目录

[49. Group Anagrams 1](#_Toc502317321)

[50. Pow(x, n) 1](#_Toc502317322)

[53. Maximum Subarray 1](#_Toc502317323)

[54. Spiral Matrix 2](#_Toc502317324)

[55. Jump Game 2](#_Toc502317325)

[56. Merge Intervals 2](#_Toc502317326)

[57. Insert Interval 2](#_Toc502317327)

[58. Length of Last Word 3](#_Toc502317328)

[59. Spiral Matrix II 3](#_Toc502317329)

[60. Permutation Sequence 3](#_Toc502317330)

[61. Rotate List 3](#_Toc502317331)

[62. Unique Paths 4](#_Toc502317332)

[63. Unique Paths II 4](#_Toc502317333)

[64. Minimum Path Sum 4](#_Toc502317334)

[65. Valid Number 4](#_Toc502317335)

[66. Plus One 5](#_Toc502317336)

[67. Add Binary 5](#_Toc502317337)

[68. Text Justification 5](#_Toc502317338)

[69. Sqrt(x) 5](#_Toc502317339)

[70. Climbing Stairs 5](#_Toc502317340)

[71. Simplify Path 6](#_Toc502317341)

[73. Set Matrix Zeroes 6](#_Toc502317342)

[74. Search a 2D Matrix 6](#_Toc502317343)

[75. Sort Colors 6](#_Toc502317344)

[76. Minimum Window Substring 7](#_Toc502317345)

[77. Combinations 7](#_Toc502317346)

[78. Subsets 7](#_Toc502317347)

[79. Word Search 8](#_Toc502317348)

[80. Remove Duplicates from Sorted Array II 8](#_Toc502317349)

[81. Search in Rotated Sorted Array II 8](#_Toc502317350)

[82. Remove Duplicates from Sorted List II 8](#_Toc502317351)

[83. Remove Duplicates from Sorted List 8](#_Toc502317352)

[84. Largest Rectangle in Histogram 9](#_Toc502317353)

[85. Maximal Rectangle 9](#_Toc502317354)

[86. Partition List 9](#_Toc502317355)

[87. Scramble String 10](#_Toc502317356)

[88. Merge Sorted Array 10](#_Toc502317357)

[89. Gray Code 10](#_Toc502317358)

[90. Subsets II 11](#_Toc502317359)

[91. Decode Ways 11](#_Toc502317360)

[92. Reverse Linked List II 11](#_Toc502317361)

[93. Restore IP Addresses 12](#_Toc502317362)

[94. Binary Tree Inorder Traversal 12](#_Toc502317363)

[95. Unique Binary Search Trees II 12](#_Toc502317364)

[96. Unique Binary Search Trees 12](#_Toc502317365)

[97. Interleaving String 13](#_Toc502317366)

[98. Validate Binary Search Tree 13](#_Toc502317367)

[99. Recover Binary Search Tree 13](#_Toc502317368)

[100. Same Tree 13](#_Toc502317369)

算法题做法：  
1.看题目  
2.简单想想方法5min  
3.看答案5个  
4.写一个答案作为练习  
5.记录

## 1. Two Sum

Given an array of integers, return **indices** of the two numbers such that they add up to a specific target.

You may assume that each input would have ***exactly*** one solution, and you may not use the *same* element twice.

**Example:**

Given nums = [2, 7, 11, 15], target = 9,

Because nums[**0**] + nums[**1**] = 2 + 7 = 9,

return [**0**, **1**].

思路：使用哈希表HashMap来寻找一对有关系的值。可以一边遍历寻找当前元素的补值（和target的差）一边向散列表加入元素。如果加入元素的操作置后，则无需担心找到的补值是当前元素本身。

containsKey(…) 成员函数，查看是否含有该键的条目

get(…) 成员函数，获取对应键的value

## 2. Add Two Numbers

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order** and each of their nodes contain a single digit. Add the two numbers and return it as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

**Example**

**Input:** (2 -> 4 -> 3) + (5 -> 6 -> 4)

**Output:** 7 -> 0 -> 8

**Explanation:** 342 + 465 = 807.

## 49. Group Anagrams

Given an array of strings, group anagrams together.

For example, given: ["eat", "tea", "tan", "ate", "nat", "bat"],   
Return:

[

["ate", "eat","tea"],

["nat","tan"],

["bat"]

]

**Note:** All inputs will be in lower-case.

## 50. Pow(x, n)

Implement pow(x, n).

**Example 1:**

**Input:** 2.00000, 10

**Output:** 1024.00000

**Example 2:**

**Input:** 2.10000, 3

**Output:** 9.26100

思路：最常用的巧妙办法就是将n视作二进制数，不断除2取余可确定每位二进制取值。最后使用动态规划的办法，且利用常数空间可以从低位到高位逐渐乘出结果，每个二进制位对应的因子都是其前一位的平方（一个乘法即可）。

## 51. N-Queens

The *n*-queens puzzle is the problem of placing *n* queens on an *n*×*n* chessboard such that no two queens attack each other.



Given an integer *n*, return all distinct solutions to the *n*-queens puzzle.

Each solution contains a distinct board configuration of the *n*-queens' placement, where 'Q' and '.' both indicate a queen and an empty space respectively.

For example,  
There exist two distinct solutions to the 4-queens puzzle:

[

[".Q..", // Solution 1

"...Q",

"Q...",

"..Q."],

["..Q.", // Solution 2

"Q...",

"...Q",

".Q.."]

]

思路：教科书题目，试探回溯法

## 52. N-Queens II

Follow up for N-Queens problem.

Now, instead outputting board configurations, return the total number of distinct solutions.

思路：教科书题目

## 53. Maximum Subarray

Find the contiguous subarray within an array (containing at least one number) which has the largest sum.

For example, given the array [-2,1,-3,4,-1,2,1,-5,4],  
the contiguous subarray [4,-1,2,1] has the largest sum = 6.

**More practice:**

If you have figured out the O(*n*) solution, try coding another solution using the divide and conquer approach, which is more subtle.

思路：简单的方法就是使用动态规划或类似的简便解法（附加要求实际上将解法复杂化了，只是为了练习的目的）。动态规划的一般思路都是由小规模问题逐渐扩大到所求规模，每扩大一次规模（一般是增加一个被考虑元素）考虑新加元素对于整体问题的影响和上一规模的结果如何利用至新规模的问题。针对本题而言，每个新加元素在问题规模上增加了“包含新加元素的所有子数组”，这些子数组中的最大值和上一规模的问题结果比较即为当前规模的问题结果。“包含新加元素的所有子数组”也是一个线性规模的量，可以试图将其分解至前面的子问题中（前提是在子问题的范围内可以解决），从而每个子问题只解决常数个子数组比较。另外，如果每个新规模的问题只需要前一或前确定个规模的问题的结果，则动态规划可只用常数空间（用常数个变量存储这些值并更新即可），否则需要线性空间（一个数组来永久存储每一步的结果并不断插入新规模问题结果）。

## 54. Spiral Matrix

Given a matrix of *m* x *n* elements (*m* rows, *n* columns), return all elements of the matrix in spiral order.

For example,  
Given the following matrix:

[

[ 1, 2, 3 ],

[ 4, 5, 6 ],

[ 7, 8, 9 ]

]

You should return [1,2,3,6,9,8,7,4,5].

思路：第一种是模拟simulation式解法，即创建一个布尔矩阵用来标记当前元素是否被访问，创建一对四维向量代表两个维度上四个方向的运动。每次遇到“阻碍”，元素访问顺序或运动方向沿顺时针变化（顺时针变化可以体现在一对四维向量中）。第二种也是类似的直观解法，将矩阵由外向内一层一层访问，通过对矩阵几何尺度的运算，可以得出遍历轨迹的数学表达式。

## 55. Jump Game

Given an array of non-negative integers, you are initially positioned at the first index of the array.

Each element in the array represents your maximum jump length at that position.

Determine if you are able to reach the last index.

For example:  
A = [2,3,1,1,4], return true.

A = [3,2,1,0,4], return false.

思路：很直观的题目，注意的是审题，每个数字指的是最大跳跃高度而不是跳跃高度。

## 56. Merge Intervals

Given a collection of intervals, merge all overlapping intervals.

For example,  
Given [1,3],[2,6],[8,10],[15,18],  
return [1,6],[8,10],[15,18].

思路：先排序再合并。排序的参照是interval的下界。这种没有明确指名考察排序算法的题目，可以使用库函数。本题使用Collections.sort(Collections e, Comparator<T> c)来为任意Java库自带集合e排序（如List等均属集合）。Comparator<T>是比较器接口，如果集合的底层数据类型是自定义的，则必须自定义类实现该接口（用自定义类型代替T），并重写int compare(T a, T b)方法。该方法在a<b时应返回-1，a==b时应返回0，a>b时应返回1。

## 57. Insert Interval

Given a set of *non-overlapping* intervals, insert a new interval into the intervals (merge if necessary).

You may assume that the intervals were initially sorted according to their start times.

**Example 1:**  
Given intervals [1,3],[6,9], insert and merge [2,5] in as [1,5],[6,9].

**Example 2:**  
Given [1,2],[3,5],[6,7],[8,10],[12,16], insert and merge [4,9] in as [1,2],[3,10],[12,16].

This is because the new interval [4,9] overlaps with [3,5],[6,7],[8,10].

思路：直观的顺序处理即可，只是需要考虑各种test case的情况，比如空输入等。

## 58. Length of Last Word

Given a string *s* consists of upper/lower-case alphabets and empty space characters ' ', return the length of last word in the string.

If the last word does not exist, return 0.

**Note:** A word is defined as a character sequence consists of non-space characters only.

**Example:**

**Input:** "Hello World"

**Output:** 5

思路：直观解法即可，使用String类的成员函数charAt(int index)返回索引字符。

## 59. Spiral Matrix II

Given an integer *n*, generate a square matrix filled with elements from 1 to *n*2 in spiral order.

For example,  
Given *n* = 3,

You should return the following matrix:

[

[ 1, 2, 3 ],

[ 8, 9, 4 ],

[ 7, 6, 5 ]

]

思路：和54题道理是一样的。

## 60. Permutation Sequence

The set [1,2,3,…,*n*] contains a total of *n*! unique permutations.

By listing and labeling all of the permutations in order,  
We get the following sequence (ie, for *n* = 3):

1. "123"
2. "132"
3. "213"
4. "231"
5. "312"
6. "321"

Given *n* and *k*, return the *k*th permutation sequence.

**Note:** Given *n* will be between 1 and 9 inclusive.（表示1-9范围内且包含1和9）

## 61. Rotate List

Given a list, rotate the list to the right by *k* places, where *k* is non-negative.

**Example:**

Given **1->2->3->4->5->NULL** and *k* = **2**,

return **4->5->1->2->3->NULL**.

思路：可以先遍历一次列表，确认长度、尾节点等信息。再行操作。

## 62. Unique Paths

A robot is located at the top-left corner of a *m* x *n* grid (marked 'Start' in the diagram below).

The robot can only move either down or right at any point in time. The robot is trying to reach the bottom-right corner of the grid (marked 'Finish' in the diagram below).

How many possible unique paths are there?



Above is a 3 x 7 grid. How many possible unique paths are there?

**Note:** *m* and *n* will be at most 100.

思路：经典的小题目，到达每个格子的独特步数等于到达其左格和上格的独特步数之和，因此采用动态规划法即可。但是注意动态规划法可以优化空间复杂度。

## 63. Unique Paths II

Follow up for "Unique Paths":

Now consider if some obstacles are added to the grids. How many unique paths would there be?

An obstacle and empty space is marked as 1 and 0 respectively in the grid.

For example,

There is one obstacle in the middle of a 3x3 grid as illustrated below.

[

[0,0,0],

[0,1,0],

[0,0,0]

]

The total number of unique paths is 2.

**Note:** *m* and *n* will be at most 100.

思路：同62题，只不过阻碍处无法到达，故取步数为0.

## 64. Minimum Path Sum

Given a *m* x *n* grid filled with non-negative numbers, find a path from top left to bottom right which *minimizes* the sum of all numbers along its path.

**Note:** You can only move either down or right at any point in time.

**Example 1:**

[[1,3,1],

[1,5,1],

[4,2,1]]

Given the above grid map, return 7. Because the path 1→3→1→1→1 minimizes the sum.

思路：仍然类似63，62题，只是运算方式不同而已，算法流程一致。

## 65. Valid Number

Validate if a given string is numeric.

Some examples:  
"0" => true  
" 0.1 " => true  
"abc" => false  
"1 a" => false  
"2e10" => true

**Note:** It is intended for the problem statement to be ambiguous. You should gather all requirements up front before implementing one.

## 66. Plus One

Given a non-negative integer represented as a **non-empty** array of digits, plus one to the integer.

You may assume the integer do not contain any leading zero, except the number 0 itself.

The digits are stored such that the most significant digit is at the head of the list.

思路：直观的做法，需要考虑各种情况，包括类似999的情况，只要遇到某一位小于9，就可以中断运算。因为是向量，可以直接通过length访问最后一个元素，或从后向前访问。

## 67. Add Binary

Given two binary strings, return their sum (also a binary string).

For example,  
a = "11"  
b = "1"  
Return "100".

思路：一个字符串运算的题目，可以很直观的计算，需要另定义个变量carry来作为进位和求和使用（确实两个功能可以合一）。对于数字字符和对应数值之间的转化，可以使用简单的A-‘0’即可，所求结果如果赋给int值，则自动转化。

## 68. Text Justification

## 69. Sqrt(x)

Implement int sqrt(int x).

Compute and return the square root of *x*.

**x** is guaranteed to be a non-negative integer.

思路：一般的方法就是二分查找法，不断检验中点值的平方和目标的相对大小，来决定下一次查找方向。高阶的方法是使用newton迭代法，按照一个特定公式进行求解。

## 70. Climbing Stairs

You are climbing a stair case. It takes *n* steps to reach to the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

**Note:** Given *n* will be a positive integer.

**Example 1:**

**Input:** 2

**Output:** 2

**Explanation:** There are two ways to climb to the top.

1. 1 step + 1 step

2. 2 steps

**Example 2:**

**Input:** 3

**Output:** 3

**Explanation:** There are three ways to climb to the top.

1. 1 step + 1 step + 1 step

2. 1 step + 2 steps

3. 2 steps + 1 step

思路：最普通的办法是动态规划，简便一点的方法是斐波那契数列法，因为对于每一个n的结果，都是n-1和n-2的结果之和，不过后一种方法只适用于这一道题。

## 71. Simplify Path

Given an absolute path for a file (Unix-style), simplify it.

For example,

**path** = "/home/", => "/home"

**path** = "/a/./b/../../c/", => "/c"

**Corner Cases:**

•Did you consider the case where **path** = "/../"?  
In this case, you should return "/".

•Another corner case is the path might contain multiple slashes '/' together, such as "/home//foo/".  
In this case, you should ignore redundant slashes and return "/home/foo".

## 73. Set Matrix Zeroes

Given a *m* x *n* matrix, if an element is 0, set its entire row and column to 0. Do it in place.

**Follow up:**

Did you use extra space?  
A straight forward solution using O(*mn*) space is probably a bad idea.  
A simple improvement uses O(*m* + *n*) space, but still not the best solution.  
Could you devise a constant space solution?

思路：用矩阵的第一行和第一列作为存储行列状态的容器。第一行第一列的交点只能充当一个容器，因此需要额外的变量，但是只占常数空间。

## 74. Search a 2D Matrix

Write an efficient algorithm that searches for a value in an *m* x *n* matrix. This matrix has the following properties:

• Integers in each row are sorted from left to right.

• The first integer of each row is greater than the last integer of the previous row.

For example,

Consider the following matrix:

[

[1, 3, 5, 7],

[10, 11, 16, 20],

[23, 30, 34, 50]

]

Given **target** = 3, return true.

思路：当成一个连续的有序向量即可。

## 75. Sort Colors

Given an array with *n* objects colored red, white or blue, sort them so that objects of the same color are adjacent, with the colors in the order red, white and blue.

Here, we will use the integers 0, 1, and 2 to represent the color red, white, and blue respectively.

**Note:**  
You are not suppose to use the library's sort function for this problem.

A rather straight forward solution is a two-pass algorithm using counting sort.  
First, iterate the array counting number of 0's, 1's, and 2's, then overwrite array with total number of 0's, then 1's and followed by 2's.

Could you come up with an one-pass algorithm using only constant space?

思路：如题所示可以使用查数排序，首先计算三个数字的出现次数，然后对一个新数组进行填充即可。但这需要2个pass。另外可以使用交换法（将所有的0置左，2置右），和直接排序法（遍历一遍数组，遍历过的区域即已排序区域，为三个数字分别维护一个秩，指向其所在的子数组的最后一个元素）

## 76. Minimum Window Substring

Given a string S and a string T, find the minimum window in S which will contain all the characters in T in complexity O(n).

For example,  
**S** = "ADOBECODEBANC"  
**T** = "ABC"

Minimum window is "BANC".

**Note:**  
If there is no such window in S that covers all characters in T, return the empty string "".

If there are multiple such windows, you are guaranteed that there will always be only one unique minimum window in S.

## 77. Combinations

Given two integers *n* and *k*, return all possible combinations of *k* numbers out of 1 ... *n*.

For example,  
If *n* = 4 and *k* = 2, a solution is:

[

[2,4],

[3,4],

[2,3],

[1,2],

[1,3],

[1,4],

]

思路：使用直观递归法即可。第一位循环取值，剩下的位数递归，最后1位递归基也循环取值。这种逐渐缩小范围的递归法很常见。

## 78. Subsets

Given a set of **distinct** integers, *nums*, return all possible subsets (the power set).

**Note:** The solution set must not contain duplicate subsets.

For example,  
If ***nums*** = [1,2,3], a solution is:

[

[3],

[1],

[2],

[1,2,3],

[1,3],

[2,3],

[1,2],

[]

]

思路：简化版90题

## 79. Word Search

Given a 2D board and a word, find if the word exists in the grid.

The word can be constructed from letters of sequentially adjacent cell, where "adjacent" cells are those horizontally or vertically neighboring. The same letter cell may not be used more than once.

For example,  
Given **board** =

[

['A','B','C','E'],

['S','F','C','S'],

['A','D','E','E']

]

**word** = "ABCCED", -> returns true,

**word** = "SEE", -> returns true,

**word** = "ABCB", -> returns false.

思路：首先遍历所有的元素，对于每个元素和其周边的元素依次顺序检验是否和单词相同，注意考虑边界限制和已访问元素不能再访问。道理上形同试探回溯法，但是语法上可以只不过是“或”关系连接的四个方向上的递归。

## 80. Remove Duplicates from Sorted Array II

Follow up for "Remove Duplicates":  
What if duplicates are allowed at most *twice*?

For example,  
Given sorted array *nums* = [1,1,1,2,2,3],

Your function should return length = 5, with the first five elements of *nums* being 1, 1, 2, 2 and 3. It doesn't matter what you leave beyond the new length.

思路：按照直观思路去编程即可。但是有一种简单思路，因为数组是排序的，遍历一次数组，只有当前元素大于两个位置之前的元素才将该元素加入新数组，最后新数组即为所求数组。

## 81. Search in Rotated Sorted Array II

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand.

(i.e., 0 1 2 4 5 6 7 might become 4 5 6 7 0 1 2).

Write a function to determine if a given target is in the array.

The array may contain duplicates.

思路：使用变化版二分法，不仅比较目标和中点值，也要包括边界值lo和hi的比较，以便确定中点相对于pivot的位置，这是不需要提前确认pivot的方法。如果允许存在重复值，则可能中点位置无法确定，此时可以削减左右边界（因为此时二者一定相等），直到可以确定中点相对位置。

## 82. Remove Duplicates from Sorted List II

Given a sorted linked list, delete all nodes that have duplicate numbers, leaving only *distinct* numbers from the original list.

For example,  
Given 1->2->3->3->4->4->5, return 1->2->5.  
Given 1->1->1->2->3, return 2->3.

思路：变化版有序链表去重，必要时需要为链表增加列表头节点

## 83. Remove Duplicates from Sorted List

Given a sorted linked list, delete all duplicates such that each element appear only *once*.

For example,  
Given 1->1->2, return 1->2.  
Given 1->1->2->3->3, return 1->2->3.

思路：标准的有序链表去重

## 84. Largest Rectangle in Histogram

Given *n* non-negative integers representing the histogram's bar height where the width of each bar is 1, find the area of largest rectangle in the histogram.



Above is a histogram where width of each bar is 1, given height = [2,1,5,6,2,3].



The largest rectangle is shown in the shaded area, which has area = 10 unit.

For example,  
Given heights = [2,1,5,6,2,3],  
return 10.

## 85. Maximal Rectangle

Given a 2D binary matrix filled with 0's and 1's, find the largest rectangle containing only 1's and return its area.

For example, given the following matrix:

1 0 1 0 0

1 0 1 1 1

1 1 1 1 1

1 0 0 1 0

Return 6.

## 86. Partition List

Given a linked list and a value *x*, partition it such that all nodes less than *x* come before nodes greater than or equal to *x*.

You should preserve the original relative order of the nodes in each of the two partitions.

For example,  
Given 1->4->3->2->5->2 and *x* = 3,  
return 1->2->2->4->3->5.

思路：建两个列表即可，顺序放置节点，最后接在一起即可

## 87. Scramble String

Given a string *s1*, we may represent it as a binary tree by partitioning it to two non-empty substrings recursively.

Below is one possible representation of *s1* = "great":

great

/ \

gr eat

/ \ / \

g r e at

/ \

a t

To scramble the string, we may choose any non-leaf node and swap its two children.

For example, if we choose the node "gr" and swap its two children, it produces a scrambled string "rgeat".

rgeat

/ \

rg eat

/ \ / \

r g e at

/ \

a t

We say that "rgeat" is a scrambled string of "great".

Similarly, if we continue to swap the children of nodes "eat" and "at", it produces a scrambled string "rgtae".

rgtae

/ \

rg tae

/ \ / \

r g ta e

/ \

t a

We say that "rgtae" is a scrambled string of "great".

Given two strings *s1* and *s2* of the same length, determine if *s2* is a scrambled string of *s1*.

思路：不明所以，为什么列举了二叉树，难道用散列表查找不是更好吗？

## 88. Merge Sorted Array

Given two sorted integer arrays *nums1* and *nums2*, merge *nums2* into *nums1* as one sorted array.

**Note:**  
You may assume that *nums1* has enough space (size that is greater or equal to *m* + *n*) to hold additional elements from *nums2*. The number of elements initialized in *nums1* and *nums2* are *m* and *n* respectively.

思路：标准的有序向量归并算法，根据题设，从后向前排序为宜，因为nums1后段为空。

## 89. Gray Code

The gray code is a binary numeral system where two successive values differ in only one bit.

Given a non-negative integer *n* representing the total number of bits in the code, print the sequence of gray code. A gray code sequence must begin with 0.

For example, given *n* = 2, return [0,1,3,2]. Its gray code sequence is:

00 - 0

01 - 1

11 - 3

10 - 2

**Note:**  
For a given *n*, a gray code sequence is not uniquely defined.

For example, [0,2,3,1] is also a valid gray code sequence according to the above definition.

For now, the judge is able to judge based on one instance of gray code sequence. Sorry about that.

思路：抛开格雷码本身的生成公式。一般的，本题可以使用动态规划，从n=1位开始逐渐增加二进制位数。增加的二进制位全部取1，其他位依次逆向取所有之前的格雷码。

## 90. Subsets II

Given a collection of integers that might contain duplicates, ***nums***, return all possible subsets (the power set).

**Note:** The solution set must not contain duplicate subsets.

For example,  
If ***nums*** = [1,2,2], a solution is:

[

[2],

[1],

[1,2,2],

[2,2],

[1,2],

[]

]

思路：先解释题意——power set指的就是一般意义上的集合set，每个元素都有出现或不出现两种可能。集合不考虑元素顺序，为了避免重复集合，需要对重复元素进行特殊对待。首先必须排序，以找到重复元素或将其聚集以处理。重复元素（重复n次）和一般元素的区别是，重复元素有出现若干次和不出现的n+1种可能。

## 91. Decode Ways

A message containing letters from A-Z is being encoded to numbers using the following mapping:

'A' -> 1

'B' -> 2

...

'Z' -> 26

Given an encoded message containing digits, determine the total number of ways to decode it.

For example,  
Given encoded message "12", it could be decoded as "AB" (1 2) or "L" (12).

The number of ways decoding "12" is 2.

思路：使用动态规划，从字符串的一端逐字符加长子串范围进行计算。根据当前字符的数值是否为0，和当前2个字符的数值是否大于26，来计算当前长度子串的解码可能性。这种统计某种属性参数的问题，是可以用动态规划来解决的。如果要全部罗列出各种输出结果，则使用动态规划并不简单，使用分而治之的递归法可能更好。

## 92. Reverse Linked List II

Reverse a linked list from position *m* to *n*. Do it in-place and in one-pass.

For example:  
Given 1->2->3->4->5->NULL, *m* = 2 and *n* = 4,

return 1->4->3->2->5->NULL.

**Note:**  
Given *m*, *n* satisfy the following condition:  
1 ≤ *m* ≤ *n* ≤ length of list.

思路：将一定范围的链表逆序，从起始节点开始依次将下一个节点前置，如此循环并顺序前进一次。

## 93. Restore IP Addresses

Given a string containing only digits, restore it by returning all possible valid IP address combinations.

For example:  
Given "25525511135",

return ["255.255.11.135", "255.255.111.35"]. (Order does not matter)

思路：

## 94. Binary Tree Inorder Traversal

Given a binary tree, return the *inorder* traversal of its nodes' values.

For example:  
Given binary tree [1,null,2,3],

1

\

2

/

3

return [1,3,2].

**Note:** Recursive solution is trivial, could you do it iteratively?

思路：二叉树中序遍历标准算法

## 95. Unique Binary Search Trees II

Given an integer *n*, generate all structurally unique **BST's** (binary search trees) that store values 1...*n*.

For example,  
Given *n* = 3, your program should return all 5 unique BST's shown below.

1 3 3 2 1

\ / / / \ \

3 2 1 1 3 2

/ / \ \

2 1 2 3

思路：无法直接使用由底到上的动态规划，应该使用分而治之的递归算法，将求整棵树(1~n)变成求某节点左右子树的范围内求树(lo~hi)问题，然后逐层递归即可。

## 96. Unique Binary Search Trees

Given *n*, how many structurally unique **BST's** (binary search trees) that store values 1...*n*?

For example,  
Given *n* = 3, there are a total of 5 unique BST's.

1 3 3 2 1

\ / / / \ \

3 2 1 1 3 2

/ / \ \

2 1 2 3

思路：动态规划，根节点为界，其左右子树的节点数目变化，左右子树的种数可由之前的计算获得。

## 97. Interleaving String

Given *s1*, *s2*, *s3*, find whether *s3* is formed by the interleaving of *s1* and *s2*.

For example,  
Given:  
*s1* = "aabcc",  
*s2* = "dbbca",

When *s3* = "aadbbcbcac", return true.  
When *s3* = "aadbbbaccc", return false.

## 98. Validate Binary Search Tree

Given a binary tree, determine if it is a valid binary search tree (BST).

Assume a BST is defined as follows:

* The left subtree of a node contains only nodes with keys **less than** the node's key.
* The right subtree of a node contains only nodes with keys **greater than** the node's key.
* Both the left and right subtrees must also be binary search trees.

**Example 1:**

2

/ \

1 3

Binary tree [2,1,3], return true.

**Example 2:**

1

/ \

2 3

Binary tree [1,2,3], return false.

思路：使用中序遍历的顺序性验证，或者使用基于树结构的递归验证，即对每个节点分别验证其是否在一个范围区间，递归返回条件是该节点的两个孩子也分别满足。

注意：BST的条件不仅仅是左右孩子小于或大于当前节点，还包括不能超过当前节点的父辈们的限制。

## 99. Recover Binary Search Tree

Two elements of a binary search tree (BST) are swapped by mistake.

Recover the tree without changing its structure.

**Note:**  
A solution using O(*n*) space is pretty straight forward. Could you devise a constant space solution?

思路：使用迭代或递归的中序遍历均需O(n)的空间复杂度。如果只占用线性空间复杂度，必须解决节点无右子树时可以找到直接后继的问题。一种办法是每次深入左子树时，将左子树的最右侧节点的右子树设为当前节点（Morris-traversal），另一种就必须依赖节点数据结构存有父节点的信息便于追溯前述的中序遍历后继。

## 100. Same Tree

Given two binary trees, write a function to check if they are the same or not.

Two binary trees are considered the same if they are structurally identical and the nodes have the same value.

**Example 1:**

**Input:** 1 1

/ \ / \

2 3 2 3

[1,2,3], [1,2,3]

**Output:** true

**Example 2:**

**Input:** 1 1

/ \

2 2

[1,2], [1,null,2]

**Output:** false

**Example 3:**

**Input:** 1 1

/ \ / \

2 1 1 2

[1,2,1], [1,1,2]

**Output:** false

思路：使用递归法，每一层递归检验当前节点位置二者是否相同，递归的返回条件是当前节点的左右孩子位置二者均相同。可以通过栈改成一般式迭代，但使用遍历是错误的，因为只能检验按照遍历顺序的节点是否值相同而无法检验两树的实际结构是否相同。