⇒ Sorted rotated array

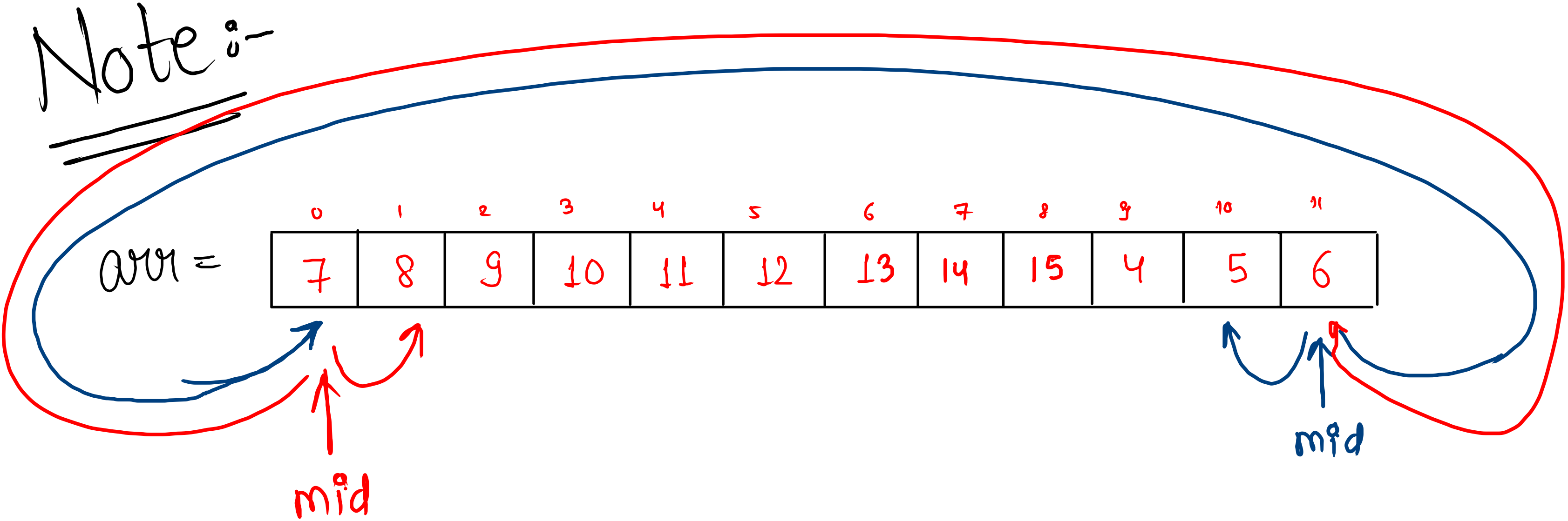
arr =

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

, ans = 8

```
int si = 0, ei = n-1;
while( si<=ei) {
    int mid = (si+ei)/2;
    if ( arr[mid] < arr[mid-1] &&
            arr[mid] < arr[mid+1] ) {
        return mid-1;
    } else if ( arr[mid] < arr[ei]) {
        ei = mid-1;
    } else if( arr[mid] > arr[si] ) {
        si = mid+1
    }
}
return -1;
```

Note :-



arr =

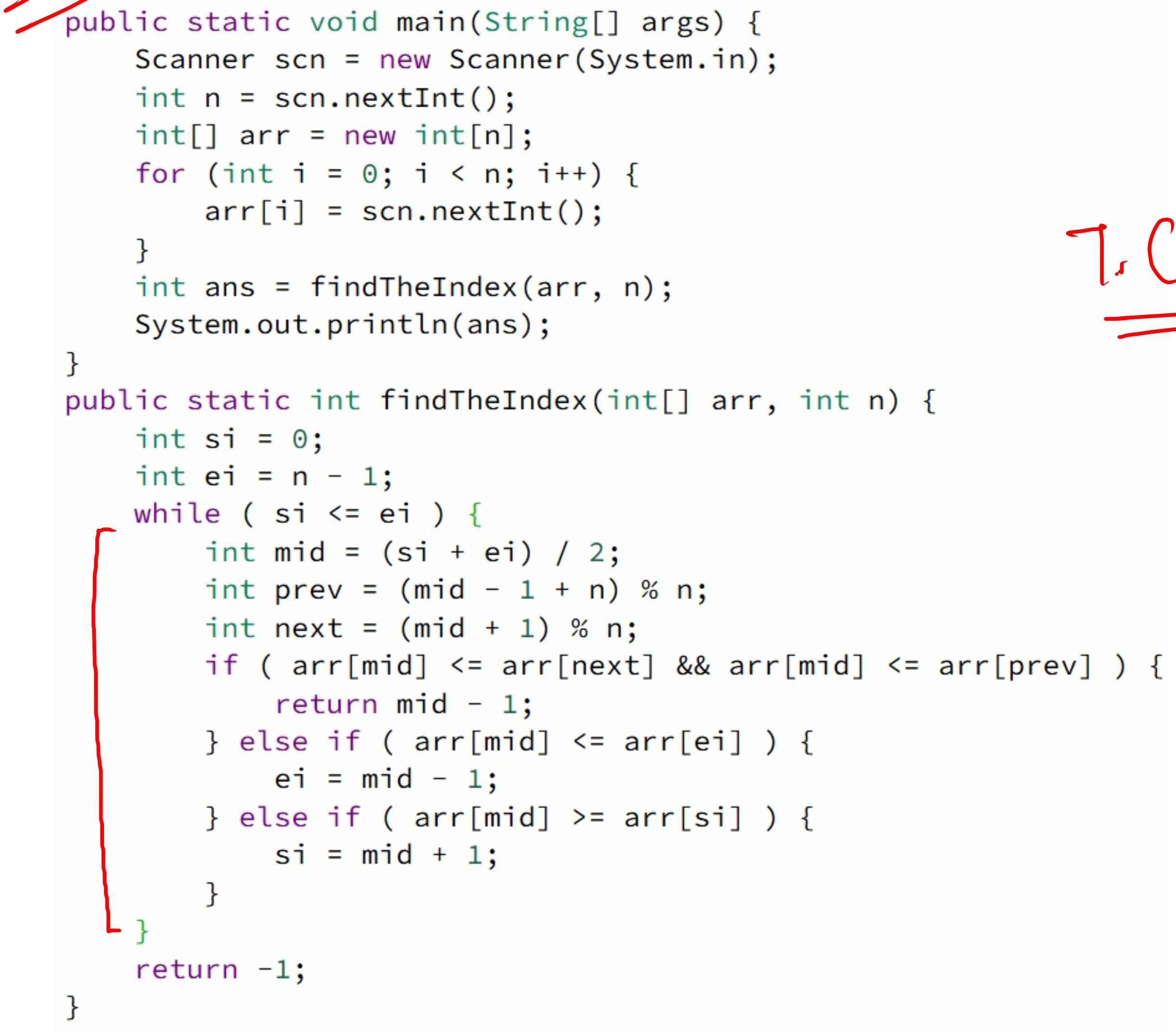
0	1	2	3	4	5	6	7	8	9	10	11
7	8	9	10	11	12	13	14	15	4	5	6

$1 \% 3 = 1$
$2 \% 3 = 2$
$3 \% 3 = 0$
$4 \% 3 = 1$
$5 \% 3 = 2$
$6 \% 3 = 0$
$7 \% 3 = 1$
$8 \% 3 = 2$
$9 \% 3 = 0$
$10 \% 3 = 1$
$11 \% 3 = 2$

clockwise rotation = $(idx) \% n$

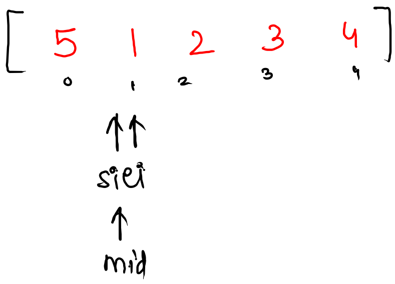
anti-clockwise rotation = $(idx + n) \% n$

code



```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    int ans = findTheIndex(arr, n);
    System.out.println(ans);
}
public static int findTheIndex(int[] arr, int n) {
    int si = 0;
    int ei = n - 1;
    while ( si <= ei ) {
        int mid = (si + ei) / 2;
        int prev = (mid - 1 + n) % n;
        int next = (mid + 1) % n;
        if ( arr[mid] <= arr[next] && arr[mid] <= arr[prev] ) {
            return mid - 1;
        } else if ( arr[mid] <= arr[ei] ) {
            ei = mid - 1;
        } else if ( arr[mid] >= arr[si] ) {
            si = mid + 1;
        }
    }
    return -1;
}
```

T.C = O(log(n))



The banana challenge (koko eating banans)

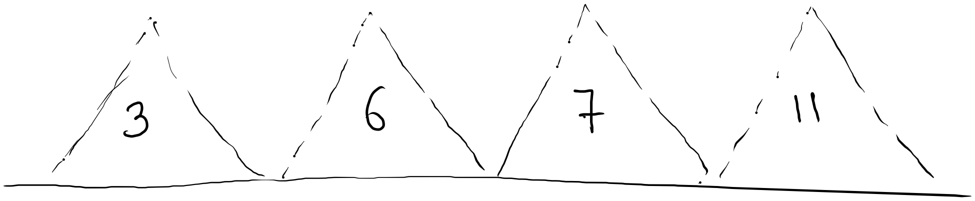
(V.V. imp)

$n = 4$

arr =

0	1	2	3
3	6	7	11

, $h = 8$



K = speed of eating banans / per hour

imp

- → we have only h hours
- → n piles of banana's each with arr[i] banana's
- ★ → koko prefers to eat slowly
- ★ → if we finish all banana's before time, then we will not pick the next pile.

assume

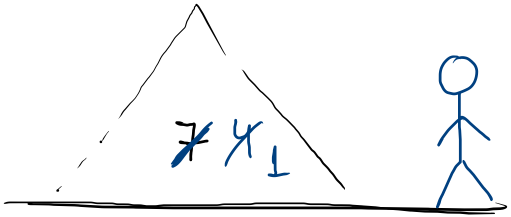
si = mini speed of eating banana's = 1

ei = max speed of eating banana's = 11 ⇒ max(arr)

	0	1	2	3
arr =	3	6	7	11
	1 hour	2 hours	3 hour	4 hour

, k = 3

total = 10 hr



1 hour
1 hour
1 hour
} = 3

K = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

ei si

mid

arr =

0	1	2	3
3	6	7	11

,

mid = speed / hour

mid = 6 → 1 + 1 + 2 + 2 = 6 hours

$\boxed{\text{mid} = 3}$ → 1 + 2 + 3 + 4 = 10 hours

mid = 4 → 1 + 2 + 2 + 3 = 8 hours

$\Rightarrow \frac{3}{3} = \boxed{1}$

$\boxed{\Rightarrow 3 \% 3 == 0}$

$6/3 = 2$

$\boxed{6 \% 3 == 0}$

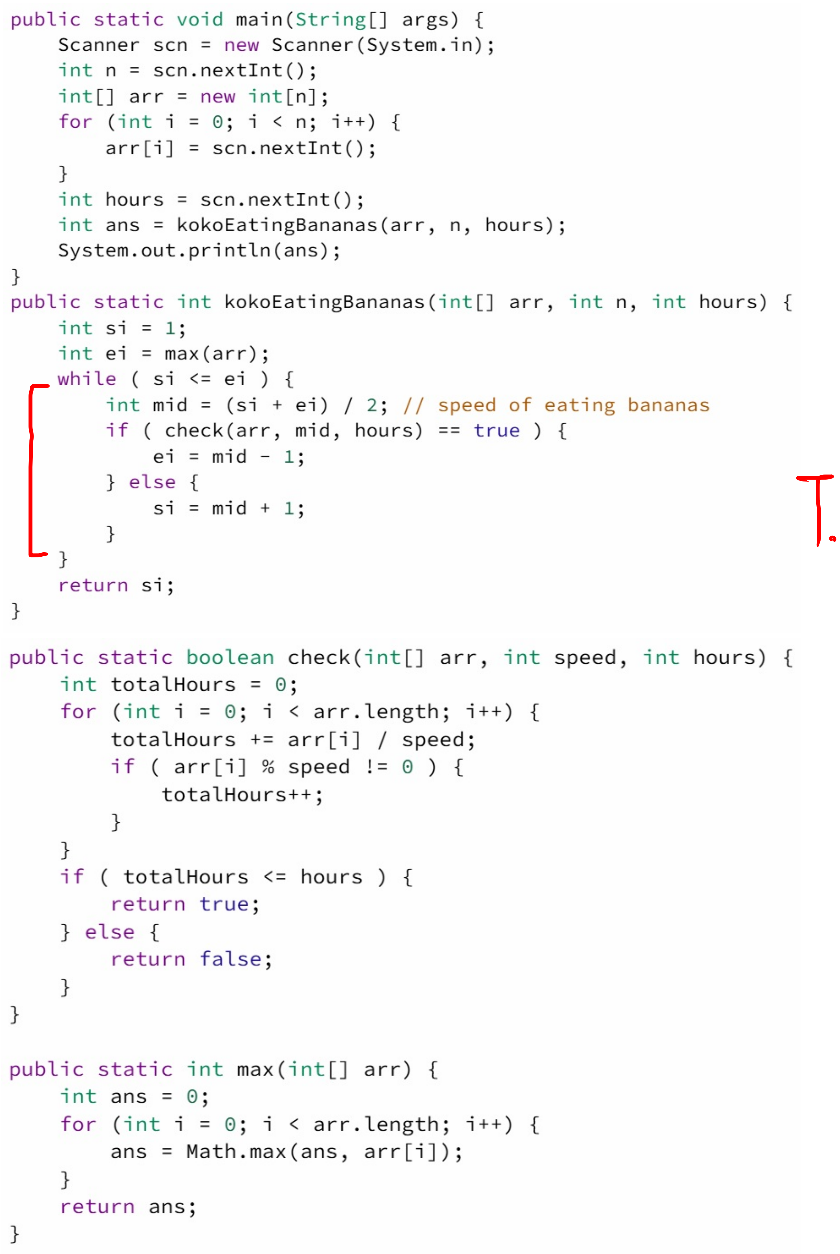
$7/3 = \boxed{2+1}$

$7 \% 3 = 1$

$11/3 = \boxed{3+1}$

$11 \% 3 = 2$

code



```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    int hours = scn.nextInt();
    int ans = kokoEatingBananas(arr, n, hours);
    System.out.println(ans);
}
public static int kokoEatingBananas(int[] arr, int n, int hours) {
    int si = 1;
    int ei = max(arr);
    while ( si <= ei ) {
        int mid = (si + ei) / 2; // speed of eating bananas
        if ( check(arr, mid, hours) == true ) {
            ei = mid - 1;
        } else {
            si = mid + 1;
        }
    }
    return si;
}

public static boolean check(int[] arr, int speed, int hours) {
    int totalHours = 0;
    for (int i = 0; i < arr.length; i++) {
        totalHours += arr[i] / speed;
        if ( arr[i] % speed != 0 ) {
            totalHours++;
        }
    }
    if ( totalHours <= hours ) {
        return true;
    } else {
        return false;
    }
}

public static int max(int[] arr) {
    int ans = 0;
    for (int i = 0; i < arr.length; i++) {
        ans = Math.max(ans, arr[i]);
    }
    return ans;
}
```

$x =$ max of array

$n + \log(\underline{\underline{x}}) * n$

$T.C = O(n + n\log x)$

$\simeq O(n\log(x))$