**Longest Increasing Subsequence**

**Medium**

Given an array of integers, find the **length**of the**longest (strictly) increasing subsequence** from the given array.

**Example 1:**

**Input:**

N = 16

A = {0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15}

**Output:**6

**Explanation:**There are more than one LIS in this array. One such Longest increasing subsequence is {0,2,6,9,13,15}.

**Example 2:**

**Input:**

N = 6

A[] = {5,8,3,7,9,1}

**Output:**3

**Explanation:**There are more than one LIS in this array. One such Longest increasing subsequence is {5,7,9}.

**Expected Time Complexity** : O( N\*log(N) )  
**Expected Auxiliary Space**: O(N)

**Constraints:**  
1 ≤ N ≤ 104  
0 ≤ A[i] ≤ 106

**Company Tags**

[**Paytm**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Paytm) [**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft) [**OYO Rooms**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=OYO%20Rooms) [**Samsung**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Samsung) [**BankBazaar**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=BankBazaar)

**Topic Tags**

[**Dynamic Programming**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Dynamic%20Programming) [**Algorithms**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Algorithms)

//{ Driver Code Starts

import java.io.\*;

import java.util.\*;

class CodingMaxima{

public static void main(String args[]) throws IOException {

//taking input using Scanner class

Scanner sc = new Scanner(System.in);

//taking total testcases

int t = sc.nextInt();

while(t > 0){

//taking size of array

int n = sc.nextInt();

int array[] = new int[n];

//inserting elements in the array

for (int i = 0; i < n; ++i)

{

array[i] = sc.nextInt();

}

//creating an object of class Solution

Solution ob = new Solution();

//calling longestSubsequence() method of class

//Solution

System.out.println(ob.longestSubsequence(n,array));

t--;

}

}

}

// } Driver Code Ends

class Solution

{

//Function to find length of longest increasing subsequence.

static int longestSubsequence(int n, int arr[])

{

int[] dp=new int[n];

int max=1;

Arrays.fill(dp, 1);

for(int i=1;i<n;i++){

for(int j=0;j<i;j++){

if(arr[i]>arr[j]){

dp[i]=Math.max(dp[i], dp[j]+1);

max=Math.max(max, dp[i]);

}

}

}

return max;

}

}