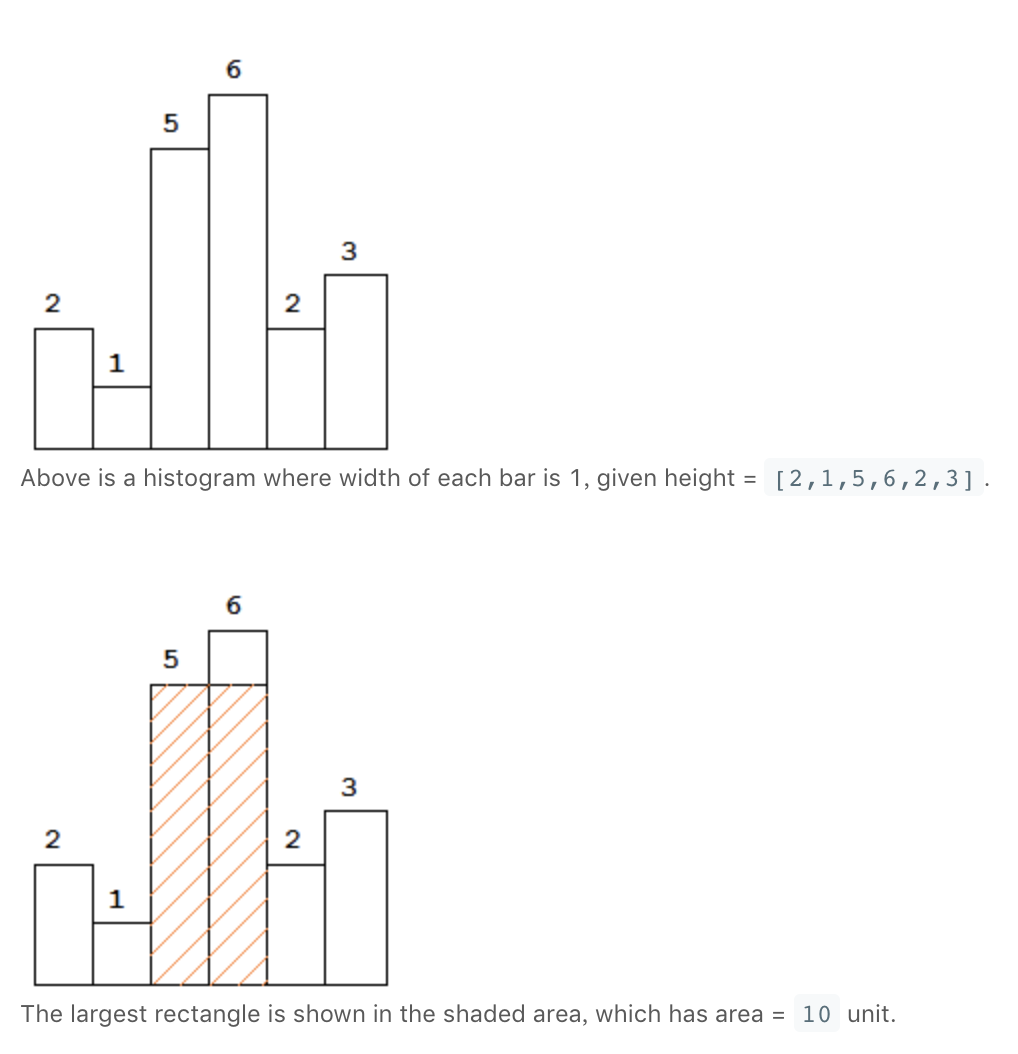
84．Largest Rectangle in Histogram

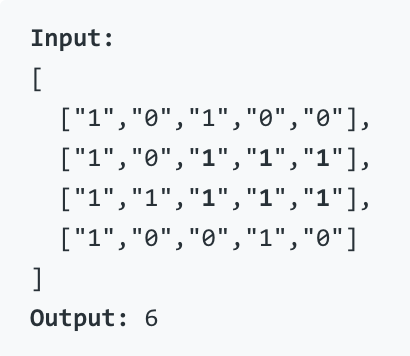


给定序列，找到标红的最大面积

解法：

循环序列每一位，从当前为往后找最大的面积。剪枝：往后找若是递增则continue

85．Maximal Rectangle



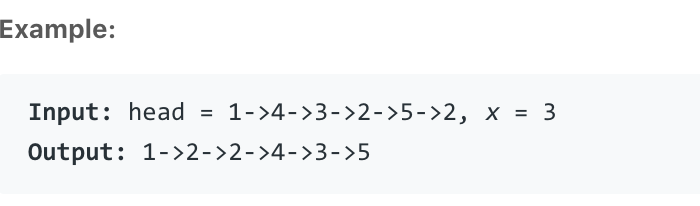
给定二维数组，找到最大的全为1的矩形。

解法：从每一行开始找起，维护一个一维序列。

一维序列表示当前行之上所有连续1的个数。此序列求最大面积转换成84题。

每行更新一次序列，求一次最大面积，最终所有行求得的最大面积即为解。

86．Partition List



给定串和一个数组，保留原串的顺序，将小于该数字的值挪到串头，大于等于该数的放在串尾。

解法：

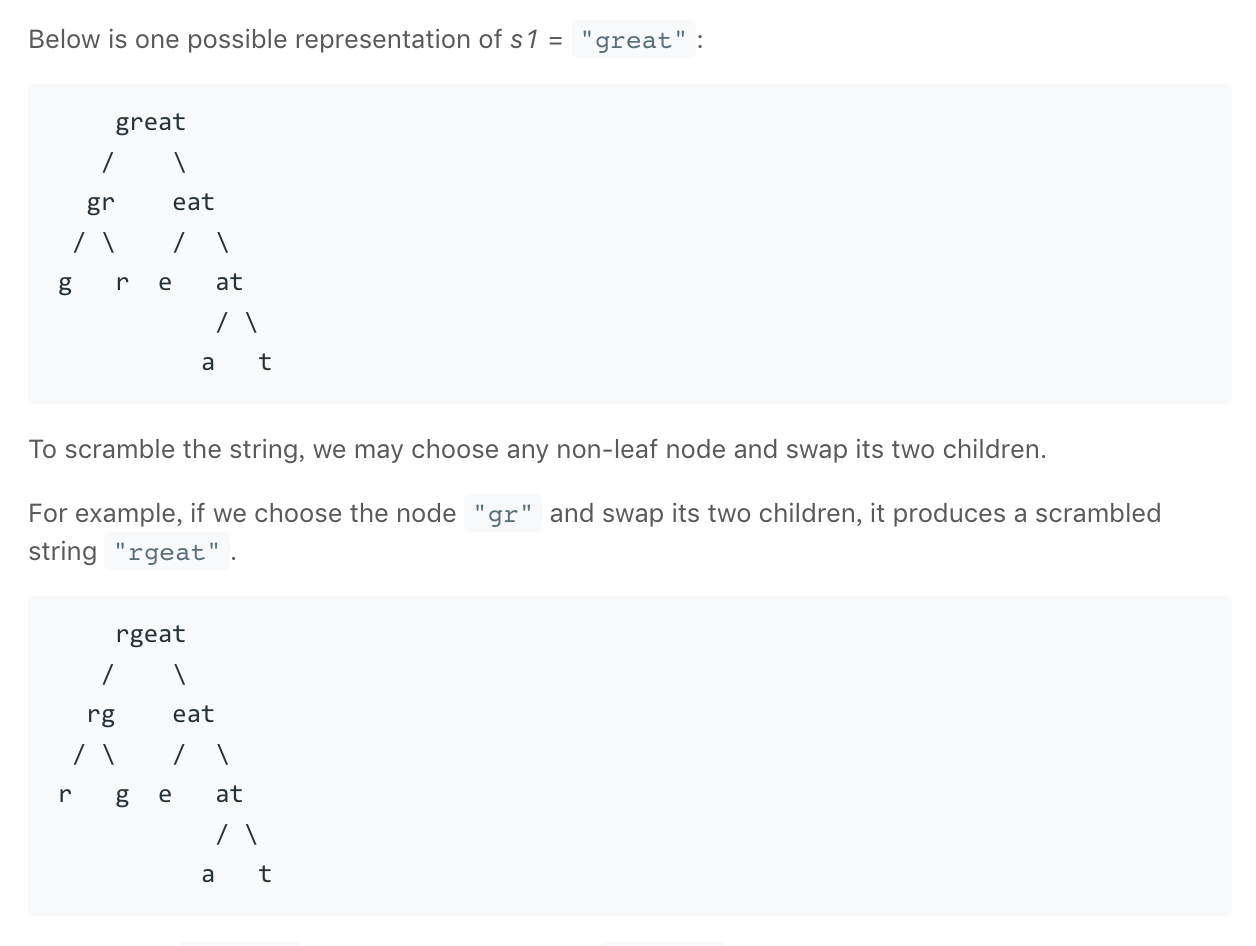
难点：维护小于数值的插入点、维护当前遍历结点的前一结点（变更位置的操作）

首先找到第一个小于该值的数放在串头，（若无则返回原始串）并维护插入点的位置。

从刚刚结点继续向后遍历，找到小值就插在插入点。更新插入点

插入时注意变动的值是否就是插入点的后继结点，是就不需要操作，不是再操作。

87 Scramble String



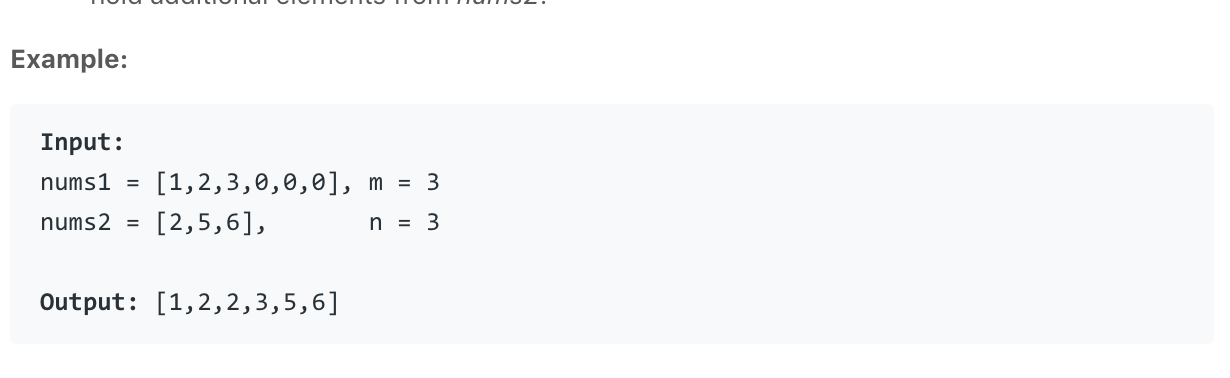
字符串对应树的某左右结点互换，是否能得到另一个字符串。

解法：

模拟递归，注意字符串拼接的结尾标志。

在每层将两个字符串分割，左右对比是否相等。

88 Merge Sorted Array



将nums2插入到nums1中（inplace），nums1有充分的空间存储nums2

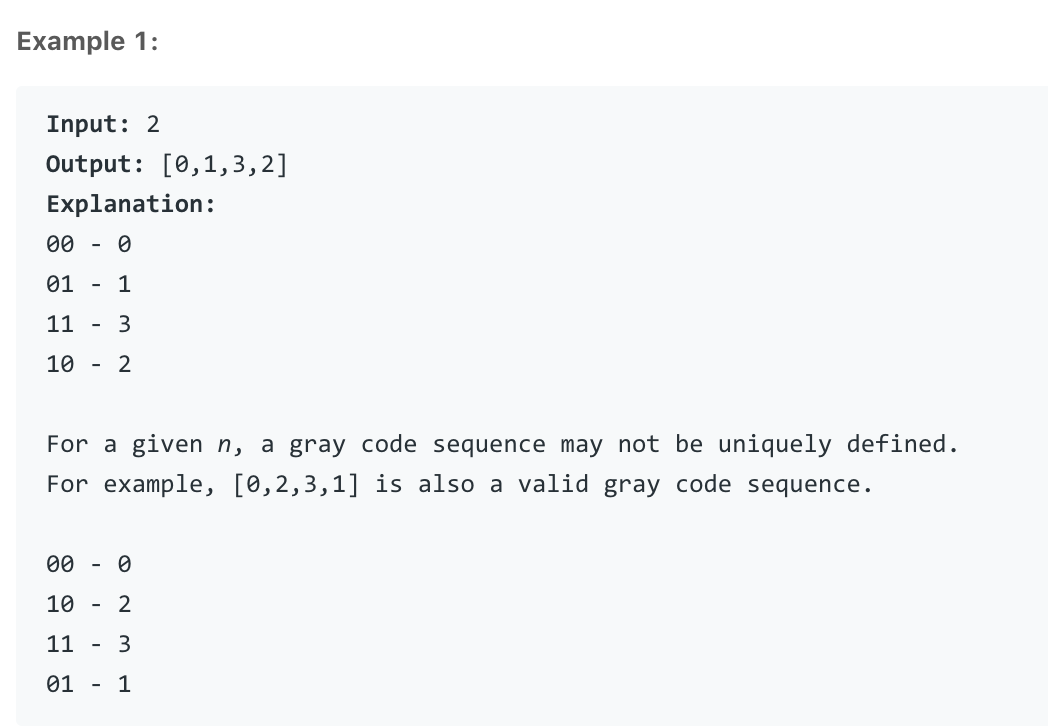
解法：

找到nums1[i]<nums2[j]…nums2[j+n]<nums1[i+1]，再插入

注意解决头插和尾插的解决。

计算nums1需要挪动位置的数的个数，以及需要挪动的位置。

89 gray code



解法：

格雷码，据说在通信领域应用广泛。

相邻数之间二进制只有一位不相同。

步骤一：初始数字

步骤二：改变最右一位

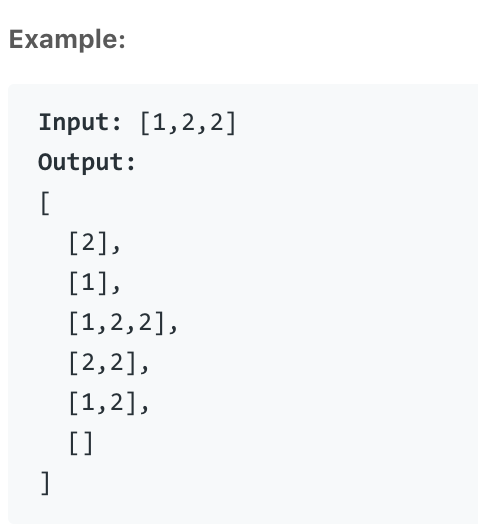
步骤三：改变最右1左边的一位

步骤四：重复二、三。直到2^n个数都算完

改变某一位：x^(1<<y) 改变x中第y位的值

寻找某数最右的第一个1：a&-a (lowbit)

90 Subsets II



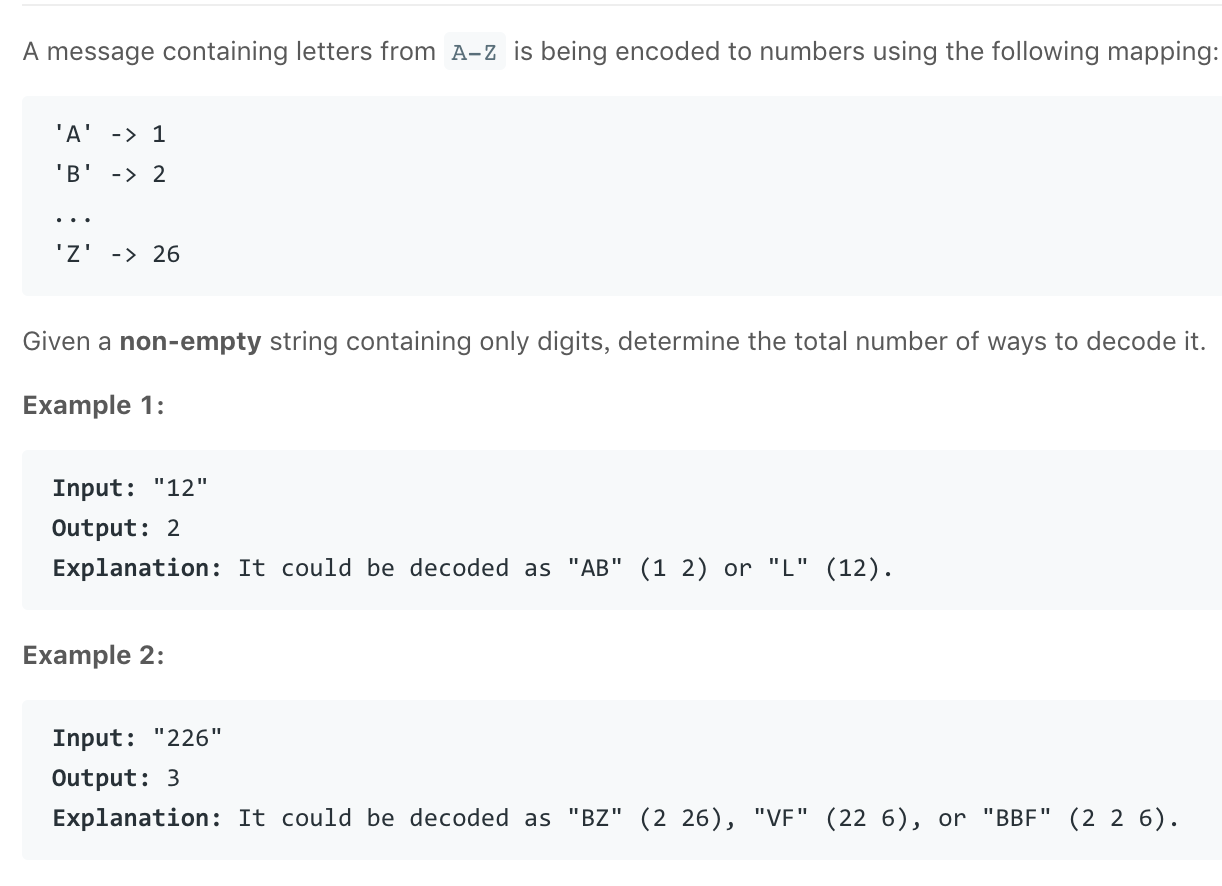
寻找给定串不重复的所有子串，该串会有重复元素。

解法：

排序 枚举长度 按长度找子串

不重复子串在找的时候需要把相邻的数字放在一起，假设有k个，然后枚举放0~k(不得超出长度)，重复操作直到找出长度相符的串。

91 Decode Ways



每个字母对应一个数字，给定一数字串，问有多少种字符串的可能。不能出现单个0的情况，出现即为不存在。

解法：

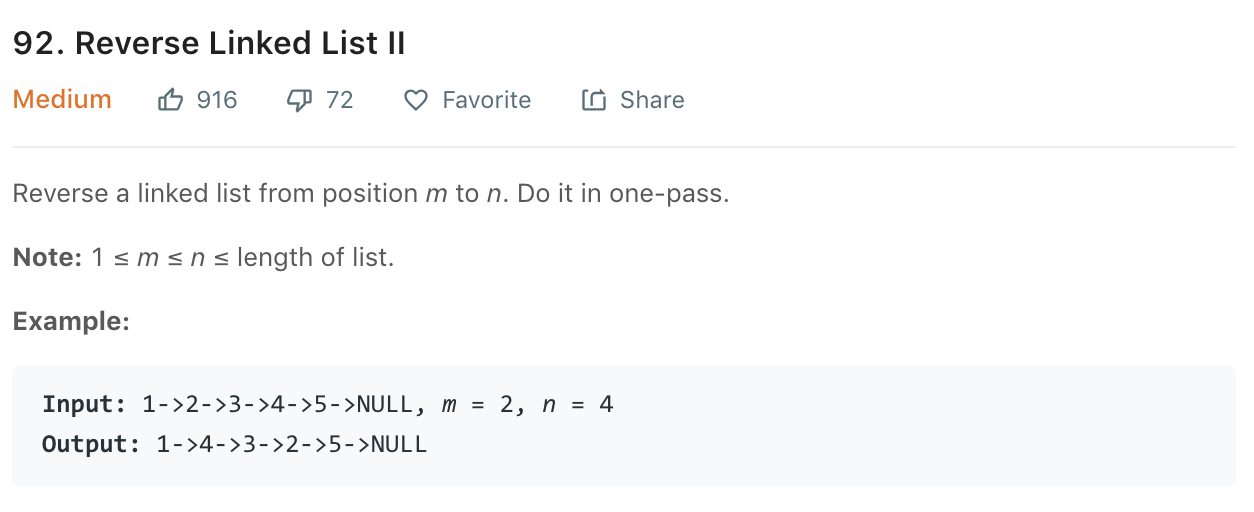
与爬楼梯相似（一次一格或两格，问最终有多少种组合的可能性）

每一位数都要考虑下面两种组合情况：

1当前数字单个成为一个字母为一种情况，个数与前一个数相同。单独若是0则为0。

2与前一个数字组成组合成为一种情况，个数与前两个相同（需要满足10~26的要求）

92 Reverse Linked List II

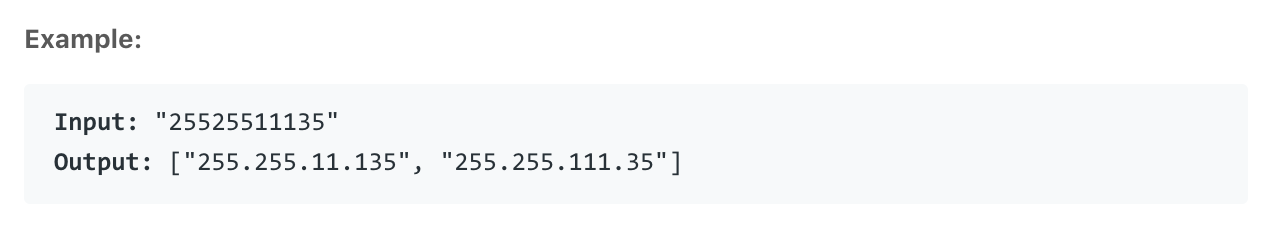


给定一个链表，和两个数字，翻转链表中第m到n之间的数。

找到第m和第n个数，头插法将m后的数依次插到n后

注意带头结点和不带头结点的区别

93 Restore IP Addresses



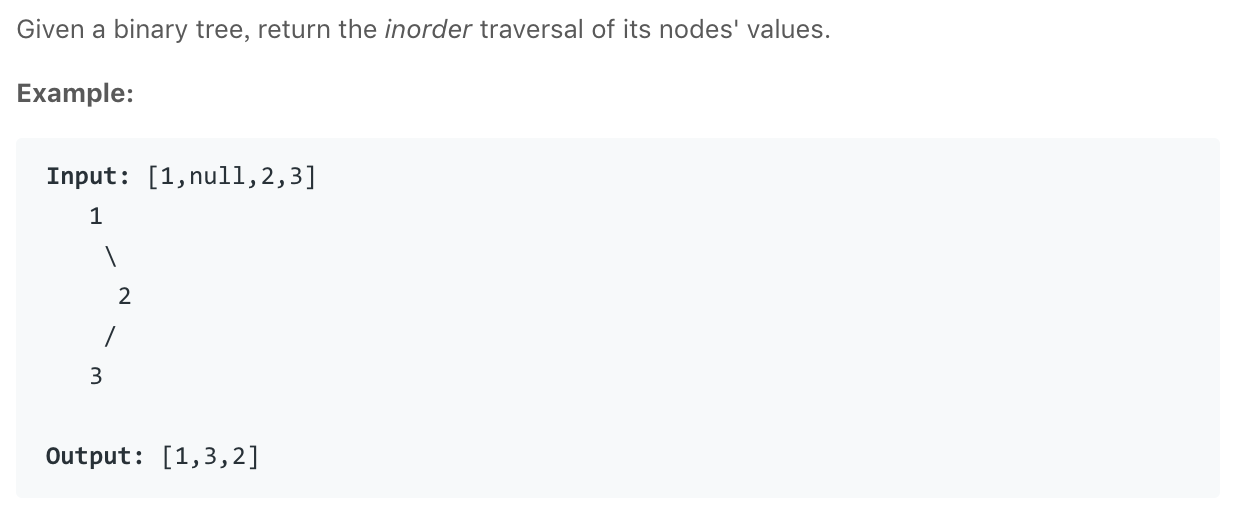
给定一个数字串，求出所有可能的IP地址

解法：

深度优先搜索

总共有四个位，每位数小于256且不能为空（0不能自己补），不能出现一个位连着两个0

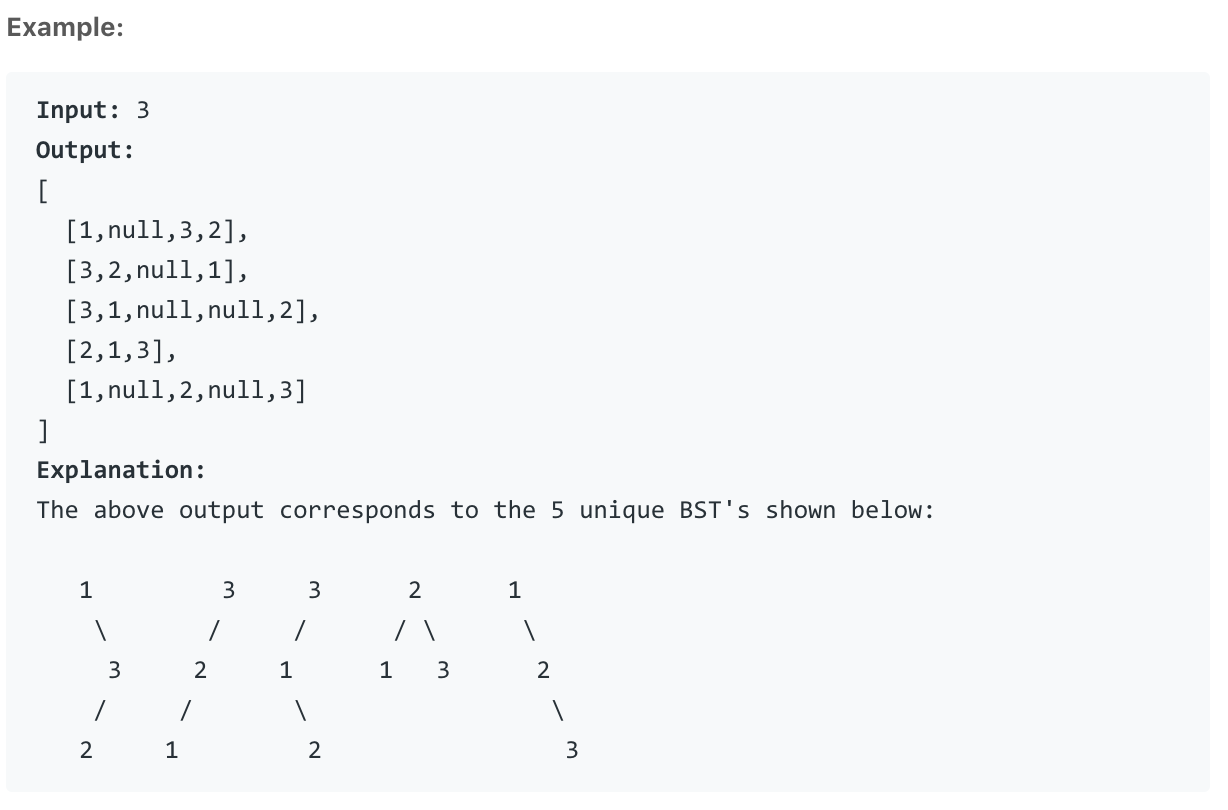
94 Binary Tree Inorder Traversal



给定一个二叉树，求中序遍历结果

解法：dfs中序遍历

95. Unique Binary Search Trees II



给定一个数字，求1~n区间内所有数值构成的所有的排序树。

解法：

每个数字都可能作为树根，然后按照该数字的左右开始分治。

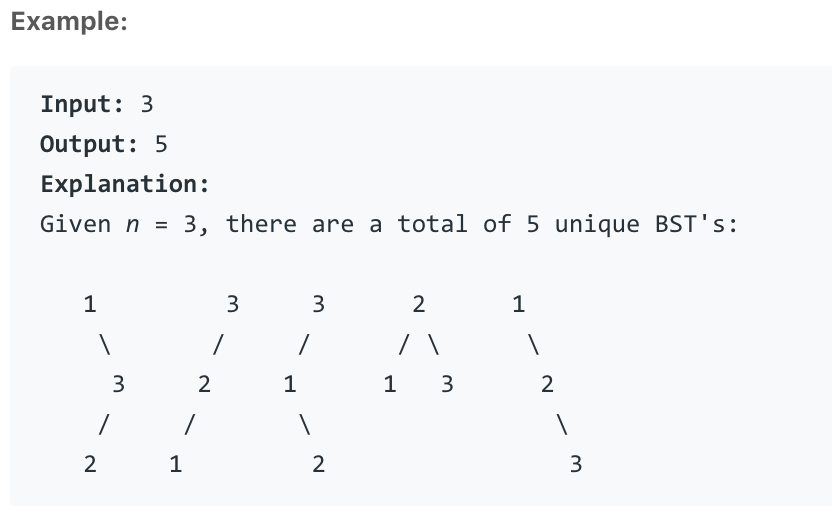
左边的数字都小于该数值，右边的数都大于该数值。

左边所有数字构成排序树的个数和右边所有数构成排序树做组合，就是以该数值为根所可能的所有排序树的可能。

注意左右子树可能为空，空的情况也算作是一种解。

枚举完所有的数值为根的情况，求完。

96 Unique Binary Search Trees



求95题中子树的个数

解法：

肯定不会按照95题构建“子森林”的路子来，一眼看来就是dp

思想还是类似，求“左子森林”和“右子森林”的个数，求积

“左右子森林”中结点的和等于结点总数即可

Dp[0] = 1

Dp[1] = 1

Dp[2] = dp[0]\*dp[1] + dp[1]\*dp[0]

Dp[3] = dp[0]\*dp[2] + dp[1]\*d[1] + dp[2]\*dp[0]

Dp[4] = dp[0]\*dp[3] + dp[1]\*dp[2] + dp[2]\*dp[1] + dp[3]\*dp[0]

…

Dp[n] = sum(dp[0]\*dp[n-1] + … + dp[i]\*dp[n-i-1] + … + dp[n]\*dp[0])