牛客网-华为机试练习题 24

题目描述

计算最少出列多少位同学，使得剩下的同学排成合唱队形

说明：

N位同学站成一排，音乐老师要请其中的(N-K)位同学出列，使得剩下的K位同学排成合唱队形。 合唱队形是指这样的一种队形：设K位同学从左到右依次编号为1，2…，K，他们的身高分别为T1，T2，…，TK， 则他们的身高满足存在i（1<=i<=K）使得T1<T2<......<Ti-1Ti+1>......>TK。 你的任务是，已知所有N位同学的身高，计算最少需要几位同学出列，可以使得剩下的同学排成合唱队形。

输入描述:

整数N

输出描述:

最少需要几位同学出列

示例1
输入

```
8
186 186 150 200 160 130 197 200
```

输出

```
4
```

解决代码：

```java
import java.util.Arrays;
import java.util.Scanner;


public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Scanner in = new Scanner(System.in);
        String tempString = null;
        String[] tempss = null;
        int[] nums = null;
        int n = -1,result = -1;

        while(in.hasNext()){
            tempString = in.nextLine().trim();
            n = Integer.parseInt(tempString.trim());

            tempString = in.nextLine().trim();
            tempss = tempString.split(" ");
            nums = new int[tempss.length];
            for(int i = 0 ; i< tempss.length;i++)
                nums[i] = Integer.parseInt(tempss[i].trim());
            result = process(nums);
            System.out.println(n - result);
        }

    }

    private static int[] largest(int[] nums) {
        // TODO Auto-generated method stub
        int[] temp = new int[nums.length];
        int lastLoc = -1,begin = -1,end = -1,curLoc = -1;

        int[] preLen = new int[nums.length];
        Arrays.fill(preLen, 0);
        preLen[0] = 1;
        Arrays.fill(temp, -1);
        temp[0] = nums[0];
        lastLoc = 0;
        for(int i = 1;i<nums.length;i++){

            if(nums[i] > temp[lastLoc]){
                lastLoc ++;
                temp[lastLoc] = nums[i];
                preLen[i] = lastLoc+1;
                continue;
            }

            begin = 0;end = lastLoc;
            while(begin <= end){
                curLoc = (begin + end)/2;
                if(temp[curLoc] < nums[i]){
                    begin = curLoc + 1 ;
                }else if(temp[curLoc] > nums[i]){
                    end = curLoc - 1;
                }else{
                    break;
                }
            }
            preLen[i] = begin+1;
            if(temp[begin] >= nums[i])
                temp[begin] = nums[i];

        }

        return preLen;
    }

    private static int process(int[] nums){
        int[] preLen = null,postLen = null;
        preLen = largest(nums);

        int[] tempNums = new int[nums.length];
        int i = nums.length-1;

        for(int n:nums)
            tempNums[i--] = n;
//      System.out.println(Arrays.toString(nums));
//      System.out.println(Arrays.toString(tempNums));
        postLen = largest(tempNums);
        int k = 0;
        for(i = 0;i < preLen.length;i++){
            k = Math.max(preLen[i]+postLen[nums.length-1-i], k);
        }

//      System.out.println(Arrays.toString(preLen));
//      System.out.println(Arrays.toString(postLen));

        return k-1;

    }

}
```