根据某条件聚类最少交换次数题目描述：

给出数字 K,请输出所有结果小于 K 的整数组合到一起的最少交换次数。

组合一起是指满足条件的数字相邻，不要求相邻后在数组中的位置。

数据范围

-100 <= K <= 100

-100 <= 数组中数值<= 100

输入描述：

第一行输入数组：1 3 1 4 0

第二行输入 K 数值：2

输出描述：

第一行输出最少较好次数：1

补充说明：

小于 2 的表达式是 1 1 0, 共三种可能将所有符合要求数字组合一起，最少交换 1 次

示例 1

输入：

```
1 3 1 4 0

2
```

输出：

```
1
```

说明：

示例 2

输入：

```
0 0 0 1 0

2
```

输出：

```
0
```

说明：

示例 3

输入：

```
2 3 2
1
```

输出：

```
0
```

说明：

```
import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.Scanner;

import java.util.concurrent.atomic.AtomicInteger;

import java.util.stream.Collectors;

//1 3 1 4 0

// 注意类名必须为 Main, 不要有任何 package xxx 信息

public class Main {

    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
        // 注意 hasNext 和 hasNextLine 的区别
        while (scanner.hasNextInt()) { // 注意 while 处理多个 case
            List<Integer> nums = Arrays.stream(scanner.nextLine().split("
")).map(Integer::parseInt).collect(Collectors.toList());
            int k = Integer.parseInt(scanner.nextLine());
            AtomicInteger result = new AtomicInteger(Integer.MAX_VALUE);
            fun(nums,k,result);
            System.out.println(result.get());
        }
    }
    public static void fun(List<Integer> nums, int k, AtomicInteger result){
        int len = 0;
        for (int i = 0; i < nums.size(); i++) {
            Integer num = nums.get(i);
            if (num < k){
                len++;
            }
        }
        if (len == 1){
            result.set(0);
            return;
```

```
}
int left = 0;
int right = len - 1;

while (left < nums.size() && right < nums.size()){
    int count = 0;
    List<Integer> tempNums = new ArrayList<>(nums);
    for (int i = left; i <= right; i++) {
        if (nums.get(i) >= k){
            boolean swap = swap(left, right, k, i, tempNums);
            if (swap){
                count++;
            }else {
                result.set(-1);
                return;
            }
        }
    }
    result.set(Math.min(result.get(),count));
    left++;
    right++;
}
```

```
    }
    public static boolean swap(int left,int right,int k,int current,List<Integer>
nums){
        //[0,left - 1],[right + 1,nums.size)
        boolean flag = false;
        if (left != 0){
            for (int i = 0; i < left; i++) {
                if (nums.get(i) < k){
                    int tmp = nums.get(current);
                    nums.set(current,nums.get(i));
                    nums.set(i,tmp);
                    return true;
                }
            }
        }
        if (right != nums.size()){
            for (int i = right; i < nums.size(); i++) {
                if (nums.get(i) < k){
                    int tmp = nums.get(current);
                    nums.set(current,nums.get(i));
                    nums.set(i,tmp);
                    return true;
```

```
                }

            }

        }

        return false;

    }


}
```