# 最长上升子序列

## 1　题目描述

广场上站着一支队伍，她们是来自全国各地的扭秧歌代表队，现在有她们的身高数据，请你帮忙找出身高依次递增的子序列。 例如队伍的身高数据是（1、7、3、5、9、4、8），其中依次递增的子序列有（1、7），（1、3、5、9），（1、3、4、8）等，其中最长的长度为4。

### 1.1　输入描述:

输入包含多组数据，每组数据第一行包含一个正整数n（1≤n≤1000）。紧接着第二行包含n个正整数m（1≤n≤10000），代表队伍中每位队员的身高。

### 1.2　输出描述:

对应每一组数据，输出最长递增子序列的长度。

### 1.3　输入例子:

7

1 7 3 5 9 4 8

6

1 3 5 2 4 6

### 1.4　输出例子:

4

4

## 2　解题思路

这题目是经典的DP题目，也可叫作LIS（Longest Increasing Subsequence）最长上升子序列或者最长不下降子序列。很基础的题目，有两种算法，复杂度分别为O()和O()。

### 2.1　O()算法分析

假设a[0:n]是给定的序列（长度为n+1，下标从0到n）

1. 对于a[n]来说，由于它是最后一个数，所以当从a[n]开始查找时，只存在长度为1的不下降子序列；
2. 若从a[n-1]开始查找，则存在下面的两种可能性：
3. 若a[n-1]<a[n]则存在长度为2的不下降子序列a[n-1]，a[n]；
4. 若a[n-1]>a[n]则存在长度为1的不下降子序列a[n-1]或者a[n]。
5. 一般若从a[t]开始，此时最长不下降子序列应该是按下列方法求出的：在a[t+1]，a[t+2]，...，a[n]中，找出一个比a[t]大的且最长的不下降子序列，作为它的后继。
6. 为算法上的需要，定义二个数组：

* 记录最大升序的数组len[0:n]：len[i]表示序列为a[0:i]，并且包含a[i]元素的最长子序列的长度。
* 记录后继位置的数组next[0:n]：next[i]表示i位置的下一个元素的位置是next[i]。

### 2.2　O()算法分析

设A[t]表示序列中的第t个数，F[t]表示从0到t这一段中以t结尾的最长上升子序列的长度，初始时设F[t]=0(t=1，2，...，len(A))。则有动态规划方程：F[t]=max{1，F[j]+1}(j=1，2，...，t-1，且A[j]<A[t])。

现在，我们仔细考虑计算F[t]时的情况。假设有两个元素A[x]和A[y]，满足

1. x<y<t
2. A[x]<A[y]<A[t]
3. F[x]=F[y]

此时，选择F[x]和选择F[y]都可以得到同样的F[t]值，那么，在最长上升子序列的这个位置中，应该选择A[x]还是应该选择A[y]呢？

很明显，选择A[x]比选择A[y]要好。因为由于条件(2)，在A[x+1] ... A[t-1]这一段中，如果存在A[z]，A[x] < A[z] < a[y]，则与选择A[y]相比，将会得到更长的上升子序列。

再根据条件(3)，我们会得到一个启示：根据F[]的值进行分类。对于F[]的每一个取值k，我们只需要保留满足F[t] = k的所有A[t]中的最小值。设D[k]记录这个值，即D[k] = min{A[t]} (F[t] = k)。

注意到D[]的两个特点：

(1)　D[k]的值是在整个计算过程中是单调不下降的。

(2)　D[]的值是有序的，即D[1] < D[2] < D[3] < ... < D[n]。

利 用D[]，我们可以得到另外一种计算最长上升子序列长度的方法。设当前已经求出的最长上升子序列长度为len。先判断A[t]与D[len]。若A [t] > D[len]，则将A[t]接在D[len]后将得到一个更长的上升子序列，len = len + 1， D[len] = A [t]；否则，在D[1]..D[len]中，找到最大的j，满足D[j] < A[t]。令k = j + 1，则有A [t] <= D[k]，将A[t]接在D[j]后将得到一个更长的上升子序列，更新D[k] = A[t]。最后，len即为所要求的最长上 升子序列的长度。

在 上述算法中，若使用朴素的顺序查找在D[1]..D[len]查找，由于共有O(n)个元素需要计算，每次计算时的复杂度是O(n)，则整个算法的 时间复杂度为O(n^2)，与原来的算法相比没有任何进步。但是由于D[]的特点(2)，我们在D[]中查找时，可以使用二分查找高效地完成，则整个算法 的时间复杂度下降为O(nlogn)，有了非常显著的提高。需要注意的是，D[]在算法结束后记录的并不是一个符合题意的最长上升子序列！