1. 题目
2. 题干

给定一个无序的整数数组，找到其中最长上升子序列的长度。

1. 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。

1. 题解
2. 思路

动态规划的题目，要找到的当前的最长子序列的长度，那么它一定包括之前的长度更短的上升子序列。设置dp数组，从小到大计算dp数组的值。可以从示例中看出，这个上升子序列并不一定是连续的，那么就不能仅仅通过判断前一个数来得出结果。即需要判断之前的所有dp数组的值，且要从之前的dp[j]状态转移到当前的dp[i]状态，需要当前的nums[i] > nums[j]，这样nums[i]才能够连接到nums[j]所在的子序列中。最后，找出dp数组中的最大值即可。

1. 代码实现

Java：

class Solution {

    public int lengthOfLIS(int[] nums) {

        int maxLength  = 1;

        if(nums == null || nums.length == 0){

            return 0;

        }

        int n = nums.length;

        int[] dp = new int[n];

        dp[0] = 1;

        for(int i= 1; i < n; ++i){

            dp[i] = 1;

            for(int j = 0; j < n; ++j){

                if(nums[i] > nums[j]){

                    dp[i] = Math.max(dp[i], dp[j]+1);

                }

            }

        }

        Arrays.sort(dp);

        return dp[n-1];

    }

}