300. 最长上升子序列

算法 动态规划

> 题目：Longest Increasing Subsequence
> 语言: python3
> 英文版链接：https://leetcode.com/problems/longest-increasing-subsequence/description/
> 中文版链接：https://leetcode-cn.com/problems/longest-increasing-subsequence/

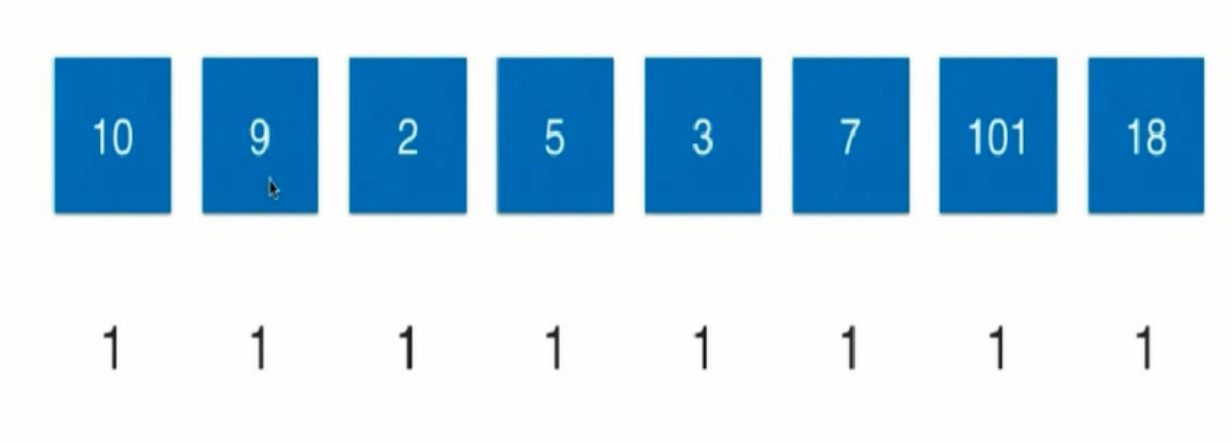
题目分析

这是一个经典的动态规划的题目。我们先来看题目中给的例子：

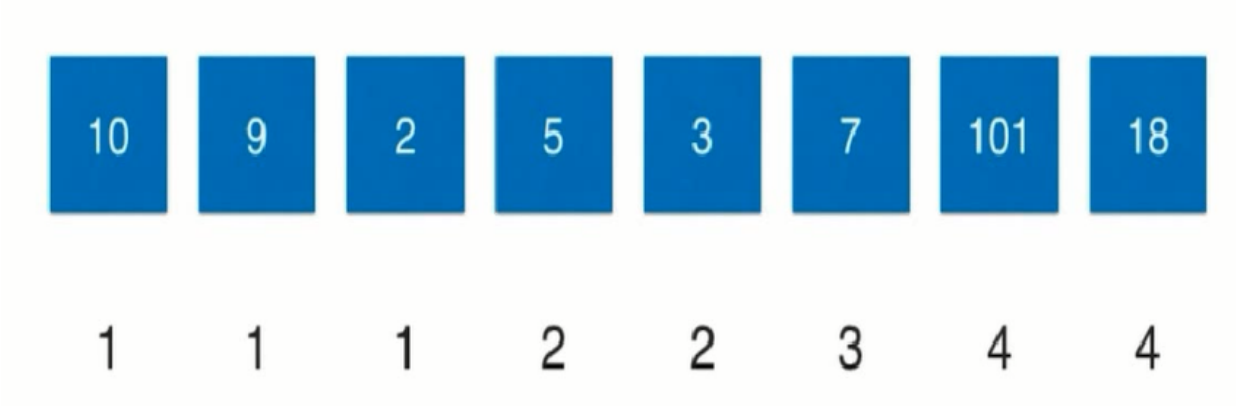
```
示例：

输入：[10,9,2,5,3,7,101,18]
输出：4
解释：最长的上升子序列是 [2,3,7,101]，它的长度是 4。
```

首先将它初始化一下：



然后很简单的推导我们能够得到下面这样子：

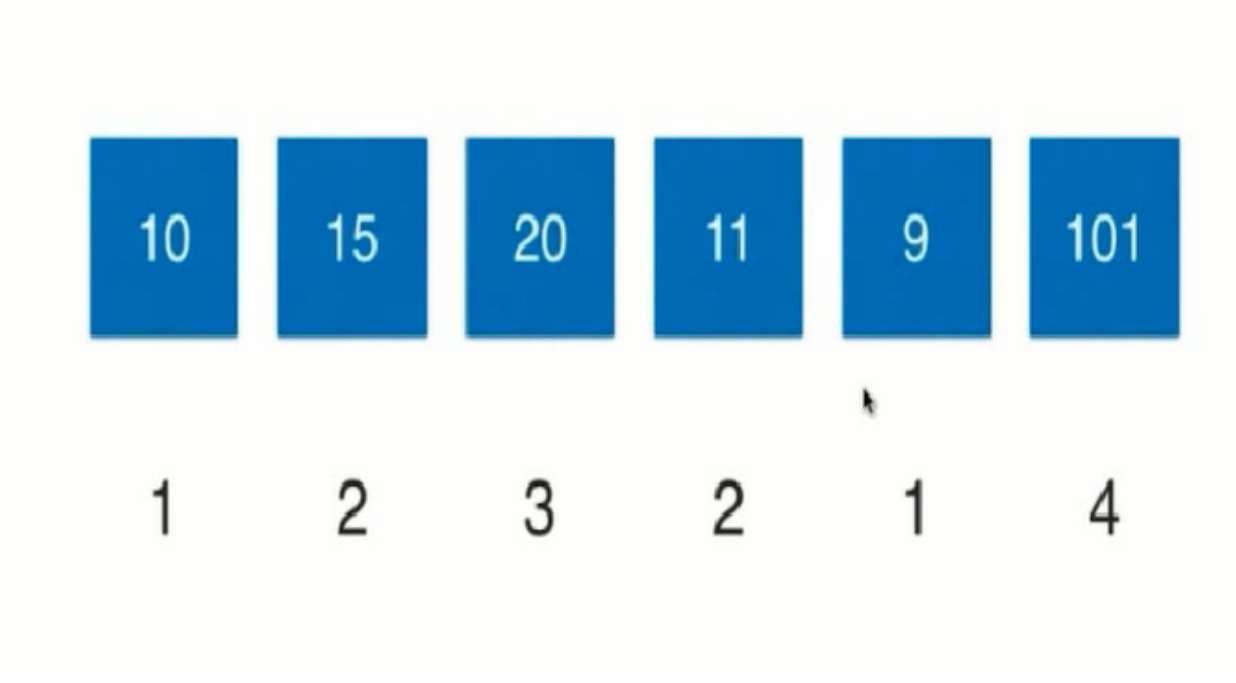


以5为例，5比10、9小，所以不构成最长上升子序列，5比2大，所以5接着2可以构成一个最长上升子序列。所以5的最长上升子序列长度是2。

例子很容易理解，我们来思考两个问题：

1. 下面的序列长度是不是一直是递增的，因为我们在这个例子里看到都是递增的？
2. 最后一个元素对应的序列长度是不是最终的解？

我们重新找一个例子来思考上面这两个问题：



可以很明显的发现，长度序列不一定是递增的，也不一定最后一个元素就是最终的解，比如序列在等于9是截断，也就是没有后面的101的话，实际的解应该是3而不是1。

结合上面的例子，我们给出最长上升子序列的递推式：
LIS(i) 表示以第i个数字为结尾的最长上升子序列的长度，那么有：

$$LIS(i) = max_{j<i}(1 + LIS(j) \ \ if nums[i] > nums[j])$$

答案

我们直接来进行动态规划算法的求解：

```python
class Solution:
    def lengthOfLIS(self, nums) -> int:
        if len(nums) == 0:
            return 0

        LIS = [1] * len(nums)

        for i in range(1, len(nums)):
            # 这里注意一点：比如10、9，这种可能前面找不到比9小的，那么这个list就有可能为空，所以加上一个[1]
            # 保证最小是0
            LIS[i] = max([1 + LIS[j] for j in range(i) if nums[j] < nums[i]] + [1])

        return max(LIS)
```