Find the index of Rotation

n = 5

arr = [5, 1, 2, 3, 4]

↳ If it is sorted then it would be like this → [1, 2, 3, 4, 5]

Rotated index → 0

Explanation

Ex1: [5, 1, 2, 3, 4]
0 1 2 3 4

arr[0] > arr[1]

Ex2: [4, 6, 8, 1, 2, 3]
0 1 2 3 4 5

Output: -2.

Solution:-

(i.) We have determine where the value on its right is smaller.

[5, 1, 2, 3, 4]
0 1 2 3 4

left right

mid = 2

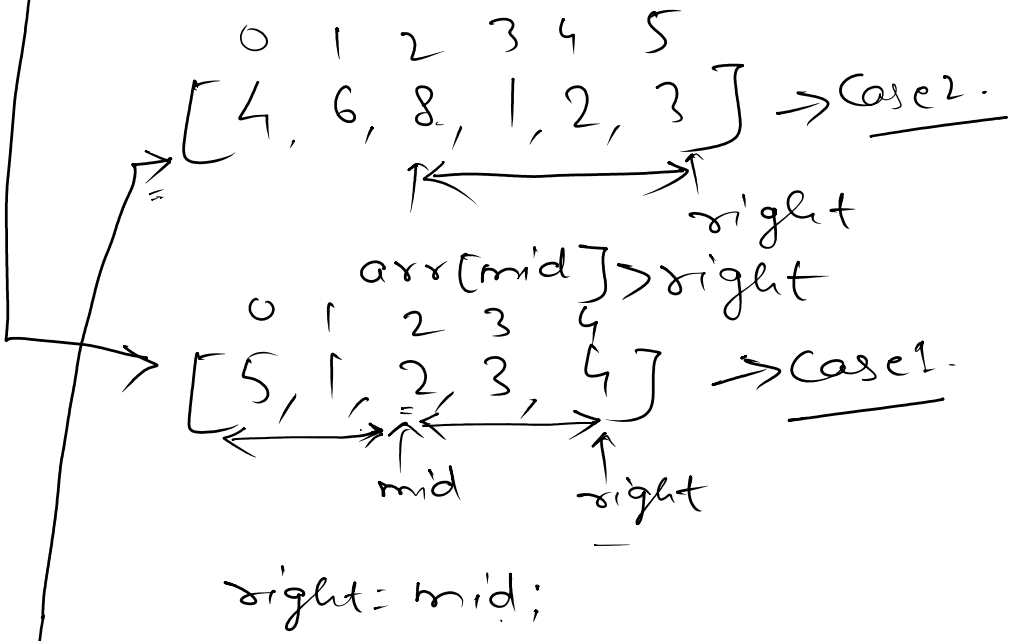
arr[2] = 2.

arr[right] = 4

arr[right] = 4

Case1. arr[right] > arr[mid]

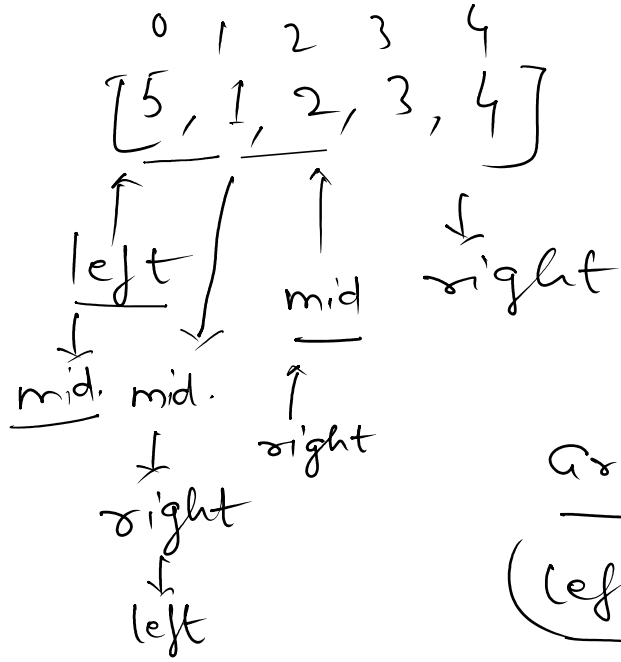
We have to check in left, because right side, it will sorted.



right = mid;

Case2:- arr[mid] > arr[right]

left = mid + 1;



arr[mid] > arr[right]

(left < right)

0 1 2 3 4 5

0 1 2 3 4 5

[4, 5, 6, 1, 2, 3]

Dry Run

1. left = 0
   right = 5
   mid = 2
   arr[mid] > arr[right]
   6 > 3
   left = mid

2. left = 2
   right = 5
   mid = 3

   0 1 2 3 4 5
   [4, 5, 6, 1, 2, 3]

   arr[mid] < arr[right]
   1 < 3.
   right = 3

3. left = 2
   right = 3
   mid = 2
   arr[mid] > arr[right]
   6 > 1.

Code.

```
int left=0, right=n-1;

while (left < right) {
  int mid=(left + right) /2 ;
  if (arr[mid] < arr[right]) {
      right=mid;
  }else {
      left=mid + 1;
    }
}
S.o.pln(left-1);
```

```
        0  1  2  3  4  5
arr = [ 6, 7, 1, 2, 3, 4 ]
```

Output -> 1

Dry Run_

Dry Run

1. left = 0,
   right = 5
   mid = 2
   arr[mid] = 1 < arr[right]
   right = mid = 2

0 1 2 3 4 5
[6, 7, 1, 2, 3, 4]

2. left = 0
   right = 2
   mid = 1
   arr[mid] = 7 > arr[right]
   left = 2.
   right = 2
   (left < right) X

S.o.pln(left - 1);

The banana Challenge

n = 4. 0 1 2 3.

arr = [3, 6, 7, 11]

h = 8.

Output → 4.

Let's assume K = 3

1 → 1st hour → 1 pile → 3 → all will be eaten

2nd hour → 2nd Pile → 6 → 3 banana can be eaten

(T) 3rd hour → 3rd Pile → 3 banana from 2nd Pile

4th hour → 4th Pile → 3 from 3rd

3rd → 4

4th → 11

5th → 3, remaining → 12

6th → 3, rem → 9

7th → 3, rem → 6

8th → 3, rem → 3

9th → 3, rem → 0

hours : 9.

That's why, speed can not be 3/hour.

[3, 6, 7, 11]

K = 4

Hour Pile Remaining

1	1st	0
2	2	2
3	2	0 (2 eaten)
4	3	3
5	3	0
6	4	7
7	4	3
8	4	0

[3, 6, 7, 11]

[3, 6, 7, 11]

K = 4

3/4 = 0.75 = 1

6/4 = 1.5 = 2

7/4 = 1.75 = 2

11/4 = 2.75 = 3

} (8)

To determine, speed we need to find max and min speed

maxspeed :- No. of bananas available in largest pile.

3, 6, 7, 11

11 bananas / hour

minspeed :- 1 banana / hour

Speed will lie between 1 and 11.

Two Approach :-

Two Approach :-

1. Using for loop, checking will every speed between 1 and 11.

i = 1, i <= 11

for any value of i, we have to calculate min hour to finish eating all bananas

if min hour to finish all banana is less than equal to given hour, then that value of i, which is speed will be my answer.

2nd Approach.

left = 1, right = 11 (6)

mid = (left + right) / 2

↓

If with speed = mid, no. of hours taken to finish banana <= h (8), then

with K = 6,

↓

1. total hours taken <= 8

2. total hour taken > 8.

1st posibility total hours <= 8

K = 6.

ans = 6.

if any speed < k., totalhours <= 8.
ans = that speed

2nd Possibility

totalhours > 8

if this case, we should increase speed
so, we should take values in right of mid

```
public static int hoursTaken(int piles[], int k){
    int totalhour=0
   for(int i=0; i<piles.length; i++){
      int hour = Math.ceil((double) piles[i]/k);
      totalhour+=hour;
    }
    return totalhour;
}

p.s.v.m(String[] args){
 int h= sc.nextInt();
 int left=1;
 int max=Integer.MIN_VALUE;
 for(int i=0; i<piles.length;i++){
    max= Math.max(max piles[i]);
```

```
for(int i=0; i<piles.length; i++){
    max = Math.max(max, piles[i]);
}
int right = max; int ans = -1;
while(left <= right){
    int mid = (left + right)/2
    if( hoursTaken(piles, mid) <= h){
        ans = mid;
        right = mid-1;
    }else{
        left = mid+1;
    }
}
S.o.pln(ans);
}
```