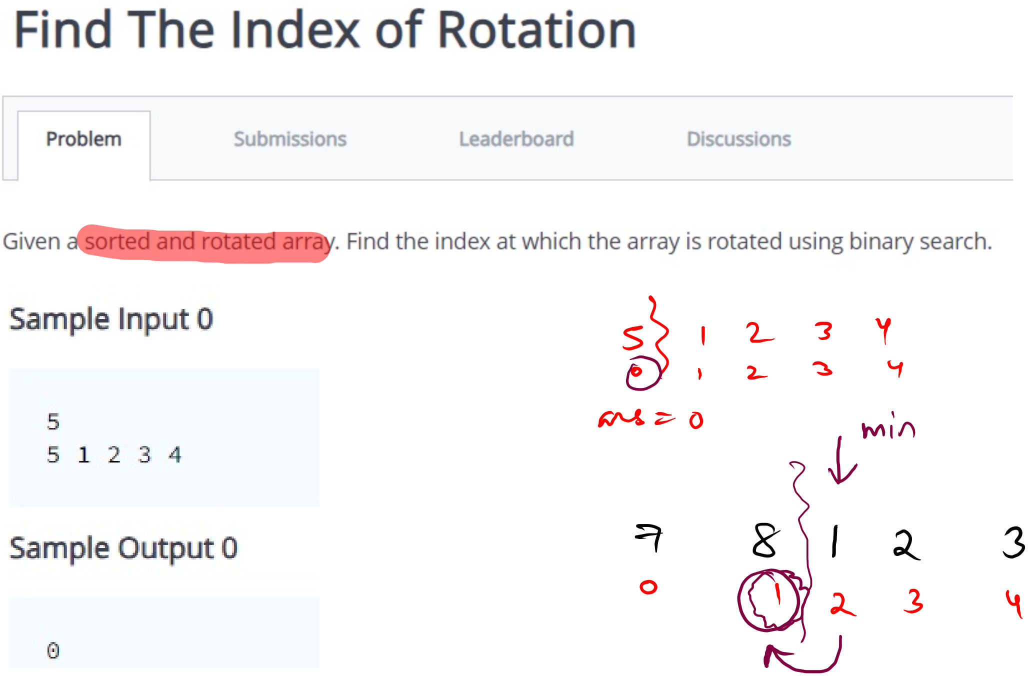


Find The Index of Rotation

Given a sorted and rotated array. Find the index at which the array is rotated using binary search.

Sample Input 0

```
5
5 1 2 3 4
```

Sample Output 0

```
0
```

→ sorted

5 1 2 3 4

0 1 2 3 4

ans = 0

min

7 8 1 2 3 4 5 6

0 1 2 3 4 5 6 7

50 60 70 10 20 30 40

0 1 2 3 4 5 6

point of rotation

max ele in s.r. array.

min " " " "

Same.

3 4 6 7 8 1 2

K = 5

position

7	8	1	2	3	4	5	6	pos = 1
0	1	2	3	4	5	6	7	

min → 1

idx = 2

⇒ idx - 1 } idx of rotation

max

idx of
min ele

brute force ? → linear search. O(n)

7	8	1	[~~2~~	~~3~~	~~4~~	~~5~~	~~6~~]
0	1	2	3	4	5	6	7
l		r	m				

binary search. O(log n).

m = 0+7/2 = 3

A[m] <= A[r]

sorted → eliminate.

4	5	[1	2	3]
0	1	2	3	4
l		m		r

x

$l \le r$

			↓				
[7	8]	1	2	3	4	5	6
0	1	2	3	4	5	6	7
	r						

sorted → elimate.

l

m

✓ $A[m] \le A[r]$

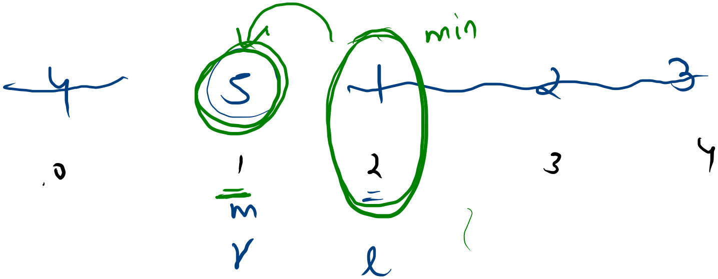
A[l

idx = ~~3~~ 2

A[3]

$1 < 2$

$l \le r$



$idx = 2$

$5 < 1$

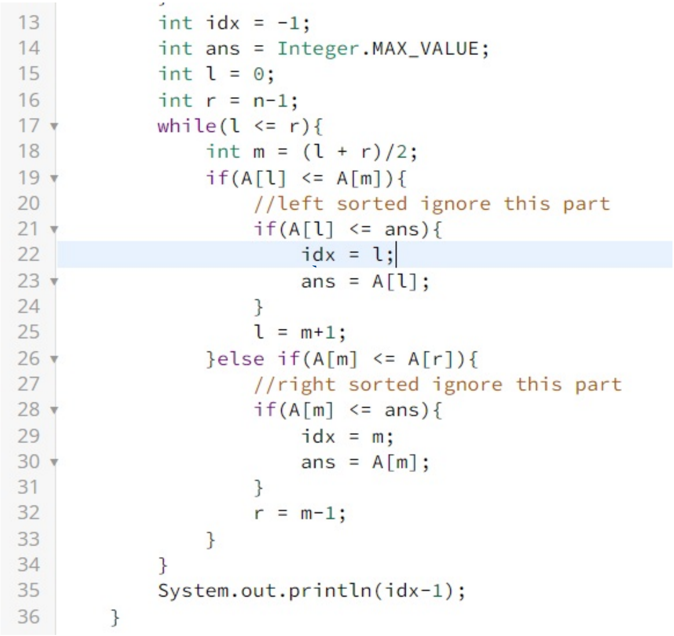
$A[l] \le A[m]$ ✓

$A[m] \le A[r]$

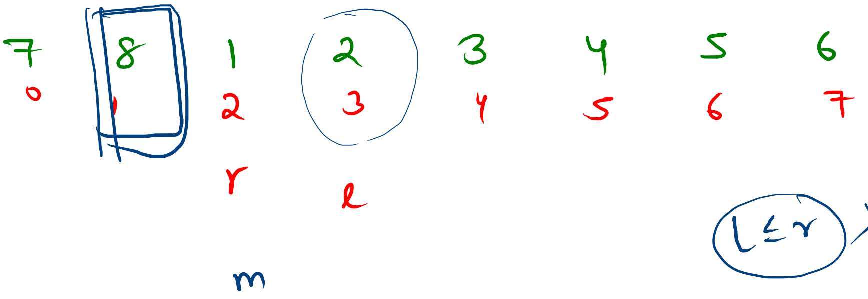
$2-1$ → idx of rotation

ans = ~~∞~~ ~~7~~ 1

idx = ~~-1~~ ~~3~~ 2



```
        int idx = -1;
        int ans = Integer.MAX_VALUE;
        int l = 0;
        int r = n-1;
        while(l <= r){
            int m = (l + r)/2;
            if(A[l] <= A[m]){
                //left sorted ignore this part
                if(A[l] <= ans){
                    idx = l;
                    ans = A[l];
                }
                l = m+1;
            }else if(A[m] <= A[r]){
                //right sorted ignore this part
                if(A[m] <= ans){
                    idx = m;
                    ans = A[m];
                }
                r = m-1;
            }
        }
        System.out.println(idx-1);
    }
```



$l \le r$ ✗

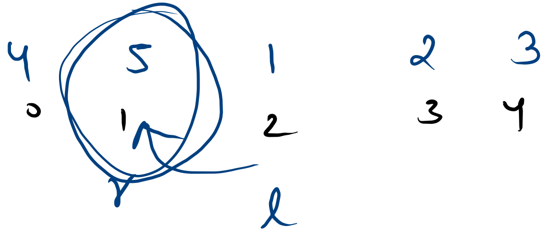
$1 \le 1$ ✓

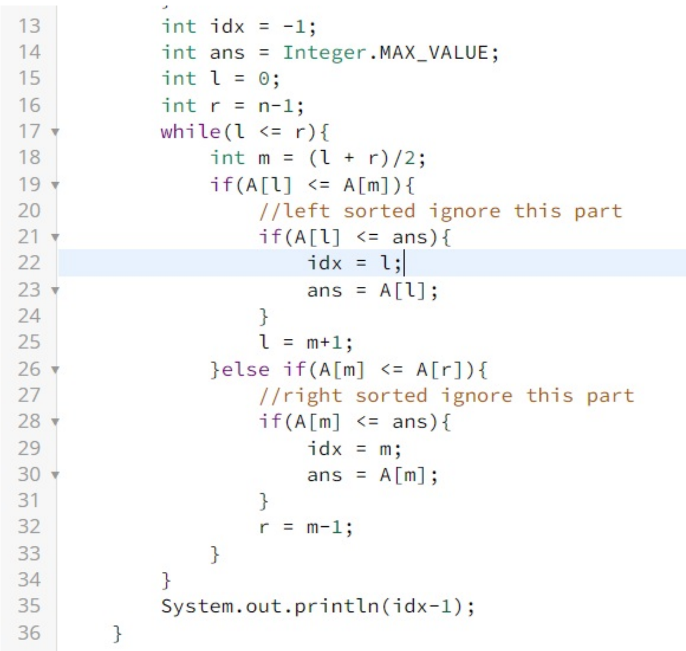
$1 \le 2$ ✓

$2-1 = 1$

idx = ~~-1~~ 2

ans = ~~6~~ 1





```java
        int idx = -1;
        int ans = Integer.MAX_VALUE;
        int l = 0;
        int r = n-1;
        while(l <= r){
            int m = (l + r)/2;
            if(A[l] <= A[m]){
                //left sorted ignore this part
                if(A[l] <= ans){
                    idx = l;
                    ans = A[l];
                }
                l = m+1;
            }else if(A[m] <= A[r]){
                //right sorted ignore this part
                if(A[m] <= ans){
                    idx = m;
                    ans = A[m];
                }
                r = m-1;
            }
        }
        System.out.println(idx-1);
    }
```

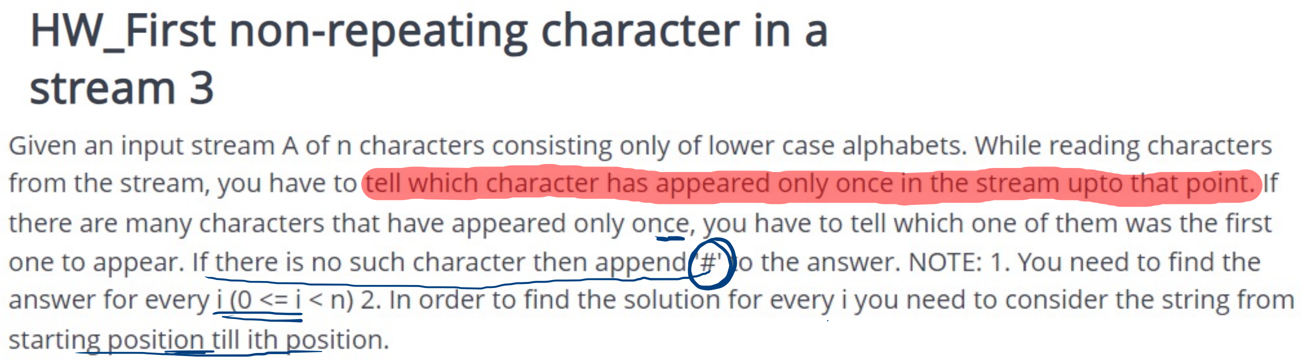
l ≤ r

$\boxed{2 \le 1}$ ✗

m

$\boxed{5 \le 5}$ ✓

$\boxed{5 \le 1}$ ✗



HW_First non-repeating character in a stream 3

Given an input stream A of n characters consisting only of lower case alphabets. While reading characters from the stream, you have to tell which character has appeared only once in the stream upto that point. If there are many characters that have appeared only once, you have to tell which one of them was the first one to appear. If there is no such character then append '#' to the answer. NOTE: 1. You need to find the answer for every i (0 <= i < n) 2. In order to find the solution for every i you need to consider the string from starting position till ith position.

"a"
a

"ab"
aa

" aaaa"
a###

" aabaa"
a#bbb

" aabcaa"
a#bbbb

[1, N]

n = 5

100 101 110 111

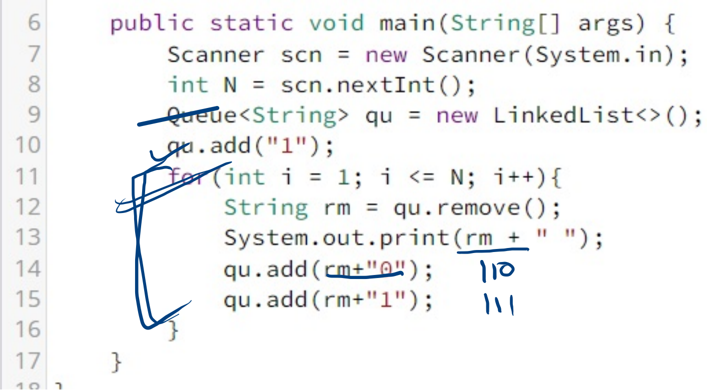
1

10

11 0 1

100

101



```
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int N = scn.nextInt();
        Queue<String> qu = new LinkedList<>();
        qu.add("1");
        for(int i = 1; i <= N; i++){
            String rm = qu.remove();
            System.out.print(rm + " ");
            qu.add(rm+"0");
            qu.add(rm+"1");
        }
    }
}
```

110

111

1

10

11