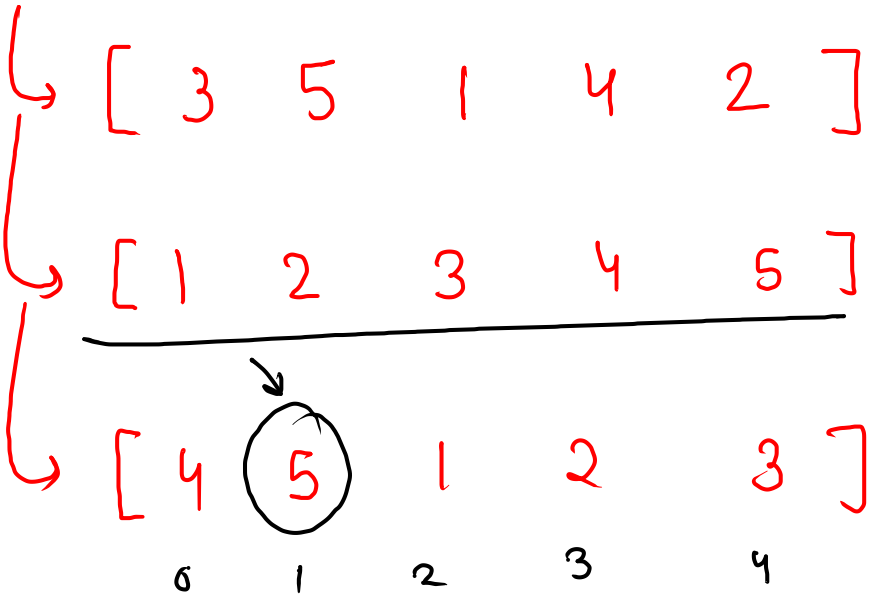
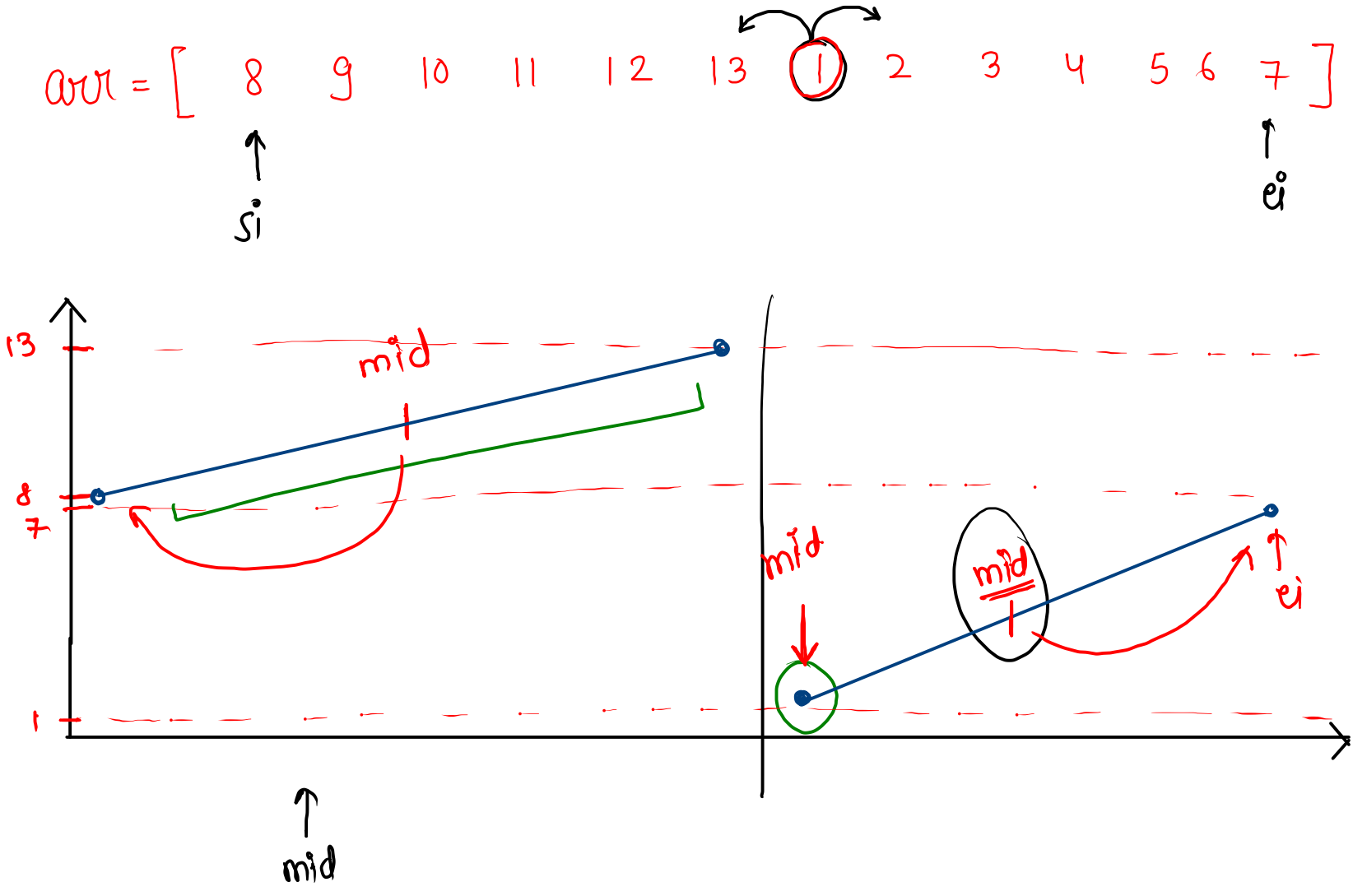
Find The Index of Rotation



using B.S

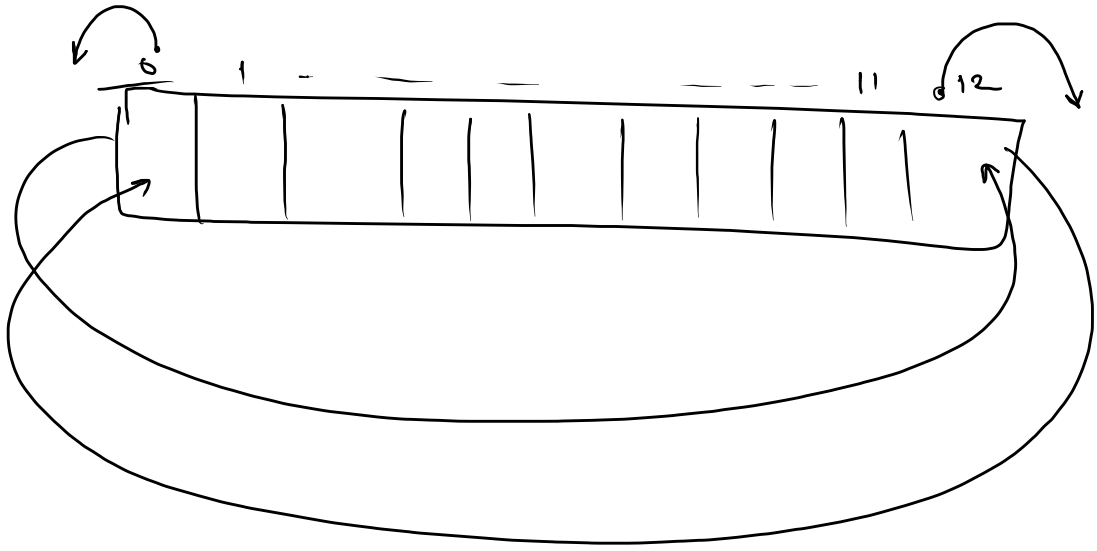
O(log N)


arr = [8 9 10 11 12 13 1 2 3 4 5 6 7]
si
ei
13
8
7
1
mid
mid
mid
mid
ei
mid

psudo code

curr = arr[mid]

```
while ( si<=ei ) {
    if ( curr <=arr[prev] && curr <=arr[next]) {
        return mid-1;
    } else if ( curr <= arr[ei]) {
        ei = mid-1;
    } else if ( curr >= arr[si] ) {
        si = mid+1;
    }
}
```



$n = 13$

$mid = 12, \quad mid+1 = 13 \,\%\, n = 13 \,\%\, 13 = 0$

$mid = 0, \quad mid-1 = ((-1) + n) \,\%\, n$

$= ((-1) + 13) \,\%\, 13$

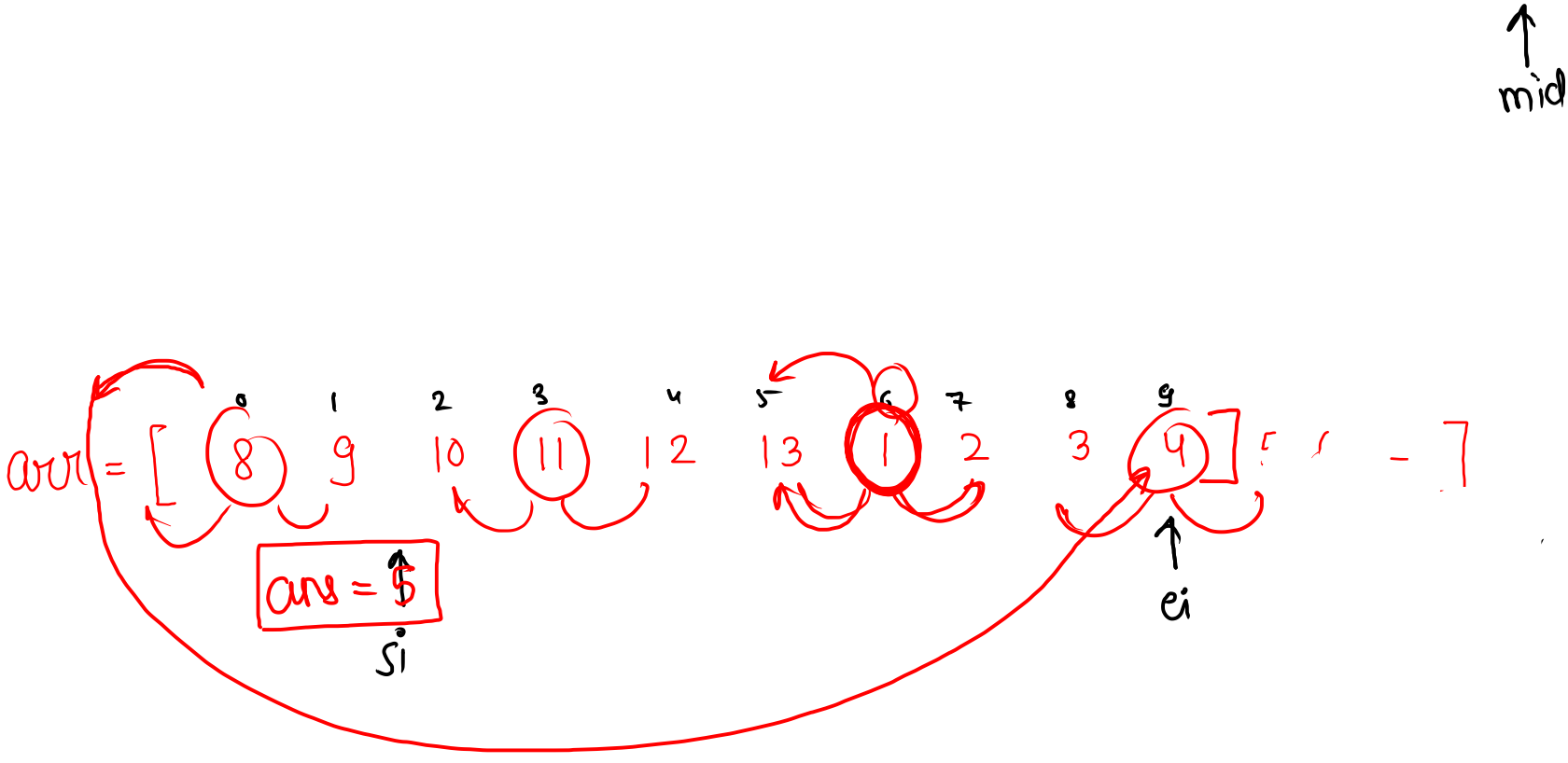
$= 12 \,\%\, 13$

$= 12$

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();
    }
    System.out.println(findIndex(arr, n));
}

public static int findIndex(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[prev] && arr[mid] <= arr[next] ) {
            return mid - 1;
        } else if ( arr[mid] <= arr[ei] ) {
            ei = mid - 1;
        } else if ( arr[mid] >= arr[si] ) {
            si = mid + 1;
        }
    }
    return -1;
}
```


mid
arr = [8 9 10 11 12 13 1 2 3 4 5 - 7
0 1 2 3 4 5 6 7 8 9
ans = 5
si
ei

The banana challenge

- we have 'h' hours to eat all bananas
- 'n' groups of banana's are there, each with piles[i] banana's
- find eating speed of koko

mid = 1

n = 4

h = 8

arr

0	1	2	3
3	6	7	11

finding range of speed (K)

speed of eating banana's

si = 1, ei = max(arr[i])

```
while (si<=ei) {
    int mid = (si+ei)/2;   //speed
    if ( check if koko able to eat all Banana's == true){
        ei = mid-1;
    } else {
        si = mid+1;
    }
}
```

n = 4

h = 8

	0	1	2	3
arr	3	6	7	11

k = 4 // speed

checks()

			arr	hours
0th index =	3/4 = 0	3%4 != 0	1	1
1st index =	6/4 = 1	6%4 != 0	1	2
2nd index =	7/4 = 1	7%4 != 0	1	2
3rd index =	11/4 = 2	11%4 != 0	1	3
				8 ←

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 totalTime = scn.nextInt();
    System.out.println(kokoEatingBananas(n, arr, totalTime));
}
public static int kokoEatingBananas(int n, int[] arr, int totalTime) {
    int si = 1;
    int ei = max(arr);
    while ( si <= ei ) {
        int mid = (si + ei) / 2;    // speed
        if ( check(arr, mid, totalTime) == true ) {
            ei = mid - 1;
        } else {
            si = mid + 1;
        }
    }
    return si;
}
```

O(logN)

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

public static int max(int[] arr) {
    int ans = -1;
    for (int i = 0; i < arr.length; i++) {
        ans = Math.max(ans, arr[i]);
    }
    return ans;
}
```

O(N) O(N)

N = max of arr

T.C $O((\log N * N) + N)$

T.C $= O(N + N \log N) = O(N * (1 + \log N))$

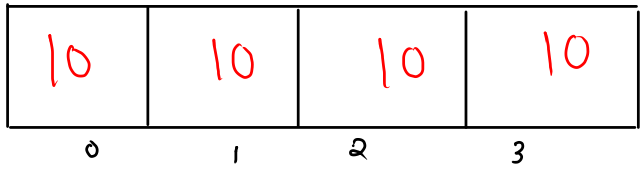
$\cong O(N \log N)$

The painter

si = max(arr)

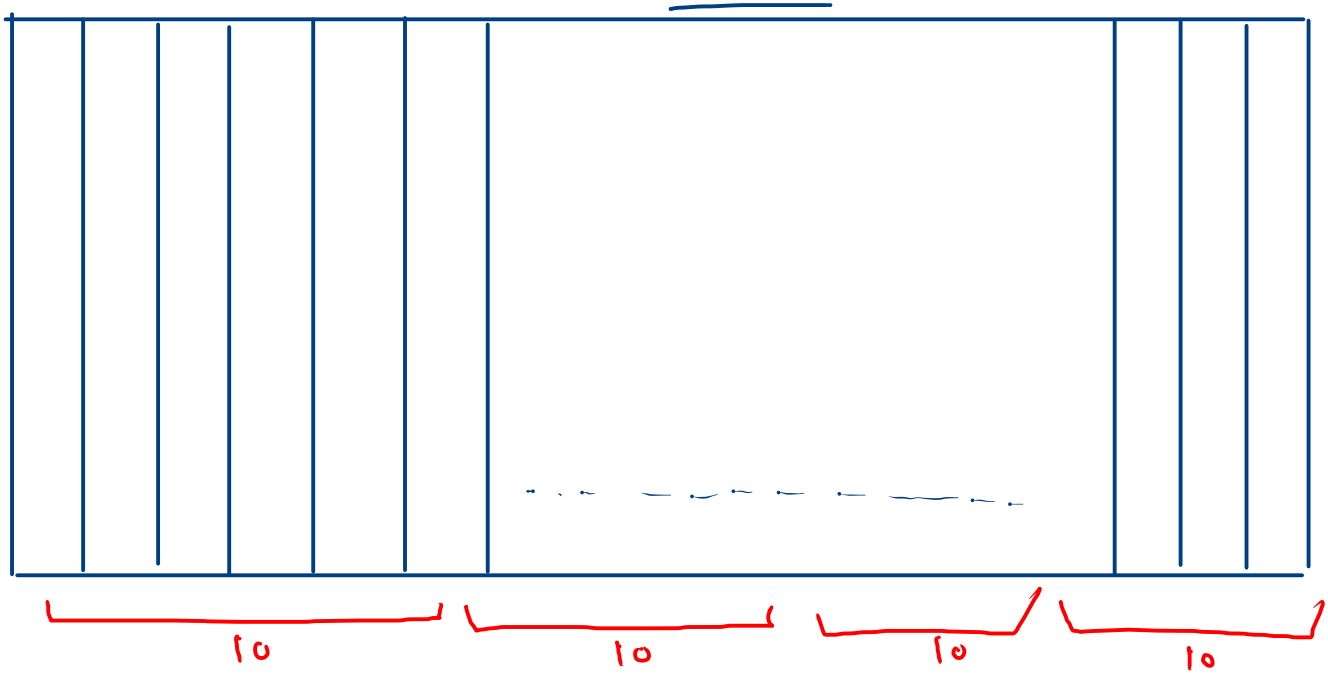
ei = sum(arr)

n = 4

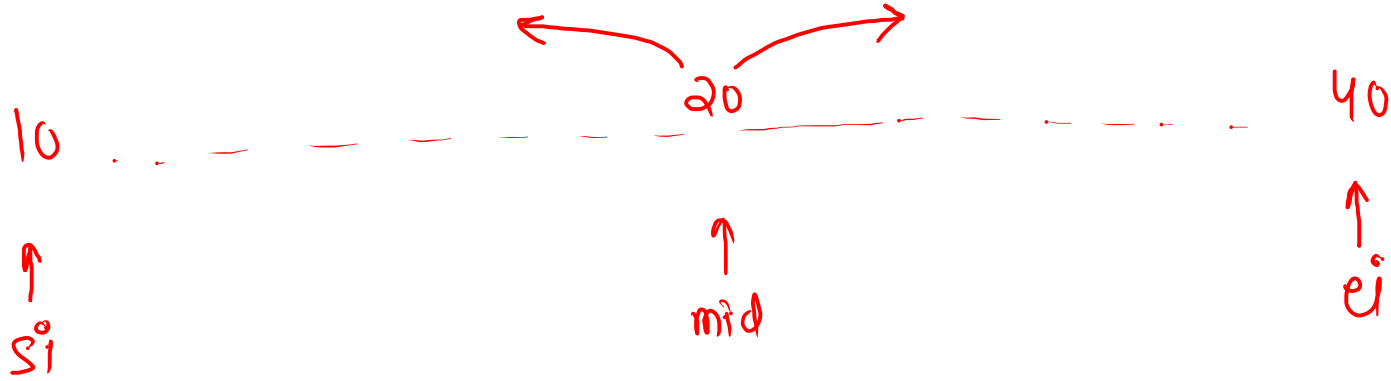


painters = 2

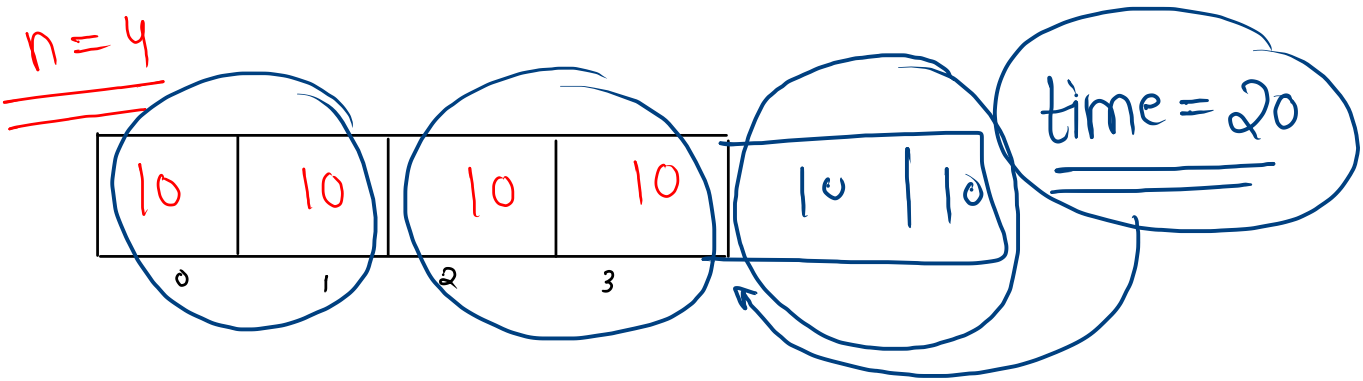
40 boards



mid = time



n = 4



painters = 2

p = 3