Dynamic Programming | Set 21 (Variations of LIS)

We have discussed Dynamic Programming solution for Longest Increasing Subsequence problem in this post and a O(nLogn) solution in this post. Following are commonly asked variations of the standard LIS problem.

1. Building Bridges: Consider a 2-D map with a horizontal river passing through its center. There are n cities on the southern bank with x-coordinates a(1) … a(n) and n cities on the northern bank with x-coordinates b(1) … b(n). You want to connect as many north-south pairs of cities as possible with bridges such that no two bridges cross. When connecting cities, you can only connect city i on the northern bank to city i on the southern bank.

```
8     1     4     3     5     2     6     7
<---- Cities on the other bank of river---->
--------------------------------------------
  <--------------- River--------------->
--------------------------------------------
1     2     3     4     5     6     7     8
<------- Cities on one bank of river------->
```

Source: Dynamic Programming Practice Problems. The link also has well explained solution for the problem.

2. Maximum Sum Increasing Subsequence: Given an array of n positive integers. Write a program to find the maximum sum subsequence of the given array such that the intgers in the subsequence are sorted in increasing order. For example, if input is {1, 101, 2, 3, 100, 4, 5}, then output should be {1, 2, 3, 100}. The solution to this problem has been published here.

3. The Longest Chain You are given pairs of numbers. In a pair, the first number is smaller with respect to the second number. Suppose you have two sets (a, b) and (c, d), the second set can follow the first set if b < c. So you can form a long chain in the similar fashion. Find the longest chain which can be formed. The solution to this problem has been published here.

4. Box Stacking You are given a set of n types of rectangular 3-D boxes, where the i^th box has height h(i), width w(i) and depth d(i) (all real numbers). You want to create a stack of boxes which is as tall as possible, but you can only stack a box on top of another box if the dimensions of the 2-D base of the lower box are each strictly larger than those of the 2-D base of the higher box. Of course, you can rotate a box so that any side functions as its base. It is also allowable to use multiple instances of the same type of box.
Source: Dynamic Programming Practice Problems. The link also has well explained solution for the problem.