https://course.acciojob.com/idle?question=76b8307d-355f-4602-9556-bbda20b47fc7

- MEDIUM
- Max Score: 40 Points

Search For A Range

Given a sorted array of integers A(0 based index) of size N, find the starting and ending position of a given integar B in array A. Print 2 integers separated by space, such that first element = starting position of B in A and second element = ending position of B in A, if B is not found in A print -1 -1.

Your algorithm's runtime complexity must be in the order of O(log n).

Input Format

The first line given is `N`, denoting the size of array `A`.

The second line given is the integer array `A`.

The third line given is the integer `B`.

Output Format

Print 2 integers, such that first element = starting position of `B` in `A` and second element = ending position of `B` in `A`, if `B` is not found in `A` print -1 -1.

Example 1

Input

```
6
5 7 7 8 8 10
8
```

Output

```
3 4
```

Explanation

First occurence of 8 in A is at index 3

Second occurence of 8 in A is at index 4

`ans` = [3, 4]

Example 2

Input

```
4
5 17 100 111
3
```

Output

```
-1 -1
```

Explanation

3 does not exist in the array

Constraints

`1 <= N <= 10^6`

`1 <= A[i], B <= 10^9`

Topic Tags

- **Binary Search**
- **Arrays**

My code

```
// n java

import java.util.*;
import java.lang.*;
import java.io.*;

public class Main {
    public static void main(String[] args) throws java.lang.Exception {

        Scanner s = new Scanner(System.in);
        int n = s.nextInt();

        int c = -1, d = -1;
        int arr[] = new int[n];
        for (int i = 0; i < n; i++)
            arr[i] = s.nextInt();
   int k = s.nextInt();
        int lp = 0, rp = n - 1, mid = 0, flag = 0;
        while (lp <= rp) {
            mid = (lp + rp) / 2;
            if (arr[mid] < k) {
                lp = mid + 1;
            } else {
                if (arr[mid] == k)
                    c = mid;
                rp = mid - 1;
            }
```

```
        }
        if (c == -1) {
            System.out.print("-1 -1");
            return;
        }
        lp = 0;
        rp = n - 1;
        mid = 0;
        while (lp <= rp) {
            mid = (lp + rp) / 2;
            if (arr[mid] > k) {
                rp = mid - 1;
            } else {
                if (arr[mid] == k)
                    d = mid;
                lp = mid + 1;
            }
        }
        System.out.println(c + " " + d);
    }

}
```