我的思路：最初想法参考LCS，提出假设"LIS的最后一个元素应为已知序列中最大的元素"。
但可举反例如 20，10，5，6，7，假设不成立。
考虑最优子结构，假设"当LIS下一个元素未出现前，LIS的该部分即为已知序列部分的LIS"。
但同如上序列，显然对前2个元素均不成立，故最优子结构不满足。

由上述可知，LIS具备"后来居上"的可能，
即后续较小数值的~~连续~~严格递增序列长度可超过原有局部的LIS。
因此已知序列中任意的严格递增序列都应被保存，以期其可能成为整个序列LIS的一部分。
由LIS的性质，LIS的最后一个元素一定严格大于LIS的倒数第二个元素。
因此考虑，每个LIS的最后元素应记录该LIS的目前长度，同时为了结果输出LIS，应设置前后连接关系。
数据结构应非本题关注重点，但实现上或可采用类似二叉树的数据结构。

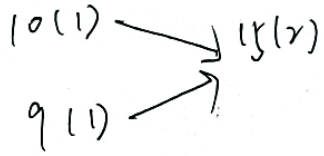
算法如下：

```
for i=0 to n:
    a[i].length=1 . // LIS长度均赋初值为1.
for i=1 to n:
    for j=0 to i-1:
        if a[j]≤a[i]
            a[j].child[i]=a[i] // a[j]下一个指向a[i]
            a[i]=max{a[i], a[j]+1} // LIS长度更新.
```

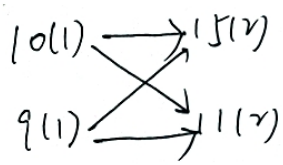
如果不考虑输出LIS结果，只求长度，则只须 a[i]=max{a[i], a[j]+1}.

例如：[10, 9, 15, 11, 12, 6]. 可类似散列或画树.
(括号内数字表示目前 length).

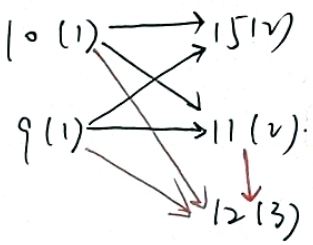
i=0, 1.



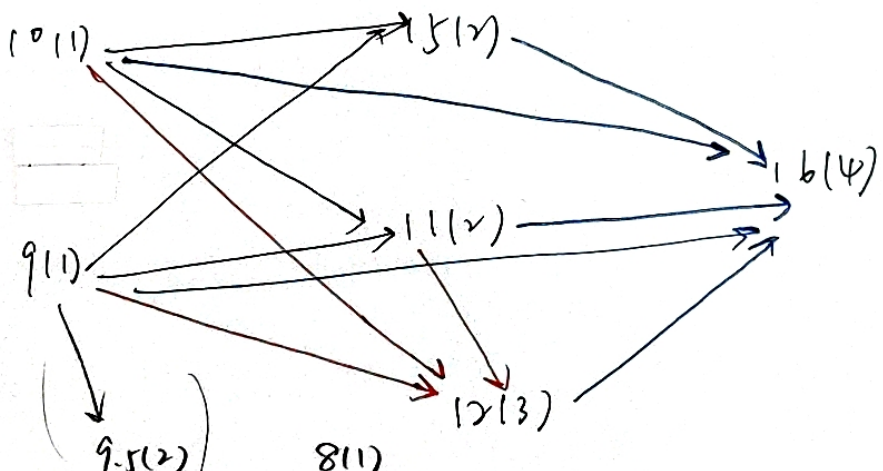
i=2



i=3



i=4



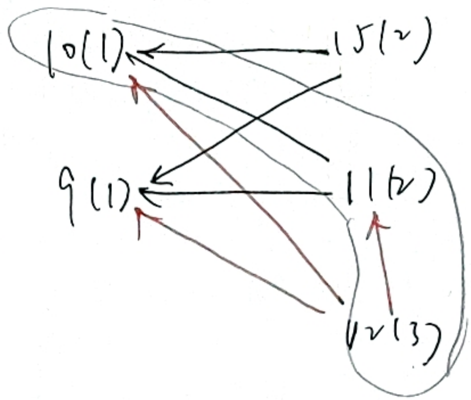
i=5

若不妨再加一个较小的如9.5，不影响LIS.
加更小的如8，分集它无连接.

由此LIS长度
即找出最大的 length 值
即可.

改变一下思路，为了方便输出LIS，
应该为 a[i].parent[j] = a[j].

如二中时



接上一步：找到并输出最大的 length 后，

从该节点找 parent 节点中 length 小1的节点，

由此一路回溯，可找到整个LIS。

复杂度分析：两个嵌套循环。

$$\sum_{i=1}^{n-1} i = 1+2+\cdots+(n-1) = n^2$$

$\therefore T(n) = O(n^2)$. （后续遍历输出为 $O(n)$，为小量可忽略）.