Peking2:

其实算是我自己的标准了。可能主观因素比较大。  
在没有见过的情况下  
难度5的话，我认为除了那些ACMers, 大家都会跪  
难度4的话，一般人会有思路，但是很难写出bug free的code  
难度3的话，比较适中，经过一定锻炼的人，只要小心应该可以写出完美代码  
难度2的话，稍微有一点小算法，即使没怎么准备过，也应该没啥问题  
难度1就是水题了  
当然难度跟出现的频率也有关系，频率高的话，难度系数会降低，因为人所众知的题，  
难度自然要降低了。

Freq: 5 4 3 2 1 N/A

Count: 16 18 37 39 22 16

Diff: 5 4 3 2 1 N/A

Count: 12 38 45 23 14 13

<http://www.zzsec.org/2013/04/leetcode-statistics/>

leetcode题目统计

April 26 2013 , [coding](http://www.zzsec.org/categories.html#coding-ref)

**前言**

这几天已经习惯依照Leetcoder来做题了，今天一刷新，就看到页面404了。还好我还有一个打开着的页面。在博客上备份一份。

原文地址<http://leetcode.cloudfoundry.com/>

### Leetcode Questions

| **ID** | **QUESTION** | **DIFFICULTY** | **FREQENCY** | **DATA STRUCTURES** | **ALGORITHMS** |
| --- | --- | --- | --- | --- | --- |
| 1 | Two Sum | 2 | 5 | array, set | sort, two pointers |
| 2 | Add Two Numbers | 3 | 4 | linked list | two pointers, math |
| 3 | Longest Substring Without Repeating Characters | 3 | 2 | string, hashtable | two pointers |
| 4 | Median of Two Sorted Arrays | 5 | 3 | array | binary search |
| 5 | Longest Palindromic Substring | 4 | 2 | string |  |
| 6 | ZigZag Conversion | 3 | 1 | string |  |
| 7 | Reverse Integer | 2 | 3 |  | math |
| 8 | String to Integer (atoi) | 2 | 5 | string | math |
| 9 | Palindrome Number | 2 | 2 |  | math |
| 10 | Regular Expression Matching | 5 | 3 | string | recursion, dp |
| 11 | Container With Most Water | 3 | 2 | array | two pointers |
| 12 | Integer to Roman | 3 | 4 |  | math |
| 13 | Roman to Integer | 2 | 4 |  | math |
| 14 | Longest Common Prefix | 2 | 1 | string |  |
| 15 | 3Sum | 3 | 5 | array | two pointers |
| 16 | 3Sum Closest | 3 | 1 | array | two pointers |
| 17 | Letter Combinations of a Phone Number | 3 | 3 | string | dfs |
| 18 | 4Sum | 3 | 2 | array |  |
| 19 | Remove Nth Node From End of List | 2 | 3 | linked list | two pointers |
| 20 | Valid Parentheses | 2 | 5 | string | stack |
| 21 | Merge Two Sorted Lists | 2 | 5 | linked list | sort, two pointers, merge |
| 22 | Generate Parentheses | 3 | 4 | string | dfs |
| 23 | Merge k Sorted Lists | 3 | 4 | linked list, heap | sort, two pointers, merge |
| 24 | Swap Nodes in Pairs | 2 | 4 | linked list |  |
| 25 | Reverse Nodes in k-Group | 4 | 2 | linked list | recursion, two pointers |
| 26 | Remove Duplicates from Sorted Array | 1 | 3 | array | two pointers |
| 27 | Remove Element | 1 | 4 | array | two pointers |
| 28 | Implement strStr() | 4 | 5 | string | two pointers, KMP, rolling hash |
| 29 | Divide Two Integers | 4 | 3 |  | binary search, math |
| 30 | Substring with Concatenation of All Words | 3 | 1 | string | two pointers |
| 31 | Next Permutation | 5 | 2 | array | permutation |
| 32 | Longest Valid Parentheses | 4 | 1 | string | dp |
| 33 | Search in Rotated Sorted Array | 4 | 3 | array | binary search |
| 34 | Search for a Range | 4 | 3 | array | binary search |
| 35 | Search Insert Position | 2 | 2 | array |  |
| 36 | Valid Sudoku | 2 | 2 | array |  |
| 37 | Sudoku Solver | 4 | 2 | array | dfs |
| 38 | Count and Say | 2 | 2 | string | two pointers |
| 39 | Combination Sum | 3 | 3 | array | combination |
| 40 | Combination Sum II | 4 | 2 | array | combination |
| 41 | First Missing Positive | 5 | 2 | array | sort |
| 42 | Trapping Rain Water | 4 | 2 | array | two pointers, stack |
| 43 | Multiply Strings | 4 | 3 | string | two pointers |
| 44 | Wildcard Matching | 5 | 3 | string | recursion, dp, greedy |
| 45 | Jump Game II | 4 | 2 | array |  |
| 46 | Permutations | 3 | 4 | array | permutation |
| 47 | Permutations II | 4 | 2 | array | permutation |
| 48 | Rotate Image | 4 | 2 | array |  |
| 49 | Anagrams | 3 | 4 | string, hashtable |  |
| 50 | Pow(x, n) | 3 | 5 |  | binary search, math |
| 51 | N-Queens | 4 | 3 | array | dfs |
| 52 | N-Queens II | 4 | 3 | array | dfs |
| 53 | Maximum Subarray | 3 | 3 | array | dp |
| 54 | Spiral Matrix | 4 | 2 | array |  |
| 55 | Jump Game | 3 | 2 | array |  |
| 56 | Merge Intervals | 4 | 5 | array, linked list, red-black tree | sort, merge |
| 57 | Insert Interval | 4 | 5 | array, linked list, red-black tree | sort, merge |
| 58 | Length of Last Word | 1 | 1 | string |  |
| 59 | Spiral Matrix II | 3 | 2 | array |  |
| 60 | Permutation Sequence | 5 | 1 |  | permutation, math |
| 61 | Rotate List | 3 | 2 | linked list | two pointers |
| 62 | Unique Paths | 2 | 3 | array | dp |
| 63 | Unique Paths II | 3 | 3 | array | dp |
| 64 | Minimum Path Sum | 3 | 3 | array | dp |
| 65 | Valid Number | 2 | 5 | string | math |
| 66 | Plus One | 1 | 2 | array | math |
| 67 | Add Binary | 2 | 4 | string | two pointers, math |
| 68 | Text Justification | 4 | 2 | string |  |
| 69 | Sqrt(x) | 4 | 4 |  | binary search |
| 70 | Climbing Stairs | 2 | 5 |  | dp |
| 71 | Simplify Path | 3 | 1 | string | stack |
| 72 | Edit Distance | 4 | 3 | string | dp |
| 73 | Set Matrix Zeroes | 3 | 5 | array |  |
| 74 | Search a 2D Matrix | 3 | 3 | array | binary search |
| 75 | Sort Colors | 4 | 2 | array | sort, two pointers |
| 76 | Minimum Window Substring | 4 | 2 | string | two pointers |
| 77 | Combinations | 3 | 4 |  | combination |
| 78 | Subsets | 3 | 4 | array | recursion, combination |
| 79 | Word Search | 3 | 4 | array | dfs |
| 80 | Remove Duplicates from Sorted Array II | 2 | 2 | array | two pointers |
| 81 | Search in Rotated Sorted Array II | 5 | 3 | array | binary search |
| 82 | Remove Duplicates from Sorted List II | 3 | 3 | linked list | recursion, two pointers |
| 83 | Remove Duplicates from Sorted List | 1 | 3 | linked list |  |
| 84 | Largest Rectangle in Histogram | 5 | 2 | array | stack |
| 85 | Maximal Rectangle | 5 | 1 | array | dp, stack |
| 86 | Partition List | 3 | 3 | linked list | two pointers |
| 87 | Scramble String | 5 | 2 | string | recursion, dp |
| 88 | Merge Sorted Array | 2 | 5 | array | two pointers, merge |
| 89 | Gray Code | 4 | 2 |  | combination |
| 90 | Subsets II | 4 | 2 | array | recursion, combination |
| 91 | Decode Ways | 3 | 4 | string | recursion, dp |
| 92 | Reverse Linked List II | 3 | 2 | linked list | two pointers |
| 93 | Restore IP Addresses | 3 | 3 | string | dfs |
| 94 | Binary Tree Inorder Traversal | 4 | 3 | tree, hashtable | recursion, morris, stack |
| 95 | Unique Binary Search Trees II | 4 | 1 | tree | dp, dfs |
| 96 | Unique Binary Search Trees | 3 | 1 | tree | dp |
| 97 | Interleaving String | 5 | 2 | string | recursion, dp |
| 98 | Validate Binary Search Tree | 3 | 5 | tree | dfs |
| 99 | Recover Binary Search Tree | 4 | 2 | tree | dfs |
| 100 | Same Tree | 1 | 1 | tree | dfs |
| 101 | Symmetric Tree | 1 | 2 | tree | dfs |
| 102 | Binary Tree Level Order Traversal | 3 | 4 | tree | bfs |
| 103 | Binary Tree Zigzag Level Order Traversal | 4 | 3 | queue, tree | bfs, stack |
| 104 | Maximum Depth of Binary Tree | 1 | 1 | tree | dfs |
| 105 | Construct Binary Tree from Preorder and Inorder Tr | 3 | 3 | array, tree | dfs |
| 106 | Construct Binary Tree from Inorder and Postorder T | 3 | 3 | array, tree | dfs |
| 107 | Binary Tree Level Order Traversal II | 3 | 1 | tree | bfs |
| 108 | Convert Sorted Array to Binary Search Tree | 2 | 3 | tree | dfs |
| 109 | Convert Sorted List to Binary Search Tree | 4 | 3 | linked list | recursion, two pointers |
| 110 | Balanced Binary Tree | 1 | 2 | tree | dfs |
| 111 | Minimum Depth of Binary Tree | 1 | 1 | tree | dfs |
| 112 | Path Sum | 1 | 3 | tree | dfs |
| 113 | Path Sum II | 2 | 2 | tree | dfs |
| 114 | Flatten Binary Tree to Linked List | 3 | 3 | tree | recursion, stack |
| 115 | Distinct Subsequences | 4 | 2 | string | dp |
| 116 | Populating Next Right Pointers in Each Node | 3 | 3 | tree | dfs |
| 117 | Populating Next Right Pointers in Each Node II | 4 | 2 | tree | dfs |
| 118 | Pascal's Triangle | 2 | 1 | array |  |
| 119 | Pascal's Triangle II | 2 | 1 | array |  |
| 120 | Triangle | 3 | 1 | array | dp |
| 121 | Best Time to Buy and Sell Stock | 2 | 1 | array | dp |
| 122 | Best Time to Buy and Sell Stock II | 3 | 1 | array | greedy |
| 123 | Best Time to Buy and Sell Stock III | 4 | 1 | array | dp |
| 124 | Binary Tree Maximum Path Sum | 4 | 2 | tree | dfs |
| 125 | Valid Palindrome | 2 | 5 | string | two pointers |
| 126 | Word Ladder II | 1 | 1 |  |  |
| 127 | Word Ladder | 3 | 5 | graph | bfs, shortest path |
| 128 | Longest Consecutive Sequence | 4 | 3 | array |  |
| 129 | Sum Root to Leaf Numbers | 2 | 4 | tree | dfs |
| 130 | Surrounded Regions | 4 | 3 | array | bfs, dfs |
| 131 | Palindrome Partitioning | 3 | 4 | string | dfs |
| 132 | Palindrome Partitioning II | 4 | 3 | string | dp |

**Leetcode Questions**

| **ID** | **QUESTION** | **DIFFICULTY** | **FREQENCY** | **DATA STRUCTURES** | **ALGORITHMS** |
| --- | --- | --- | --- | --- | --- |
| 1 | Two Sum | 2 | 5 | array, set | sort, two pointers |
| 2 | Median of Two Sorted Arrays | 5 | 3 | array | binary search |
| 3 | Longest Substring Without Repeating Characters | 3 | 2 | string, hashtable | two pointers |
| 4 | Add Two Numbers | 3 | 4 | linked list | two pointers, math |
| 5 | Longest Palindromic Substring | 4 | 2 | string |  |
| 6 | ZigZag Conversion | 3 | 1 | string |  |
| 7 | Reverse Integer | 2 | 3 |  | math |
| 8 | String to Integer (atoi) | 2 | 5 | string | math |
| 9 | Palindrome Number | 2 | 2 |  | math |
| 10 | Regular Expression Matching | 5 | 3 | string | recursion, dp |
| 11 | Container With Most Water | 3 | 2 | array | two pointers |
| 12 | Integer to Roman | 3 | 4 |  | math |
| 13 | Roman to Integer | 2 | 4 |  | math |
| 14 | Longest Common Prefix | 2 | 1 | string |  |
| 15 | 3Sum | 3 | 5 | array | two pointers |
| 16 | 3Sum Closest | 3 | 1 | array | two pointers |
| 17 | Letter Combinations of a Phone Number | 3 | 3 | string | dfs |
| 18 | 4Sum | 3 | 2 | array |  |
| 19 | Remove Nth Node From End of List | 2 | 3 | linked list | two pointers |
| 20 | Valid Parentheses | 2 | 5 | string | stack |
| 21 | Merge Two Sorted Lists | 2 | 5 | linked list | sort, two pointers, merge |
| 22 | Generate Parentheses | 3 | 4 | string | dfs |
| 23 | Merge k Sorted Lists | 3 | 4 | linked list, heap | sort, two pointers, merge |
| 24 | Swap Nodes in Pairs | 2 | 4 | linked list |  |
| 25 | Reverse Nodes in k-Group | 4 | 2 | linked list | recursion, two pointers |
| 26 | Remove Duplicates from Sorted Array | 1 | 3 | array | two pointers |
| 27 | Remove Element | 1 | 4 | array | two pointers |
| 28 | Implement strStr() | 4 | 5 | string | two pointers, KMP, rolling hash |
| 29 | Divide Two Integers | 4 | 3 |  | binary search, math |
| 30 | Substring with Concatenation of All Words | 3 | 1 | string | two pointers |
| 31 | Next Permutation | 5 | 2 | array | permutation |
| 32 | Longest Valid Parentheses | 4 | 1 | string | dp |
| 33 | Search in Rotated Sorted Array | 4 | 3 | array | binary search |
| 34 | Search for a Range | 4 | 3 | array | binary search |
| 35 | Search Insert Position | 2 | 2 | array |  |
| 36 | Valid Sudoku | 2 | 2 | array |  |
| 37 | Sudoku Solver | 4 | 2 | array | dfs |
| 38 | Count and Say | 2 | 2 | string | two pointers |
| 39 | Combination Sum | 3 | 3 | array | combination |
| 40 | Combination Sum II | 4 | 2 | array | combination |
| 41 | First Missing Positive | 5 | 2 | array | sort |
| 42 | Trapping Rain Water | 4 | 2 | array | two pointers, stack |
| 43 | Multiply Strings | 4 | 3 | string | two pointers |
| 44 | Wildcard Matching | 5 | 3 | string | recursion, dp, greedy |
| 45 | Jump Game II | 4 | 2 | array |  |
| 46 | Permutations | 3 | 4 | array | permutation |
| 47 | Permutations II | 4 | 2 | array | permutation |
| 48 | Rotate Image | 4 | 2 | array |  |
| 49 | Anagrams | 3 | 4 | string, hashtable |  |
| 50 | Pow(x, n) | 3 | 5 |  | binary search, math |
| 51 | N-Queens | 4 | 3 | array | dfs |
| 52 | N-Queens II | 4 | 3 | array | dfs |
| 53 | Maximum Subarray | 3 | 3 | array | dp |
| 54 | Spiral Matrix | 4 | 2 | array |  |
| 55 | Jump Game | 3 | 2 | array |  |
| 56 | Merge Intervals | 4 | 5 | array, linked list, red-black tree | sort, merge |
| 57 | Insert Interval | 4 | 5 | array, linked list, red-black tree | sort, merge |
| 58 | Length of Last Word | 1 | 1 | string |  |
| 59 | Spiral Matrix II | 3 | 2 | array |  |
| 60 | Permutation Sequence | 5 | 1 |  | permutation, math |
| 61 | Rotate List | 3 | 2 | linked list | two pointers |
| 62 | Unique Paths | 2 | 3 | array | dp |
| 63 | Unique Paths II | 3 | 3 | array | dp |
| 64 | Minimum Path Sum | 3 | 3 | array | dp |
| 65 | Valid Number | 2 | 5 | string | math |
| 66 | Plus One | 1 | 2 | array | math |
| 67 | Add Binary | 2 | 4 | string | two pointers, math |
| 68 | Text Justification | 4 | 2 | string |  |
| 69 | Sqrt(x) | 4 | 4 |  | binary search |
| 70 | Climbing Stairs | 2 | 5 |  | dp |
| 71 | Simplify Path | 3 | 1 | string | stack |
| 72 | Edit Distance | 4 | 3 | string | dp |
| 73 | Set Matrix Zeroes | 3 | 5 | array |  |
| 74 | Search a 2D Matrix | 3 | 3 | array | binary search |
| 75 | Sort Colors | 4 | 2 | array | sort, two pointers |
| 76 | Minimum Window Substring | 4 | 2 | string | two pointers |
| 77 | Combinations | 3 | 4 |  | combination |
| 78 | Subsets | 3 | 4 | array | recursion, combination |
| 79 | Word Search | 3 | 4 | array | dfs |
| 80 | Remove Duplicates from Sorted Array II | 2 | 2 | array | two pointers |
| 81 | Search in Rotated Sorted Array II | 5 | 3 | array | binary search |
| 82 | Remove Duplicates from Sorted List II | 3 | 3 | linked list | recursion, two pointers |
| 83 | Remove Duplicates from Sorted List | 1 | 3 | linked list |  |
| 84 | Largest Rectangle in Histogram | 5 | 2 | array | stack |
| 85 | Maximal Rectangle | 5 | 1 | array | dp, stack |
| 86 | Partition List | 3 | 3 | linked list | two pointers |
| 87 | Scramble String | 5 | 2 | string | recursion, dp |
| 88 | Merge Sorted Array | 2 | 5 | array | two pointers, merge |
| 89 | Gray Code | 4 | 2 |  | combination |
| 90 | Subsets II | 4 | 2 | array | recursion, combination |
| 91 | Decode Ways | 3 | 4 | string | recursion, dp |
| 92 | Reverse Linked List II | 3 | 2 | linked list | two pointers |
| 93 | Restore IP Addresses | 3 | 3 | string | dfs |
| 94 | Binary Tree Inorder Traversal | 4 | 3 | tree, hashtable | recursion, morris, stack |
| 95 | Unique Binary Search Trees II | 4 | 1 | tree | dp, dfs |
| 96 | Unique Binary Search Trees | 3 | 1 | tree | dp |
| 97 | Interleaving String | 5 | 2 | string | recursion, dp |
| 98 | Validate Binary Search Tree | 3 | 5 | tree | dfs |
| 99 | Recover Binary Search Tree | 4 | 2 | tree | dfs |
| 100 | Same Tree | 1 | 1 | tree | dfs |
| 101 | Symmetric Tree | 1 | 2 | tree | dfs |
| 102 | Binary Tree Level Order Traversal | 3 | 4 | tree | bfs |
| 103 | Binary Tree Zigzag Level Order Traversal | 4 | 3 | queue, tree | bfs, stack |
| 104 | Maximum Depth of Binary Tree | 1 | 1 | tree | dfs |
| 105 | Construct Binary Tree from Preorder and Inorder Tr | 3 | 3 | array, tree | dfs |
| 106 | Construct Binary Tree from Inorder and Postorder T | 3 | 3 | array, tree | dfs |
| 107 | Binary Tree Level Order Traversal II | 3 | 1 | tree | bfs |
| 108 | Convert Sorted Array to Binary Search Tree | 2 | 3 | tree | dfs |
| 109 | Convert Sorted List to Binary Search Tree | 4 | 3 | linked list | recursion, two pointers |
| 110 | Balanced Binary Tree | 1 | 2 | tree | dfs |
| 111 | Minimum Depth of Binary Tree | 1 | 1 | tree | dfs |
| 112 | Path Sum | 1 | 3 | tree | dfs |
| 113 | Path Sum II | 2 | 2 | tree | dfs |
| 114 | Flatten Binary Tree to Linked List | 3 | 3 | tree | recursion, stack |
| 115 | Distinct Subsequences | 4 | 2 | string | dp |
| 116 | Populating Next Right Pointers in Each Node | 3 | 3 | tree | dfs |
| 117 | Populating Next Right Pointers in Each Node II | 4 | 2 | tree | dfs |
| 118 | Pascal's Triangle | 2 | 1 | array |  |
| 119 | Pascal's Triangle II | 2 | 1 | array |  |
| 120 | Triangle | 3 | 1 | array | dp |
| 121 | Best Time to Buy and Sell Stock | 2 | 1 | array | dp |
| 122 | Best Time to Buy and Sell Stock II | 3 | 1 | array | greedy |
| 123 | Best Time to Buy and Sell Stock III | 4 | 1 | array | dp |
| 124 | Binary Tree Maximum Path Sum | 4 | 2 | tree | dfs |
| 125 | Valid Palindrome | 2 | 5 | string | two pointers |
| 126 | Word Ladder II | 1 | 1 |  |  |
| 127 | Word Ladder | 3 | 5 | graph | bfs, shortest path |
| 128 | Longest Consecutive Sequence | 4 | 3 | array |  |
| 129 | Sum Root to Leaf Numbers | 2 | 4 | tree | dfs |
| 130 | Surrounded Regions | 4 | 3 | array | bfs, dfs |
| 131 | Palindrome Partitioning | 3 | 4 | string | dfs |
| 132 | Palindrome Partitioning II | 4 | 3 | string | dp |

<http://www.cnblogs.com/feiling/p/3269195.html>

Here is a [difficulty and frequency distribution chart](https://docs.google.com/spreadsheet/pub?key=0Aqt--%20wSNYfuxdGxQWVFsOGdVVWxQRlNUVXZTdEpOeEE&output=html) for each problem (which I got from the Internet and is very useful).

### Dynamic Programming

* Edit Distance
* Maximum Subarray
* Minimum Path Sum
* Unique Paths
* Unique Paths II
* Longest Palindromic Substring
* Interleaving String
* Triangle
* Distinct Subsequences
* Decode Ways
* Palindrome Partitioning II
* Maximal Rectangle

### Recursion

* N-Queens
* N-Queens II
* Balanced Binary Tree
* Binary Tree Inorder Traversal
* Binary Tree Maximum Path Sum
* Convert Sorted Array to Binary Search Tree
* Convert Sorted List to Binary Search Tree
* Flatten Binary Tree to Linked List
* Maximum Depth of Binary Tree
* Minimum Depth of Binary Tree
* Path Sum
* Permutations
* Permutations II
* Populating Next Right Pointers in Each Node
* Pow(x, n)
* Same Tree
* Subsets
* Sum Root to Leaf Numbers
* Swap Nodes in Pairs
* Symmetric Tree
* Valid Palindrome
* Validate Binary Search Tree
* Restore IP Addresses
* Combinations
* Interleaving String (dp is the best)
* Combination Sum II
* Letter Combinations of a Phone Numbers
* Word Search
* Construct Binary Tree from Inorder and Postorder Traversal
* Construct Binary Tree from Preorder and Inorder Traversal
* Generate Parentheses
* Surrounded Regions (runtime error)
* Palindrome Partitioning
* Combination Sum
* Sudoku Solver
* Unique Binary Search Trees II

### Binary Search

* Search Insert Position
* Search a 2D Matrix
* Search for a Range
* Search in Rotated Sorted Array
* Sqrt(x)

### Sequence

* Container With Most Water
* Count and Say
* First Missing Positive
* Implement strStr()
* Jump Game
* Jump Game II
* Length of Last Word
* Longest Common Prefix
* Longest Substring Without Repeating Characters
* Merge Sorted Array
* Palindrome Number
* Plus One
* Remove Duplicates from Sorted Array
* Remove Duplicates from Sorted Array II
* Remove Element
* Reverse Integer
* Search in Rotated Sorted Array II
* Sort Colors
* Two Sum
* 3Sum
* 3Sum Closest
* 4Sum
* Add Binary
* Longest Palindromic Substring
* Next Permutation
* Longest Valid Parentheses
* Climbing Stairs
* Permutation Sequence
* Simplify Path
* String to Integer (atoi)
* Minimum Window Substring
* Longest Consecutive Sequence
* Trapping Rain Water
* Valid Number

### Linked List

* Add Two Numbers
* Convert Sorted List to Binary Search Tree
* Merge Two Sorted Lists
* Partition List
* Remove Duplicates from Sorted List
* Remove Duplicates from Sorted List II
* Remove Nth Node From End of List
* Reverse Linked List II
* Reverse Nodes in k-Group
* Rotate List
* Swap Nodes in Pairs

### Stack

* Binary Tree Inorder Traversal
* Binary Tree Level Order Traversal II
* Valid Parentheses

### Queue

* Binary Tree Level Order Traversal
* Binary Tree Level Order Traversal II
* Populating Next Right Pointers in Each Node II
* Symmetric Tree
* Surrounded Regions
* Word Ladder

### Tree

* Balanced Binary Tree
* Binary Tree Inorder Traversal
* Binary Tree Level Order Traversal
* Binary Tree Level Order Traversal II
* Binary Tree Maximum Path Sum
* Convert Sorted Array to Binary Search Tree
* Convert Sorted List to Binary Search Tree
* Flatten Binary Tree to Linked List
* Maximum Depth of Binary Tree
* Minimum Depth of Binary Tree
* Path Sum
* Same Tree
* Sum Root to Leaf Numbers
* Symmetric Tree
* Validate Binary Search Tree

### zz. 面试总结 from peking2

首先声明一下，这里的面试题主要所指数据结构和算法的题目，题目的分析集中在Leetcode上面的题目上。  
  
我认为一道面试题由以下几个方面组成的

1. Question
2. Data structure in question
3. Data structure in solution
4. Algorithm in solution
5. Coding

**题目：**非常关键，一个题目通常有一些相应的变形题目，同一个题目可能有不同的要求。比如时间复杂度，空间复杂度的要求，比如recursive, iterative的要求。而根据题目的变形与要求，可能会极大的影响到你能够采取的数据结构和算法。  
  
**问题中的数据机构：**问题中有可能带数据结构，有可能没有数据结构，有可能是可以自定义数据结构  
  
**解决方案中的数据结构：**可以是in-place的，也就是利用已有的数据结构，也可能是创建新的数据结构。新的数据结构跟已有的数据结构没有必然的联系，而很多问题都是一题多解，可能采取不同的数据结构。  
  
**算法：**一般来说，当解决方案中的数据结构确定以后，算法也就确定了。同样，一旦解决方案的算法确定，相应的数据结构也就确定了。这两个没有先后的关系，但解决方案中的数据结构和算法具有非常紧密的联系。  
  
**代码：**非常关键。代码就是解决方案的数据结构和算法的实现了。目前来看，题目，数据结构和算法在面试中出现的类型比较固定，因此代码的好坏则是拉开面试者水平的一个有效手段。这也是为什么F，G如此看中代码的质量了。我发现上面几点比较容易突击，但是写代码的功力还是需要实打实的积累的。  
  
总结面试题目的关键就是要把面试题目进行分类之后分析。由于面试题目由以上几个部分组成并且混杂在一起，因此怎样合理的分类就变得非常的困难。其实Careercup150的分类就比较好，它是这样进行分类的。  
数据结构：Arrays and Strings, Linked Lists, Stacks and Queues, Trees and Graphs  
算法：Bit Manipulation, Mathematics and Probability, Recursion and Dynamic Programming, Sorting and Searching  
但是我感觉这样分类比较适合初级阶段学习，并不适合系统地对面试题目进行分析。我其实目前也没有什么特别好的idea，想跟着感觉先来，可能慢慢就能悟出更多了。  
  
1. 首先算法要简洁，算法不简洁代码好不了，这是根本。  
2. 对惯用的代码要形成习惯和模式，基本上大家都这么写  
比如一个array倒置[1,2,3]变为[3,2,1]，基本上都会用while, 而我以前由于没有练习过，当场用的for，就很难看。  
3. 每一句代码都是必要的。有些代码虽说放在那里无关大局，但是完全可以删掉，就显得冗余难看  
4. 代码要模块化，如果一段代码会重复使用要写子函数。或者一部分功能自成一体，最好也写子函数  
  
简称two pointers吧。大概把分类粗略的搞了一遍（<http://leetcode.cloudfoundry.com/>), 发现利用two pointers解决的题目数量很大。two pointers我指的是一类题，而不一定是真正的two pointers, 比如可能是three pointers, 也可能不是pointer， 而是index。这类题基本上就是发生在array, string, linked list这三种数据结构上，是一种基本的算法和编程技巧，同样超高频率的出现，可以说是面试必遇的题。  
two pointers常常和其他的算法混杂起来出现。比如binary search本身也可以归类为two pointers的。如果这样算的话，Leetcode上边1/4的题目都跟它相关。因此，two pointers是必须熟练掌握的基本编程技巧。  
  
Two pointers大概分三种类型  
1. 两个pointers从头往后走：感觉绝大多数的linked list的题目都涉及到这个操作，当然还有array。这类题目很多时候又可以称为sliding window。

* Implement strStr()
* Longest Substring Without Repeating Characters
* Minimum Window Substring
* Remove Duplicates from Sorted Array & II
* Remove Duplicates from Sorted List & II
* Remove Element
* Remove Nth Node From End of List
* Reverse Linked Llist II
* Rotate List
* Substring with Concatenation of All Words
* Swap Nodes in Pairs

2. 两个pointers从两头往中间走：一般面试出现的的都是singly linked list, 因此这类题主要是array题。

* 3Sum
* 3Sum Closest
* 4Sum
* Container With Most Water
* Sort Colors
* Trapping Rain Water
* Two Sum
* Binary search (will discuss it in a separate section)

3. 两个pointers控制两个不同的数组或链表：一般出现在跟merge相关的题目上。

* Add Binary
* Add Two Numbers
* Merge Sorted Array
* Merge Two Sorted Lists
* Multiply Strings
* Partition List

基本题，但是非常重要。面试中碰到任何一题一点也不奇怪。PIE, CC150和Leetcode都不约而同地包含了这类题。把这些题目做熟是必须的。基本上来说这类题的解法都是DFS，程序的大体框架非常类似，只是根据题目的要求代码稍作修改。当然每道题也有不同的解法，但是你应该根据自己的喜好把这类题目的解决方案统一化。熟悉了这类题目以后对于DFS(will be discussed in a separate section) 的理解会非常深刻。基本上一般的DFS的题目应该没什么问题了。  
无论是排列还是组合，这类题都有一个变形，就是要求不能有重复的输出。PIE和CC150都没有提到相应的解法，大家应该很好的体会一下。如果没有相应的准备，属于面试的时候比较容易跪的题目。  
  
Permutation  
输入没有重复：Permutations, CC150 9.5, PIE Chapter7 Permutations of a String输入有重复，输出不能有重复：Permutations II  
Next Permutation: 经典算法，背吧  
Permutation Sequence: 非常有意思的题目  
  
Combination  
纯粹的subset  
输入没有重复：Subsets, CC150 9.4, PIE Chapter7 Combinations of a String输入有重复，输出不能有重复：Subsets II  
需要满足一定要求的组合  
一个元素只能取一次(输入没有重复): Combinations  
一个元素可以取多次(输入没有重复): Combination Sum, CC150 9.8一个元素只能取一次(输入有重复，输出不能有重复）: Combination Sum II  
  
Gray Code: 具有subset的序列特点 （考虑CC150 9.4 Solution#2: Combinatorics)

<http://jane4532.blogspot.com/2013/06/zz-from-peking2.html>

数据结构  
Array, ArrayList  
String, StringBuffer  
LinkedList  
HashMap, HashSet  
Stack and Queue  
Tree:

* BT: binary tree
* BST: binary search tree,
* Balanced BST (red-black tree): TreeMap, TreeSet
* Trie: prefix tree
* Heap: PriorityQueue

Grpah  
  
注意：

1. Array和Linkedlist是最最基本的数据结构，因为他们可以构造很多其他的数据结构，比如String (C语言里String就是字符数组），HashMap, Stack和Queue (Java里Queue就是LinkedList实现了Queue的interface; Ruby里stack和queue都是array）, 以及Heap。
2. Heap is a tree logically, but array physically.
3. HashMap, Stack and Queue通常不会出现在问题里的数据结构中，因此一般作为solution的数据结构。但是面试中也常会让你实现这三种数据结构，另外就是CC150的3.2和3.5都是典型的Stack和Queue的题。Leetcode中并不涵盖这些内容，这几种数据结构在Leetcode中都是作为solution数据结构出现的 (没有的原因是这些题目不太适合OJ，因此需要单独练习）。
4. 目前Leetcode还不包含graph的题型

算法  
Sort: quick sort, merge sort, count sort, heap sort, bucket sort, radix sort, external sort, K selection etc.  
Merge: 2 array/list merge, k-way merge, interval merge etc.  
Binary search:  
Stack:  
Recursion and Iteration:  
DFS：pre-order, in-order, post-order for trees  
BFS: 需要用Queue  
DP: Top down and bottom up  
Greedy:  
Toposort: 需要用Queue  
  
注意：

1. Leetcode并没有包含各种sort算法，而通常面试很少让你直接去实现sort算法，但是大部分的相关编程技巧是包含在很多题目当中的, 比如quick sort的two pointers。
2. Merge跟sort是紧密相关的，单独拿出来是因为有很多这个类型的题目，需要一起总结。Merge操作的对象基本都是已经排好序的。
3. Stack虽说是数据结构，但是一般出现在solution里，因此代表了一类算法
4. Toposort面试作为难题也很有可能遇到，目前Leetcode还没有包括进去

玩竞赛对面试不利的一个地方就是面试经常遇到的数据结构比如LinkedList, Tree, 和算法Binary search，竞赛很少涉及到，因此一直心里都感觉到有些不安。  
Binary search非常tricky，虽说道理简单，但是面试的时候却很容易出bug，因此总结一下是必须的。假设i=0, j=A.length-1, 我做了一下LeetCode上的所有binary search的题目，发现了以下几点值得注意。

1. 终止条件不同 i<=j, i<j
2. mid的上下取向不同 i+(j-i)/2, j-(j-i)/2
3. 如何合理分半
4. 分半的时候取=mid, mid-1, or mid+1

Search a 2D Matrix： 这是一道普通的binary search。终止条件i<=j, mid取向i+(j-i)/2, 分半的时候=mid-1 or mid+1。  
Search for a Range：这道题需要终止条件i<j, mid取向两种都需要用到，分半的时候需要用到=mid。我发现一般＝mid的时候，终止条件往往是i<j, 不然会有死循环。  
  
如何合理分半：下边这几道题都很tricky，分半的时候都有各自的特点，很不容易一次写对。需要多多练习和体会。  
Search in Rotated Sorted Array  
Search in Rotated Sorted Array II  
Median of Two Sorted Arrays  
  
还有一个有趣的现象就是很多数学相关的题目也是通过binary search来解决的：  
Divide Two Integers：这题没做过面试也容易跪  
Pow(x, n)  
Sqrt(x)：其实算是一道典型的binary search题目，不过里边包括了几个tricky的地方，很难一次写对  
  
总的来说，LeetCode上边的这些binary search的题目cover的还比较全面，而且题目全部是面试高频题，需要练习一次写对

首先LeetCode上几乎所有的Linked list的题目都可以用two pointers来解决，或者会用到two pointers这个基本编程技巧。因此two pointers跟linked list是紧密相关的。因为two pointers以前已经总结过了，就不多讲了。  
  
其次，因为LinkedList和Array/ArrayList一样都具备有List的特性，因此很多题目都出现在了两种数据结构上，或者说很多题目都是可以把这两种数据结构互换的。比如：

* Add Two Numbers
* Convert Sorted List to Binary Search Tree
* Insert Interval
* Merge Intervals
* Merge k Sorted Lists
* Merge Two Sorted Lists
* Remove Duplicates from Sorted List
* Remove Duplicates from Sorted List  II

第三，LinkedList的题目大多自然而然使用iteration来解决的，但是我发现有些时候iteration比较容易出bug，换成recursion实现更容易。面试的时候万一iteration卡住可以换换recursion的思路。  
  
第四，dummy head非常有用，可以使代码简洁很多，并且容易写bug free的code。这个技巧可以大量使用。  
  
第五，今天做了一遍LinkedList的题目，发现两个地方容易出bug。一是two pointers loop完之后常常会有一个收尾的工作，比如Add Two Numbers需要处理carrier>0的情况。二是在swap了nodes之后，新的tail需要把next置空，不然就出现死循环了。

一直没有总结Tree，这次想总结一下结果却发现没有什么太多可以总结的。Leetcode上tree的题目还是比较全面的。我做了一遍发现基本上跑不出三个套路：  
1. Recursive DFS  
2. Iterative DFS  
3. BFS  
  
有些tree的题目比较tricky一些，但是最终解法还是逃不出这三个套路，所以我觉得面试的时候代码的质量就变得更加的重要了。因为没有什么太多总结的，下边就随便聊一下了。  
Leetcode上graph的题目涉及的很少，不过从算法和coding来说DFS，BFS完全适用于tree和graph。因此，把tree的题目练好了，graph的多数题目应该也不会有什么问题才对。当然graph涉及的算法比tree还是要多的，比如shortest path, toposort等等，但是DFS,BFS还是基本中的基本。因此做Leetcode上的tree的题目也相当于练习了graph的题目了。  
  
由于Tree的题目比较多，我感觉一些可以skip掉，如果时间不充裕的话。或者做一遍即可，不需要反复练习。这些题目或者太简单，或者面试不太可能碰到。

* Balanced Binary Tree
* Binary Tree Level Order Traversal II
* Maximum Depth of Binary Tree
* Minimum Depth of Binary Tree
* Same Tree
* Symmetric Tree
* Unique Binary Search Trees
* Unique Binary Search Trees II

Pre-order, In-order, Post-order traversal 需要会recursive和iterative的两种实现方式。可惜Leetcode上只包含了In-order，有些遗憾。  
  
Tree的serialization/deserialization也是常常被考到的题目，这个Leetcode目前还没有包含，当然套路还是DFS/BFS。  
  
LinkedList和Binary Tree相互转换的题目。

* Convert Sorted List to Binary Search Tree
* Flatten Binary Tree to Linked List (这题原题在CC150是一道双向链表题，不知道Leetcode上怎么改单向了。双向链表应该更复杂一些，大家要注意一下）