https://course.acciojob.com/idle?question=c56cdd7f-2f80-4a1e-b996-11670f949c0e

- EASY
- Max Score: 20 Points
-
-

You are given ‘N’ cubes in an array ‘ARR’ in a certain order, and your task is to build towers using them. Whenever two cubes are on top of the other, the upper cube must be smaller than the lower cube. You must process the cubes in the given order. You can always either place the cube on top of an existing tower or begin a new tower. What is the minimum possible number of towers?

For example: Given ‘N’ = 3, ‘ARR’[] = 3, 2, 1. The answer will be one because you can stack one over two over 3. Therefore only these can be inserted in the same tower.

Input format: The first line of input contains a single integer ‘N’, where ‘N’ is the number of elements of the array. The second line of input contains ‘N’ space-separated integers, denoting the array elements.

Output format: Print a single line containing a single integer denoting the minimum possible number of towers.

Example 1: Input: 3 3 2 1 Output: 1 Explanation: The answer will be one because you can stack one over two over 3. Therefore only these can be inserted in the same tower.

Example 2: Input: 3 1 2 3 Output: 3 Explanation: The answer will be three because the array is in ascending order, and since we can only stack smaller values, three towers will have to be made.

Constraints: 1 <= N <= 2000 1 <= ARR[i] <= 2000

Topic Tags

- Binary Search

My code

```
// n java
import java.io.*;
import java.util.*;

public class Main {
	// function to sertch position of next cube
	static int sertchpossition(ArrayList<Integer> list,int k)
	{
		int lp=0;
		int rp=list.size()-1;
		int mid=0;

		while(lp<=rp)
			{
			 mid=(lp+rp)/2;
				int t=list.get(mid);
				if(t>k)
					rp=mid-1;
				else lp=lp+1;
			}
		return lp;
	}
	public static void main(String args[]) {
		// your code here
		Scanner s=new Scanner(System.in);
		int n=s.nextInt();
		ArrayList<Integer>list=new ArrayList<Integer>();
```

```
            int arr[]=new int[n];
            for(int i=0;i<n;i++)
                  arr[i]=s.nextInt();
            list.add(arr[0]);
            for(int i=1;i<n;i++)
                  {
                        int t=sertchpossition(list,arr[i]);
                        if(t==list.size())//ie next tower or grater then
size , 0 base indexing
                              list.add(arr[i]);
                        else list.set(t,arr[i]);
                  }

            System.out.print(list.size());
      }
}
```