3-1:

> *设计一个$O(n^2)$时间的算法，找出由n个数组成的序列的最长单调递增子序列。*

题目分析

我们要构造最长单调递增子序列，从直观上想，肯定是想尽可能取小的序列，增长的缓慢一点，使得之后可以添加更多的元素。

对于此题，我们规定："递增"的意思为$a_i \geq a_j(i > j)$（严格递增才不取等号），且该序列是一个**整数**序列。

我们使用动态规划，需要先找到最优子结构。

现有一 n 个元素的数字序列，设为 $L = \{a_1, a_2, \cdots, a_n\}$，设它的一条最优的最长单调递增子序列为$S_k = \{b_1, b_2, \cdots, b_k\}$【最优的含义为：在保证符合条件的前提下，该子序列的最后一个元素最小】，则必有如下性质：$S_k = S_{k-1} + a_p$，即长为 k 的最长单调递增子序列由长 $k-1$ 的子序列递推而来，且由于要使得 S_k 最优，S_{k-1} 也必须最优，否则可能存在 $b'_{k-1} > a_p$，即 S_k 不能取 a_p 为第 k 项，从而不是最优的。**因此本问题有最优子结构性质**。

据此，我们可以构造一个数组 DP，$DP[i]$ 表示在前 i 个数中以 $L[i]$ 为结尾的最长单调递增子序列长度。我们遍历序列 L 的同时遍历 DP 数组（需要两个 for 循环），若 $L[j] \leq L[i](j \leq i)$，则需判断 $DP[i]$ 是否需要修改，这代表 $L[i]$ 要么可以接在以 $L[j]$ 为结尾的最长单调递增子序列上，新子序列长度为 $DP[j] + 1$；要么就不接，长度仍为 $DP[i]$。

同时，为了构造最优解，我们还需开辟一个 pre 数组，$pre[i]$ 表示以 $L[i]$ 为结尾的最长单调递增子序列前一个元素在第几个位置。

由此，我们可以获得如下式子：

```
if(j <= i && L[j] <= L[i]){
    if(DP[j]+1 > DP[i]) pre[i] = j;
    DP[i] = max(DP[j]+1, DP[i]);
}
```

代码实现

详见 3-1 Code.cpp

输出示例

```
输入1:
9
1 5 2 6 9 10 3 15 14
输出1:
长度为: 6
1 2 6 9 10 14 // 最优子序列结尾应是14而非15
```

```

输入2：
4
4 3 2 1
输出2：
长度为：1
1

输入3：
15
3 5 4 7 3 2 1 6 3 4 6 9 8 11 12
输出3：
长度为：8
3 3 3 4 6 8 11 12

输入4：
10
3 -1 2 -4 6 3 5 7 -1 1
输出4：
长度为：5
-1 2 3 5 7

输入5：25
9 7 3 5 9 1 9 21 12 44 33 23 26 88 39 9 38 45 61 19 28 33 39 12 90
输出5：
长度为：11
3 5 9 9 12 23 26 28 33 39 90
```

算法分析

本算法分为几个部分：

- 输入部分：时间复杂度显然为 $O(n)$
- 求解 DP 过程，由于为两层循环，循环内部均为 $O(1)$，即总复杂度为 $O(n^2)$
- 遍历找到最优子序列：显然 $O(n)$
- 回溯最优解及输出部分：不超过 $O(n)$

因此，算法的总复杂度为 $O(n^2)$。