LeetCode354-俄罗斯套娃信封问题

2020 年 9 月 28 日

1 题目描述

给定一些标记了宽度和高度的信封，宽度和高度以整数对形式 (w, h) 出现。当另一个信封的宽度和高度都比这个信封大的时候，这个信封就可以放进另一信封里，如同俄罗斯套娃一样。

请计算最多能有多少个信封能组成一组“俄罗斯套娃”信封（即可以把一个信封放到另一个信封里面）。

说明:

不允许选择信封。

示例:

```
Input: envelopes=[[5,4],[6,4],[6,7],[2,3]]
Output: 3
Explanation: The maximum number of envelopes you can Russian doll is 3
    ([2,3] => [5,4] => [6,7]).
```

2 解法

2.1 方法 1: 排序 + 最长递增子序列

该问题为最长递增子序列的二维问题。

我们要找的最长子序列，且满足 $seq[i+1]$ 中的元素大于 $seq[i]$ 中的元素。

该问题是输入按任意顺序排列的–我们不能直接套用标准的 LIS 算法，需要先对数据进行排序。我们如何对数据进行排序，以便我们的 LIS 算法总能找到最佳答案?

我们可以在这里 找到最长递增子序列的解决办法。

算法:

假设我们知道了信封套娃顺序，那么从里向外的顺序必须是按 w 升序排序的子序列。

在对信封按 w 进行排序以后，我们可以找到 h 上最长递增子序列的长度。

我们考虑输入 $[[1,3],[1,4],[1,5],[2,3]]$，如果我们直接对 h 进行 LIS 算法，我们将会得到 [3,4,5]，显然这不是我们想要的答案，因为 w 相同的信封是不能够套娃的。

为了解决这个问题。我们可以按 w 进行升序排序，若 w 相同则按 h 降序排序。则上述输入排序后为 $[[1,5],[1,4],[1,3],[2,3]]$，再对 h 进行 LIS 算法可以得到 $[5]$，长度为 1，是正确的答案。这个例子可能不明显。

我们将输入改为 $[[1,5],[1,4],[1,2],[2,3]]$。则提取 h 为 $[5,4,2,3]$。我们对 h 进行 LIS 算法将得到 $[2,3]$，是正确的答案。

```java
class Solution {

   public int lengthOfLIS(int[] nums) {
      int[] dp = new int[nums.length];
      int len = 0;
      for (int num : nums) {
         int i = Arrays.binarySearch(dp, 0, len, num);
         if (i < 0) {
            i = -(i + 1);
         }
         dp[i] = num;
         if (i == len) {
            len++;
         }
      }
      return len;
   }

   public int maxEnvelopes(int[][] envelopes) {
      // sort on increasing in first dimension and decreasing in second
      Arrays.sort(envelopes, new Comparator<int[]>() {
         public int compare(int[] arr1, int[] arr2) {
            if (arr1[0] == arr2[0]) {
               return arr2[1] - arr1[1];
            } else {
               return arr1[0] - arr2[0];
            }
        }
      });
      // extract the second dimension and run LIS
      int[] secondDim = new int[envelopes.length];
      for (int i = 0; i < envelopes.length; ++i) secondDim[i] = envelopes[i
          ][1];
      return lengthOfLIS(secondDim);
   }
}
```

复杂度分析:

- 时间复杂度: $O(N \log N)$。其中，N 是输入数组的长度。排序和 LIS 算法都是 $O(N \log N)$。
- 空间复杂度：lis 函数需要一个数组 dp，它的大小可达 N，另外，我们使用的排序算法也需要额外的空间。