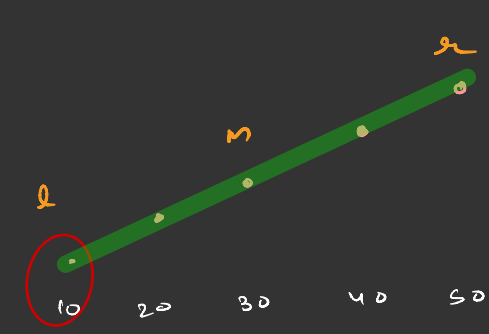
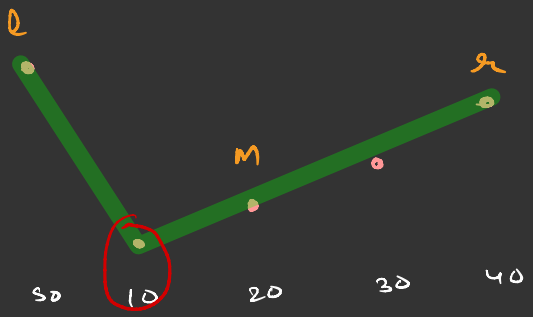
Ques Find pivot in rotated sorted array

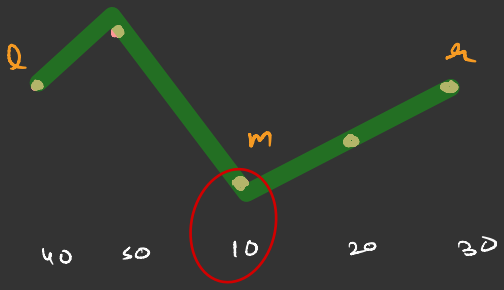
↳ index of smallest element

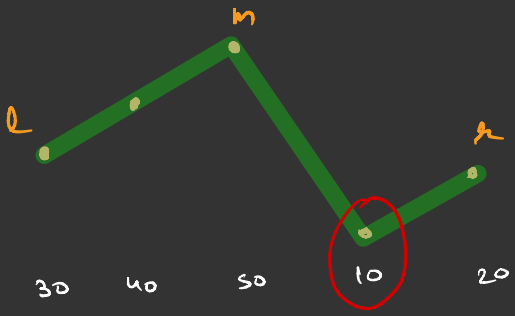
0	1	2	3	4	
30	40	50	10	20	⇒ 3

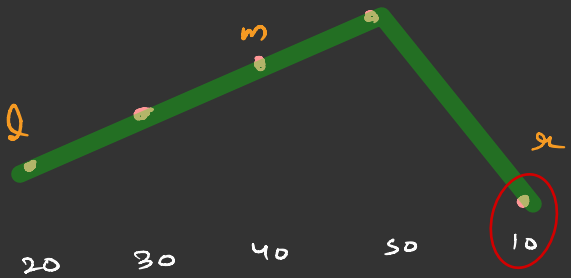
Brute Force → Linear Search O(n)











l - m

m+1 r

0	1	2	3	4
30	40	50	10	20

l
r
m

Ques Find element in rotated sorted array

	0	1	2	3	4	
40	30	40	50	10	20	⇒ 1

p1

p1 == 0

0 to size - 1

else if (target >= arr[e]
binary search (0, p-1)
else binary search (p1, r)

Q Aggressive Cows

n=5 | 10 | 1 | 2 | 7 | 5 | R=3 ⇒ 4

(1) 1 2 5 3
(1) 1 2 7 5
(1) 1 2 10 8
4 1 5 7 (2)
(4) 1 5 10 5
6 1 7 10 (3)
3 2 5 7 (2)
(3) 2 5 10 5
5 2 7 10 (3)
(2) 5 7 10 3

(1) Brute Force

Find all the combinations — n!

find max of min distance in all the combinations — R

TC → O(n! × R)

(2) min dist = 1
max distance ⇒ max − min

my ans will be from 1 to 9

Linearly move from max to min

TC ⇒ O(n(max − min)) + O(n log n)

```java
public static boolean canPlaceCows(int stalls[], int k, int minDist) {
    int lastPlacedCow = stalls[0];
    int cowsPlaced = 1;
    for(int i = 1; i < stalls.length; i++) {
        if(stalls[i] - lastPlacedCow >= minDist) {
            cowsPlaced++;
            lastPlacedCow = stalls[i];
        }

        if(cowsPlaced == k) return true;
    }
    return false;
}

public static int solve(int n, int k, int[] stalls) {
    Arrays.sort(stalls);
    int l = 1, r = stalls[stalls.length - 1] - stalls[0];
    int ans = 1;
    while(l <= r) {
        int m = l + (r-l)/2;
        if(canPlaceCows(stalls, k, m)) {
            ans = m;
            l = m + 1;
        } else r = m - 1;
    }
    return ans;

}
```

l = 1 ~~4~~

r = ~~9~~ 4

m = ~~3~~ 4

R = 3

mind = 2

C = ~~1~~ ~~2~~ 3

ans = ~~2~~ ~~3~~ 4

TC ⇒ O(log(max − min) × n) + O(n log n)