练习3 – 最长上升子序列（LIS）

- 最长上升子序列（最长递增子序列，Longest Increasing Subsequence，LIS）

- leetcode_300_最长上升子序列：https://leetcode-cn.com/problems/longest-increasing-subsequence/

- 给定一个无序的整数序列，求出它最长上升子序列的长度（要求严格上升）
  - 比如 [10, 2, 2, 5, 1, 7, 101, 18] 的最长上升子序列是 [2, 5, 7, 101]、[2, 5, 7, 18]，长度是 4

最长上升子序列 – 动态规划 – 状态定义

- 假设数组是 nums，[10, 2, 2, 5, 1, 7, 101, 18]
- dp(i) 是以 nums[i] 结尾的最长上升子序列的长度，i ∈ [0, nums.length)
  - ✓ 以 nums[0] 10 结尾的最长上升子序列是 10，所以 dp(0) = 1
  - ✓ 以 nums[1] 2 结尾的最长上升子序列是 2，所以 dp(1) = 1
  - ✓ 以 nums[2] 2 结尾的最长上升子序列是 2，所以 dp(2) = 1
  - ✓ 以 nums[3] 5 结尾的最长上升子序列是 2、5，所以 dp(3) = dp(1) + 1 = dp(2) + 1 = 2
  - ✓ 以 nums[4] 1 结尾的最长上升子序列是 1，所以 dp(4) = 1
  - ✓ 以 nums[5] 7 结尾的最长上升子序列是 2、5、7，所以 dp(5) = dp(3) + 1 = 3
  - ✓ 以 nums[6] 101 结尾的最长上升子序列是 2、5、7、101，所以 dp(6) = dp(5) + 1 = 4
  - ✓ 以 nums[7] 18 结尾的最长上升子序列是 2、5、7、18，所以 dp(7) = dp(5) + 1 = 4

- 最长上升子序列的长度是所有 dp(i) 中的最大值 max { dp(i) }，i ∈ [0, nums.length)

最长上升子序列 – 动态规划 – 状态转移方程

- 遍历 j ∈ [0, i)
  - 当 nums[i] > nums[j]
    - ✓ nums[i] 可以接在 nums[j] 后面，形成一个比 dp(j) 更长的上升子序列，长度为 dp(j) + 1
    - ✓ dp(i) = max { dp(i), dp(j) + 1 }
  - 当 nums[i] ≤ nums[j]
    - ✓ nums[i] 不能接在 nums[j] 后面，跳过此次遍历（continue）

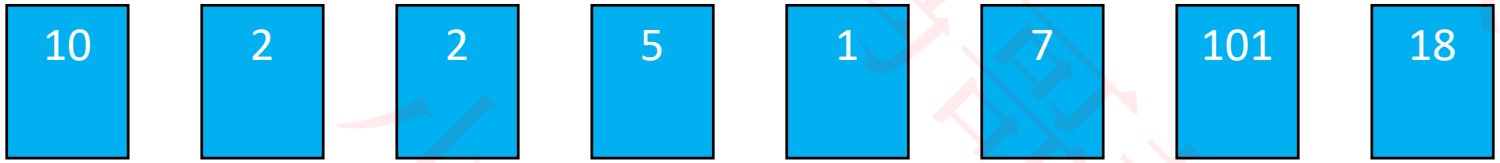
- 状态的初始值
  - dp(0) = 1
  - 所有的 dp(i) 默认都初始化为 1

最长上升子序列 – 动态规划 – 实现

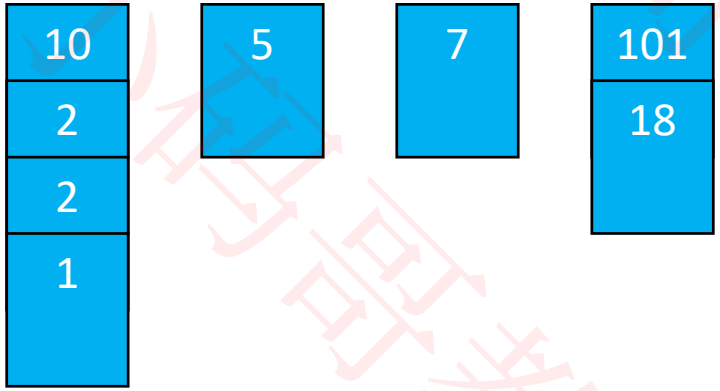
```
int lengthOfLIS(int[] nums) {
    if (nums == null || nums.length == 0) return 0;
    int[] dp = new int[nums.length];
    int max = dp[0] = 1;
    for (int i = 1; i < dp.length; i++) {
        dp[i] = 1;
        for (int j = 0; j < i; j++) {
            if (nums[i] <= nums[j]) continue;
            dp[i] = Math.max(dp[i], dp[j] + 1);
        }
        max = Math.max(dp[i], max);
    }
    return max;
}
```

- 空间复杂度：$O(n)$，时间复杂度：$O(n^2)$

最长上升子序列 – 二分搜索 – 思路



- 把每个数字看做是一张扑克牌，从左到右按顺序处理每一个扑克牌
  - 将它压在（从左边数过来）第一个牌顶 ≥ 它的牌堆上面
  - 如果找不到牌顶 ≥ 它的牌堆，就在最右边新建一个牌堆，将它放入这个新牌堆中



- 当处理完所有牌，最终牌堆的数量就是最长上升子序列的长度

最长上升子序列 – 二分搜索 – 思路

- 思路（假设数组是 nums，也就是最初的牌数组）
  - top[i] 是第 i 个牌堆的牌顶，len 是牌堆的数量，初始值为 0
  - 遍历每一张牌 num
    - ✓ 利用二分搜索找出 num 最终要放入的牌堆位置 index
    - ✓ num 作为第 index 个牌堆的牌顶，top[index] = num
    - ✓ 如果 index 等于 len，相当于新建一个牌堆，牌堆数量 +1，也就是 len++

最长上升子序列 – 二分搜索 – 实现

```java
int lengthOfLIS(int[] nums) {
    if (nums == null || nums.length == 0) return 0;
    int[] top = new int[nums.length];
    int len = 0;
    for (int num : nums) {
        int begin = 0, end = len;
        while (begin < end) {
            int mid = (begin + end) >> 1;
            if (num <= top[mid]) {
                end = mid;
            } else {
                begin = mid + 1;
            }
        }
        top[begin] = num;
        if (begin == len) len++;
    }
    return len;
}
```

- 空间复杂度：O(n)
- 时间复杂度：O(nlogn)