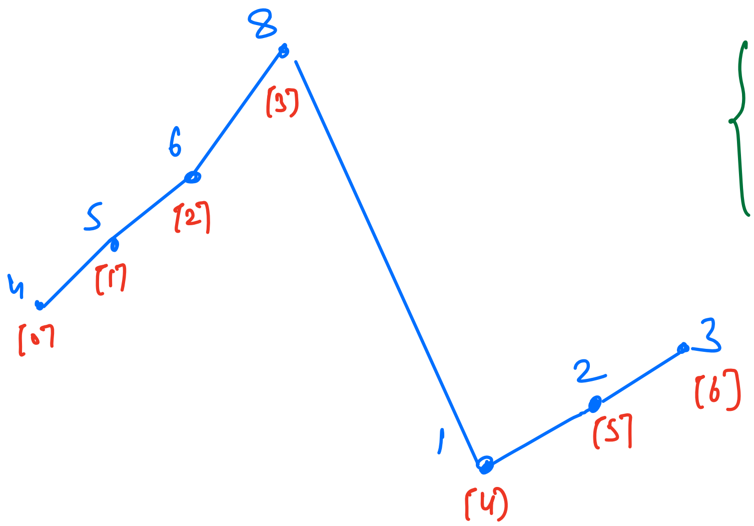
③ Median of two sorted arrays (Google)

Q.1 Search an element in sorted and rotated array. [distinct]

[8, 10, 15, 2, 4] , K=4 ans → 4.
0 1 2 3 4

Quiz. → [4, 5, 6, 8, 1, 2, 3] , K=2 ans = 5.
0 1 2 3 4 5 6

idea.1. → Linear Search T.C → O(N) , S.C → O(1)



element ≥ arr[0] ⇒ part 1
element < arr[0] ⇒ part 2

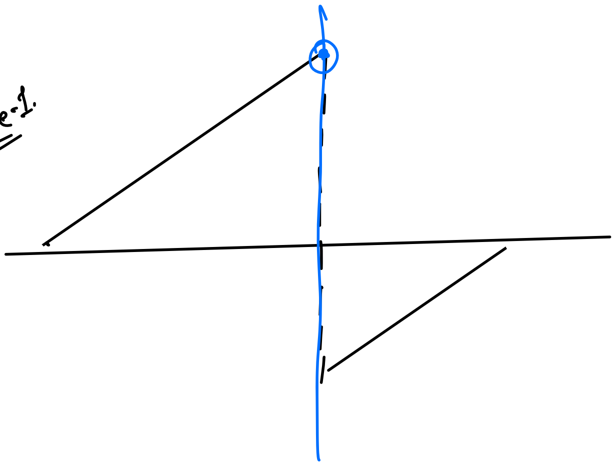
idea.2.→ if idx of largest element is known → p1

Apply B.S → [0, p1] and BS → [p1+1, N-1]

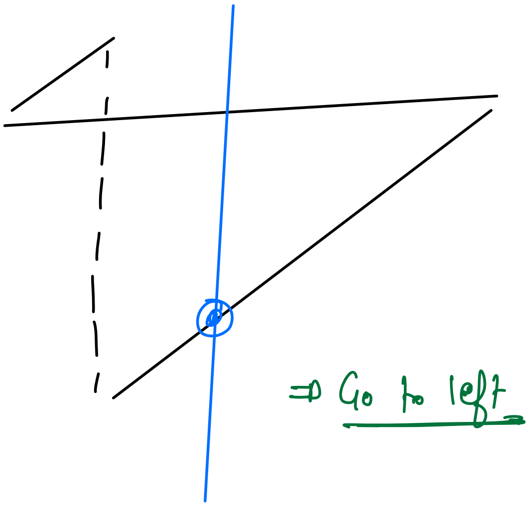
T.C → $O(\log_2 N)$, S.C → O(1)

Q.→ How to find index of largest element in optimized way?

Case-1.

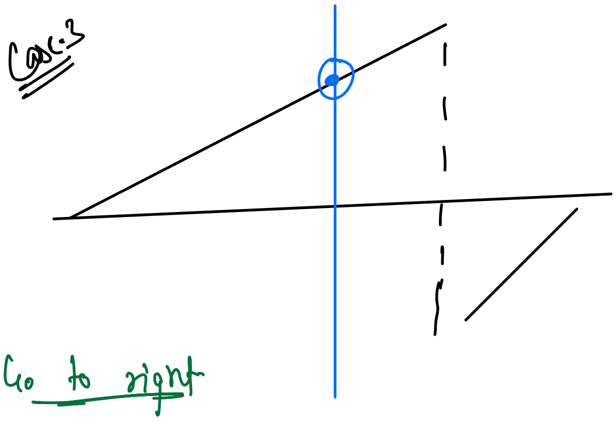


Case-2.



⇒ Go to left

Case-3



Go to right

target → idx of largest ele

search space → [0, N-1]

#code: →

```
l = 0, r = N-1
while( l <= r){
        mid = (l+r)/2;
        if (arr[mid-1] < arr[mid] && arr[mid+1] < arr[mid]){
                return mid;
        }
        else if (arr[0] < arr[mid]){
                l = mid+1;
        }
        else{
                r = mid-1;
        }
}
```

[T.C → O(log N)
S.C → O(1)]

[4, 5, 6, 8, 10, 1, 2, 3] , K=2

0 1 2 3 4 5 6 7

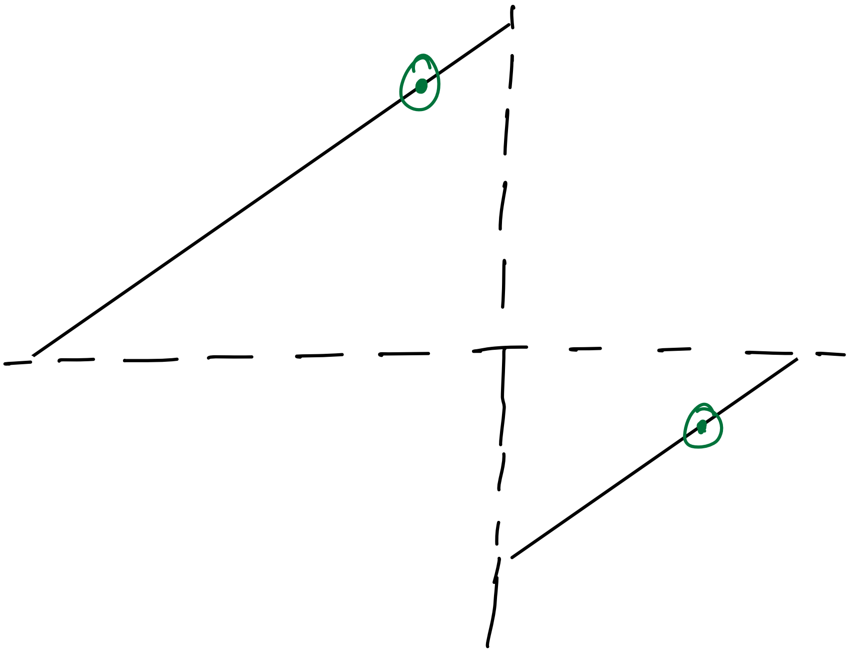
l r

l	r	mid
0	7	$\frac{0+7}{2} = 3$
4	7	$\frac{4+7}{2} = 5$
4	4	$\frac{4+4}{2} = 4$

l = mid+1

r = mid-1

return mid.



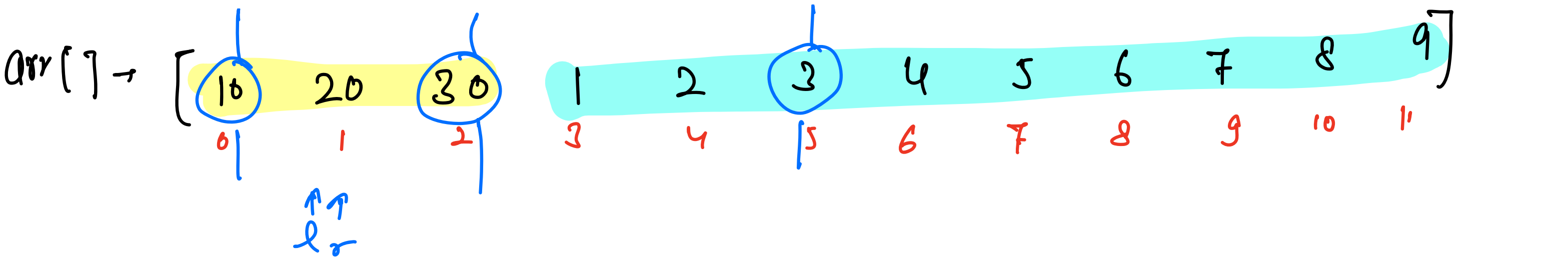
```
if (element < arr[0])
        Its
        part 2
else {
        part 1
}
```

Twist → You can apply Binary Search only once.

Get mid., Find in which part our middle element is lying and in which part target element is lying.

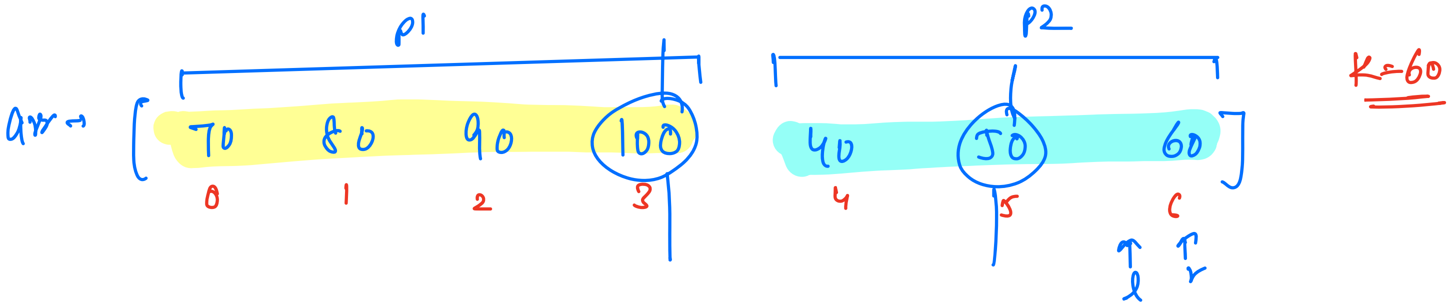
If both are lying in different parts, we will move mid towards the target, otherwise, apply Binary Search.

K=20

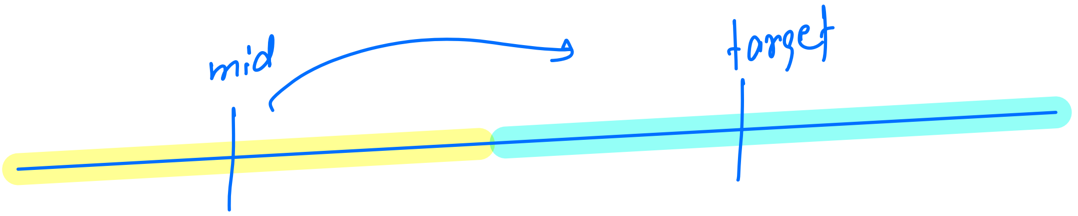


l	r	mid	mid-area	target-area	
0	11	$\frac{0+11}{2} = \boxed{5}$	p2	p1	Go to left.
0	4	$\frac{0+4}{2} = \boxed{2}$	p1	p1	Go to left
0	1	$\frac{0+1}{2} = 0$	p1	p1	Go to right
1	1	$\frac{1+1}{2} = 1$	p1	p1	return mid.

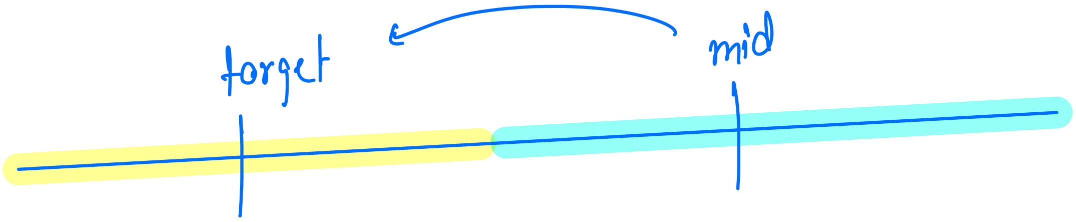
All elements in p2 < arr[0]



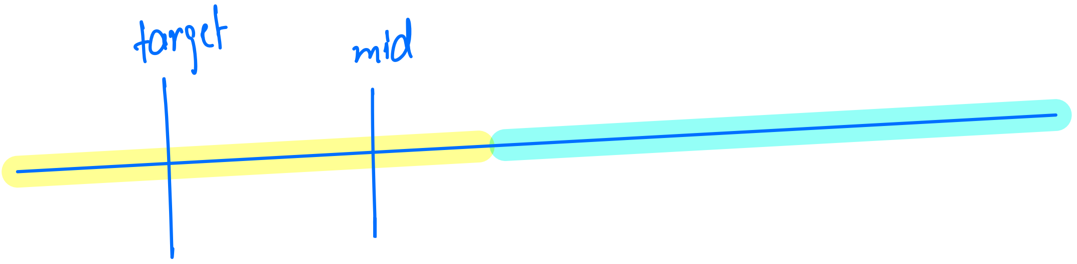
l	r	mid	part of mid?	part of target?	
0	6	$\frac{0+6}{2} = 3$	p1	p2	Go to right l=mid+1
4	6	$\frac{4+6}{2} = 5$	p2	p2	Go to right l = mid+1
6	6	$\frac{6+6}{2} = 6$	p2	p2	return mid.



Go to right.



Go to left.



Binary Search

#code. l=0 , r= N-1

```
while ( l <= r){
        mid = (l+r) / 2;
        if (arr[mid] == K) { return mid }
        if ( target < arr[0]){    // target in p2
                if( arr[mid] >= arr[0]) {  // mid in p1
                {
                        l = mid+1;
                }
                else {          // mid in p2
                {
                        if ( arr[mid] < target) {
                        {
                                l = mid + 1
                        }
                        else {
                        {
                                r = mid - 1
                        }
                }
        }
        else {            // target in p1
                if( arr[mid] < arr[0] ) {  // mid in p2
                {
                        r = mid - 1
                }
                else {          // mid in p1
                {
                        if ( arr[mid] < target) {
                        {
                                l = mid + 1
                        }
                        else {
                        {
                                r = mid - 1
                        }
                }
        }
}
return -1;
```

T.C → $O(\log_2 N)$

S.C → $O(1)$

Q.1 find floor(sqrt(N)) .

N = 10	→	3
N = 16	→	4
N = 29	→	5

idea-1.

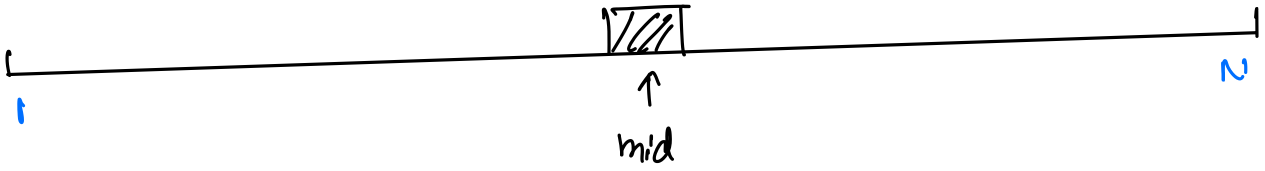
```
i = 1 , ans = 1
while ( i*i ≤ N ){
        ans = i ;
        i++
}
return ans;
```

[T.C → O(√n) , S.C → O(1)]

1*1 ≤ 29	ans = 1
2*2 ≤ 29	ans = 2
3*3 ≤ 29	ans = 3
4*4 ≤ 29	ans = 4
5*5 ≤ 29	ans = 5
6*6 ≰ 29	

target → floor(sqrt(N))

search-space → [1 – N]



mid * mid < N

ans = mid

// Go to right

l = mid + 1

mid * mid == N

return mid

mid * mid > N

// Go to left

r = mid - 1

$N = 29.$

l	r	mid	Compare mid*mid with N
1	29	$\frac{1+29}{2} = 15$	$15*15 > 29 \Rightarrow r = mid-1$
1	14	$\frac{1+14}{2} = 7$	$7*7 > 29 \Rightarrow r = mid-1$
1	6	$\frac{1+6}{2} = 3$	$3*3 < 29 \Rightarrow ans = 3$ $l = mid+1$
4	6	$\frac{4+6}{2} = 5$	$5*5 < 29 \Rightarrow ans = 5$ $l = mid+1$
6	6	$\frac{6+6}{2} = 6$	$6*6 > 36 \Rightarrow r = mid-1$
6	5	⇒ stop.	

T.C → $O(\log_2 N)$

S.C → $O(1)$

Median of an array →

⇒ middle element in sorted array.

10, 20, 30, 40, 45, 46, 50, 60

Median of two sorted arrays →

A[] → [1, 4, 5] → N   ans = 3.

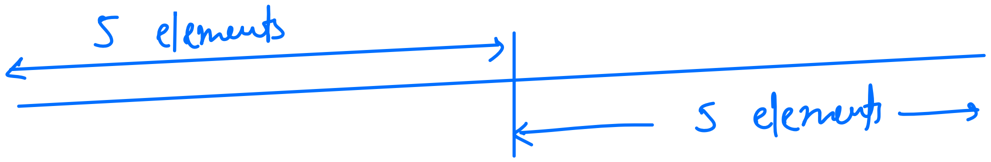
B[] → [2, 3] → m

A[] → {1, 3, 4, 7, 10, 12}   ans = 5.

B[] → {2, 3, 6, 15}

idea-1. → Create a merged sorted array & find middle element / elements.   T.C → O(N+m) , S.C → O(N+m)

10 elements

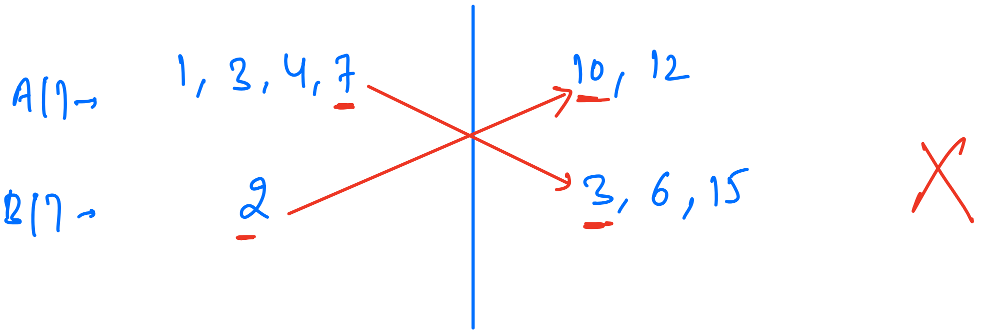


[l ≤ r]

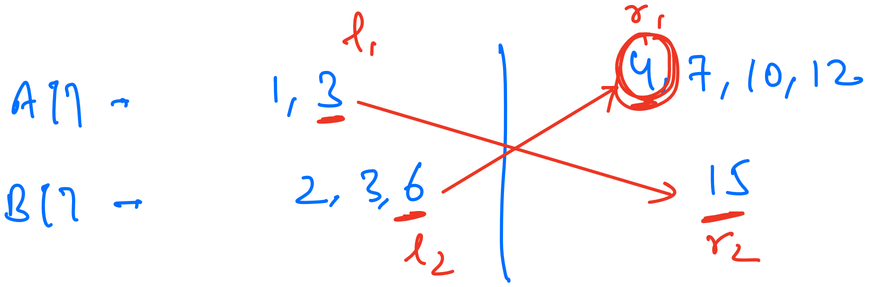
A[] → { 1, 3, 4, 7, 10, 12 } → N

B[] → { 2, 3, 6, 15 }

Case-1: If we choose 4 elements on l.h.s from A[]



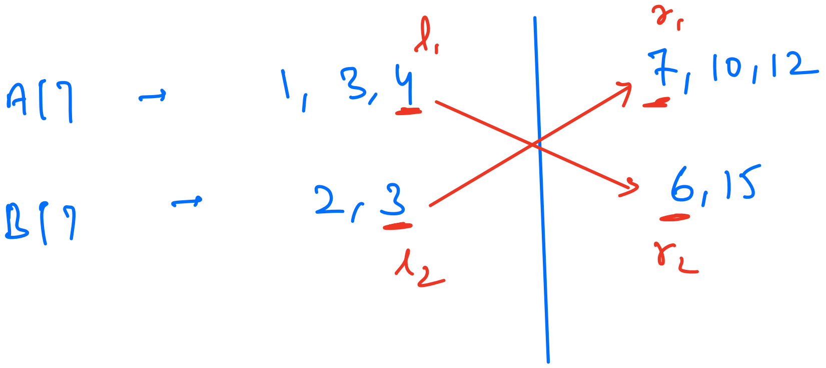
Case-2: If we choose 2 elements on l.h.s from A[]

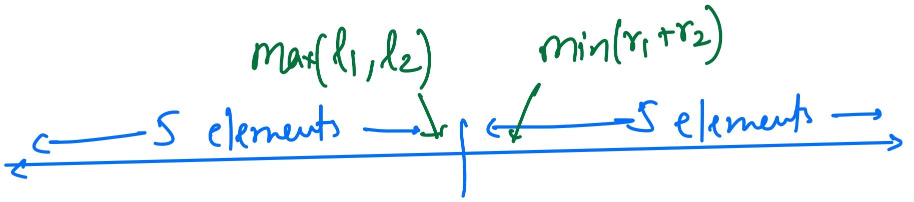


$l_1 \le r_2$ ✓

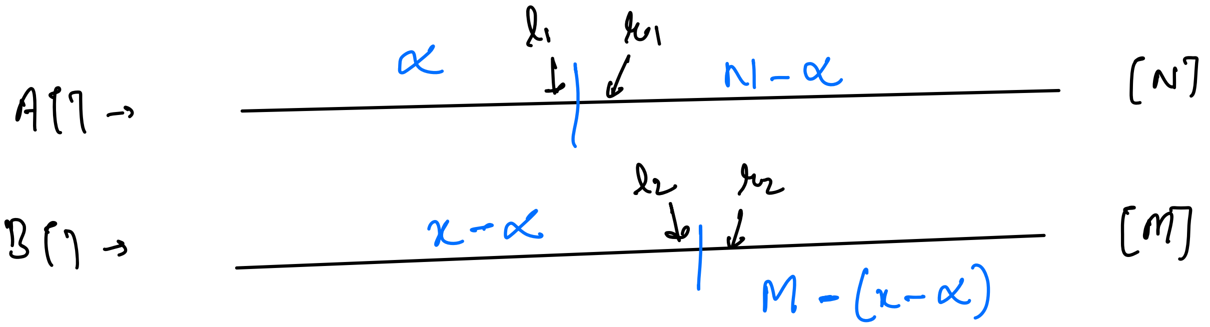
$l_2 \le r_1$ ✗

Case-3: If we choose 3 elements on l.h.s from A[]

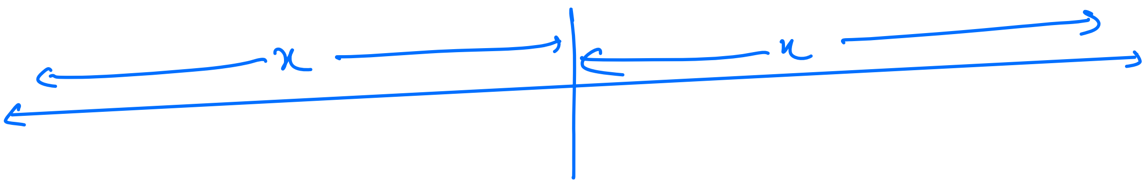




ans = $\frac{4+6}{2} = \frac{10}{2} = 5.$



$x \to \frac{N+M}{2}$



$$\{ l_1 \le r_2 \ \&\& \ l_2 \le r_1 \}$$

c_1

A[] → { 7, 12, | 14, 15 }

B[] → { 1, 2, 3, | 4, 9, 11 }

c_2

] → 10 element

N+M → 10

$x = \frac{N+M}{2} = 5$

5 elements | 5 elements

B.S → no. of elements to be on l.h.s from A[].

l	r	mid	is this a valid split?
0	4	$\frac{0+4}{2} = 2$	$l_1 > r_2$ ⇒ Go to left.
0	1	$\frac{0+1}{2} = 0$	$l_2 \ge r_1$ ⇒ Go to right
1	1	$\frac{1+1}{2} = 1$	$\frac{7+9}{2} = \frac{16}{2} = 8$ return 8

#code.

```
double findMedian( int[] A, int[] B , int N, int m){
    if( m> N){
            return find median( B[], A[], m, N);
    }

    l = 0   , r = N;

    while( l ≤ r){
        mid = (l+r)/2;
        cut1 = mid;
        cut2 = x - mid;
          l1 = cut1 ≥1 ? A[cut1-1] : -∞
          r1 = cut1 < N ? A[cut1] : ∞
          l2 = cut2 ≥1 ? B[cut2-1] : -∞
          r2 = cut2 < M ? B[cut2] : ∞
        if( l1 ≤ r2  && l2 ≤ r1){
            if(N+m%2 ==0){
                    return  max(l1,l2) + min(r1,r2) ;
                            ------------------------
                                       2
            }
        }   else { return max(l1,l2); }
        else if ( l1 > r2){
                    r = mid-1
        }
        else{
                    l = mid +1
        }
    }
}
```

$$x = \frac{N+m+1}{2}$$

$T.C \rightarrow O(\log_2 \max(N,m))$

$S.C \rightarrow O(1)$

9, 30, 34] ⇒ Comperator

23, 230, 134, 15, 19.

Unit's → [19, 15] 134, 23, 230

→ Inversion Count

→

→

Doubts.