1. **两数之和**：unordered\_map遍历[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/two-sum/" \t "_blank)
2. **两数相加**：注意最后的进位不为0，[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/add-two-numbers/" \t "_blank)
3. **无重复字符最长子串**：滑动窗口+unordered\_map记录，[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/longest-substring-without-repeating-characters/" \t "_blank)
4. **寻找两个正序数组的中位数**：相当于找两个有序数组的第k大的数，二分，先判断k/2和小数组长度大小。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/median-of-two-sorted-arrays/" \t "_blank)
5. **最长回文子串**: 二维动态规划  
   dp[i] [j] = dp[i + 1] [j - 1]. if s[i + 1] [j - 1] 注意这个循环的遍历，外层是len from 0 to n，内层是i from 0 to n，j = i + len
6. **整数反转**：注意是否越界，可以用INT\_MAX / 10或者INT\_MIN / 10判断一下. [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/reverse-integer/submissions/)
7. **字符串转整数atoi**：主要就是越界的处理，和6一样，注意一点的就是-12 % 10 = -2，[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/string-to-integer-atoi/" \t "_blank)
8. **正则表达式匹配**：二维动态规划或者递归求解。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/string-to-integer-atoi/) dp[i] [j]：s的前i个和p的前j个是否匹配，dp[i] [0] = false, dp[0] [j] = dp[0] [j - 2] if p[j - 1] == '\*' dp[0] [0] = true
9. **盛最多水的容器**：双指针，每次选择小的方向往前/往后移动 [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/container-with-most-water/)
10. **罗马数字转整数**： 一次遍历，遍历到第i时，看i + 1的数字，来判断+还是-。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/roman-to-integer/" \t "_blank)
11. **数字转罗马数字**：这个就把所有的罗马数字对应的数字列举出来，然后循环相减。 [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/integer-to-roman/)
12. **最长公共前缀**：横向比较，每次取两个算出最长公共前缀，得到的结果和后面一个继续算。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/longest-common-prefix/)
13. **三数之和**：排序+双指针 [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/3sum/)
14. **电话号码的字母组合**：当循环数不定时，就去递归/回溯吧。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/letter-combinations-of-a-phone-number/" \t "_blank)
15. **删除链表倒数第N个节点**：双指针，先走N个节点，再一起走。找到删除。 注意用一个dummy节点，放到头部。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/remove-nth-node-from-end-of-list/" \t "_blank)
16. **有效的括号**：栈解决，遇到左括号，就进栈，遇到右括号，就出栈，出的时候判断对错。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/valid-parentheses/)
17. **合并两个有序链表**：头部的判断，为了不用dummy，先判断两个链表是否都为空。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/merge-two-sorted-lists/" \t "_blank)
18. **括号生成**：组合问题，就上回溯，回溯就是不断加左括号，回溯，弹出左括号，左括号数量大于右括号时，加右括号，回溯，弹出右括号。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/generate-parentheses/)
19. **合并K个有序链表**：归并排序的典型应用。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/merge-k-sorted-lists/)
20. **删除有序数组中的重复项**：用一个临时变量一致保存前面的不一样的元素。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/remove-duplicates-from-sorted-array/)
21. **实现strStr()**：KMP算法，没法，只能硬背，不要尝试理解了，太费时间。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/implement-strstr/" \t "_blank)
22. **两数相除**：倍增方法，注意把数都处理成负的，防止越界。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/divide-two-integers/)
23. **旋转图像（90度**）：两种方法：先主对角交换，再左右交换；或者 找到旋转后的递推关系：dp[i] [j]---->dp[j] [n - i - 1] [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/rotate-image/)
24. **搜索旋转排序数组**：二分，注意先用nums[mid]和nums[left]相比较，确定[left, mid]和[mid+ 1, right]那个有序，再比较target和nums[left]/nums[right]进行比较 [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/search-in-rotated-sorted-array/)
25. **在排序数组中查找元素的第一个和最后一个位置**: 二分，查找左边界，右边界。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/find-first-and-last-position-of-element-in-sorted-array/" \t "_blank)
26. **有效的数独**： 注意小数独的遍历[k / 3 \* 3, k / 3 \* 3 + 2]，[k % 3 \* 3, k % 3 \* 3 + 2] [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/valid-sudoku/)
27. **外观数列**：直接写。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/count-and-say/)
28. **缺失的第一个正数**：[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/first-missing-positive/submissions/) 原地置换，把[1, n]范围内的数对应的下标都取负。（这步之前先把<=0的变成n+1)
29. **接雨水**：考虑每个位置接多少雨水，也就是求左右两边在该位置最高高度的较小那个减去该位置高度。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/trapping-rain-water/)
30. **通配符匹配**：二维动态规划
31. **全排列**：回溯法[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/permutations/)
32. **字母异位词分组**：使用hash，key是排序后的字符串，value是vector，存的是原始的key。
33. **Pow(x, n)**：主要考虑越界
34. **最大子序和**：一维动态规划，dp[i] = dp[i - 1] + nums[i - 1] if dp[i - 1] > 0 [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/maximum-subarray/)
35. **螺旋矩阵**：确定好上下左右四个边界，遍历时一直更新，注意跳出条件[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/spiral-matrix/)
36. **跳跃游戏**：解法1: dp[i] = dp[i - k] && nume[i - k] >= k. 解法2: 贪心 记录当每一步能到的最大值.该值要大于等于i。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/jump-game/" \t "_blank)
37. **合并区间**：先排序，然后插入。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/merge-intervals/submissions/)
38. **不同路径**：解法1: 动态规划。 解法2: 数学方法，组合问题。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/unique-paths/" \t "_blank)
39. **加1**: 注意对9的处理。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/plus-one/submissions/" \t "_blank)
40. **x的平方根**：典型的二分求左边界。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/sqrtx/)
41. **爬楼梯**：简单的dp。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/climbing-stairs/" \t "_blank)
42. **矩阵置零**：用第一行第一列记录该行该列是否有0，然后再单独处理首列首行。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/set-matrix-zeroes/)
43. **颜色分类**：双指针。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/sort-colors/)
44. **最小覆盖子串**：滑动窗口。比较难，用两个map存储，一个用于存储信息，一个用于记录窗口里的信息。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/minimum-window-substring/" \t "_blank)
45. **子集**：回溯法，选择本次元素，回溯，不选本次元素，回溯 [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/subsets/)
46. **单词搜索**：就是从每一个坐标，递归搜索. 回溯法。visted[i] [j] = true 回溯 false
47. **柱状图中最大矩形**：对于每个点，需要求出左边的第一个小于该点高度的坐标，右边第一个小于该点长度的坐标，这就需要递增栈，一个是i = 0...n-1，一个是i = n - 1...0。递增栈模板：[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/largest-rectangle-in-histogram/" \t "_blank)
48. **合并两个有序数组**：从后往前遍历 [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/merge-sorted-array/)
49. 解码方法：一维动态规划，dp[i]：表示s的前i个字符编码数。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/decode-ways/" \t "_blank)  
    c++ dp[i] = dp[i - 1] if s[i - 1] // 是合法字符 dp[i] = dp[i] + dp[i - 2] if s[i - 2, i - 1] // 是合法字符
50. **二叉树的中序遍历**：递归和非递归两种写法。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/binary-tree-inorder-traversal/)
51. **编辑距离**：二维dp. [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/edit-distance/)  
    c++ // dp[i][j]: 表示word1的前i字母，word2的前j个字母，编辑距离 if (word1[i - 1] == word2[j - 1]) dp[i][j] = dp[i - 1][j - 1] else dp[i][j] = min(dp[i][j - 1], dp[i - 1][j], dp[i - 1][j - 1]) + 1
52. **验证二叉搜索树**：递归，注意需要传递辅助信息：最小节点，最大节点。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/validate-binary-search-tree/submissions/)  
    c++ isValidBSTCore(TreeNode\* root, TreeNode\* min\_tree, TreeNode\* max\_tree)
53. **对称二叉树**：递归，需要辅助函数，输入是两个节点，判断这两个节点是否是对称的。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/symmetric-tree/)  
    c++ bool isSymmetricCore(TreeNode\* left, TreeNode\* right);
54. **二叉树的层序遍历**：easy题，借助队列[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/binary-tree-level-order-traversal/" \t "_blank)
55. **二叉树的锯齿形层序遍历**：层序+双栈结合，再使用一个标识位来标识每次是左右孩子哪个先入栈。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/binary-tree-zigzag-level-order-traversal/submissions/" \t "_blank)
56. **二叉树的最大深度**：max(l, r) + 1，递归。 [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/maximum-depth-of-binary-tree/)
57. **从前序与中序遍历序列构造二叉树**：找到根，递归构造左右子树。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal/)
58. **将有序数组转换为二叉搜索树**: 每次取中间节点，进行根节点的构建。 [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/convert-sorted-array-to-binary-search-tree/)
59. **填充每个节点的下一个右侧节点指针**：构建辅助函数，传入左右孩子节点。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/populating-next-right-pointers-in-each-node/)
60. **杨辉三角**：easy，找到每一行元素与上一行元素的关系。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/pascals-triangle/" \t "_blank)
61. **买卖股票的最佳时机**：easy，记录每个元素之前的最小元素值。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/best-time-to-buy-and-sell-stock/" \t "_blank)
62. **买卖股票的最佳时机II，不限次数**：二维dp，其中第一维是第i天，第二维是持有/不持有。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/best-time-to-buy-and-sell-stock-ii/" \t "_blank)
63. **二叉树的最大路径和**：像这种不直接的题，肯定需要辅助函数，辅助函数记录每个节点加上其一个左右孩子节点中的一个可能构成的最大的边。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/binary-tree-maximum-path-sum/submissions/)
64. **验证回文串**：左右指针遍历即可。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/valid-palindrome/)
65. **单词接龙**：解法1: BFS求解，利用队列存每层结果，利用hashset存字典，再用一个hashset存已经访问过的节点（必须有，否则必出现死循环）。解法2: 双向BFS，一个hashset存字典，一个hashset存正向遍历，一个hashset存反向遍历，一个hashset存已经访问过的，每层遍历，交换前两个hashset。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/word-ladder/" \t "_blank)
66. **最长连续序列**：一个hashset存储所有的，然后从begin开始，遍历begin，以及其所有上和下连续节点，遍历时不断删除，并同时更新最大连续值。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/longest-consecutive-sequence/)
67. **被围绕的区域**：先对边界的O进行DFS寻找其所有的O，都修改成#，然后把矩阵都变成X，再把所有的#都修改成O。核心是DFS那块，四个方向。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/surrounded-regions/" \t "_blank)
68. **分割回文串**：这种分割问题，求组合的，等等，都是回溯法，这一题需要先用二维数组dp[i] [j]存储字符串中i到j是否是回文串。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/palindrome-partitioning/" \t "_blank)  
    c++ // 看下回溯的模板 void dfs(vector<vector<string>> & res, vector<string> &temp, string s, int index) { if (index == s.length()) { res.push\_back(temp); return; } for (int i = index; i < s.length(); i++) { if (dp[index][i]) { temp.push\_back(s.substr(index, i - index + 1)); dfs(res, temp, s, i + 1); temp.pop\_back(); } } }
69. **加油站**：贪心，用两个变量分别记录gas-cost的累积和total和cur，如果cur小于0，则重新置cur=0，更新res为下一个i，继续记录。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/gas-station/submissions/" \t "_blank)  
    c++ for (int i = 0; i < gas.size(); i++) { total += gas[i] - cost[i]; cur += gas[i] - cost[i]; if (cur < 0) { res = (i + 1) % gas.size(); cur = 0; } }
70. **只出现一次的数字**：使用异或操作。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/single-number/)
71. **复制带随机指针的链表**：在每个节点后面多加一个节点，都加完以后再单独处理每个节点的random节点，然后再拆分。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/copy-list-with-random-pointer/" \t "_blank)
72. **单词拆分**：一维动态规划。 拿到题目如果没有思路，就想想能不能用动态规划来解决，不能再考虑回溯，BFS等方法。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/word-break/" \t "_blank)  
    c++ // d[i] : 表示前i个字符是否可以被拆分都在单词表里 dp[i] = dp[i - k] && s[i-k+1...i] in wordDict
73. **单词拆分II**：就是求所有的可能，这种就是回溯法（DFS），确定回溯的输入参数，以及退出条件，回溯的可能步骤。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/word-break-ii/submissions/" \t "_blank)
74. **LRU缓存机制**：使用双向链表存储key，value。使用unordered\_map存储key和节点，方便寻找。注意get操作需要把访问的节点移动到链表头，put操作需要把访问的节点移动到头，超出capacity 的，要删除尾部的，所有过程都要更新map。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/lru-cache/" \t "_blank)
75. **排序链表**：归并排序，每次找到链表的中点（注意奇数节点个数/偶数节点个数有点区别），把中点->next修改成nullptr。然后对这两段分别调用排序，排好序的，子集再调用合并两个链表的操作。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/sort-list/" \t "_blank)
76. **直线上最多的点数**：对于每一个点，统计该点与其他点的斜率，用unordered\_map的key存储斜率，用value存储个数。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/max-points-on-a-line/" \t "_blank)
77. **基本计算器II**：[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/basic-calculator-ii/" \t "_blank)
78. **寻找峰值**：这个二分，有点意思。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/find-peak-element/)
79. **分数到小数**：长除法，先判断分子分母是否为0，再判断正负，再判断是否能够整除，然后循环存储，并记录余数。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/fraction-to-recurring-decimal/submissions/" \t "_blank)
80. **多数元素**：[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/majority-element/)
81. **逆波兰表达式求值**：利用栈。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/evaluate-reverse-polish-notation/)
82. **乘积最大子数组**：使用两个一维dp，注意压缩成临时状态时，变量不要叠加，要换成其他变量代替。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/maximum-product-subarray/submissions/" \t "_blank)  
    c++ max\_dp[i] = max(min\_dp[i - 1] \* nums[i - 1], max\_dp[i - 1] \* nums[i- 1], nums[i - 1]); min\_dp[i] = min(min\_dp[i - 1] \* nums[i - 1], max\_dp[i - 1] \* nums[i- 1], nums[i - 1]);
83. **最小栈**：使用两个栈。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/min-stack/)
84. **相交链表**：先计算出两个链表长度差，然后让长的链表先走这个差，再一起走。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/intersection-of-two-linked-lists/)
85. **N皇后**：经典回溯。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/n-queens/)
86. **阶乘后的零**：[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/factorial-trailing-zeroes/)  
    c++ return n == 0 ? 0 : n / 5 + trailingZeroes(n / 5);
87. **最大数**：排序+重构比较函数。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/largest-number/" \t "_blank)
88. **旋转数组**：三次翻转。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/rotate-array/)
89. **颠倒二进制位**：利用左移右移，n & 1取最后一位。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/reverse-bits/)
90. **位1的个数**：每次利用n & (n - 1)去掉右边的1 [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/number-of-1-bits/)
91. **打家劫舍**：一维dp，easy。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/house-robber/" \t "_blank)  
    dp[i]表示抢劫到第i家，累计最大收益 dp[i] = max(dp[i - 1], dp[i - 2] + nums[i]); 这种都可以进行状态压缩
92. **打家劫舍III**：后续遍历。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/house-robber-iii/" \t "_blank)  
    c++ 正确的理解: 自底向上的 rob(root)就是以root为根，所能偷到的最高金额，这里涉及到可能偷root，也可能不偷root的 由于不能偷相邻的，如果偷root，则不能偷其左右孩子，但是可以偷其左右孩子的孩子 max(root->val + rob(root->left->left) + rob(root->left->right) + rob(root->right->left) + rob(root->right->right), rob(root->left) + rob(root->right)) 使用map存储，减少遍历
93. **岛屿数量**：[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/number-of-islands/)
94. **快乐数**：用一个set保存每次变换结果，如果某次结果在set里，说明重复出现过。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/happy-number/)
95. **计数质数**：筛选法。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/count-primes/)
96. **反转链表**：[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/reverse-linked-list/)
97. **课程表**：要做某事，必须先做其他事，问是否有可能，说白了就是判断图中是否有环，拓扑排序。两种做法：一种是DFS：  
      
    一种是BFS：先构建邻接矩阵，并统计每个课程（节点）的入度。把入度为0的都加到队列中，然后遍历队列，弹出元素（入度为0的节点），以从邻接矩阵中找到以该节点为入度的其他节点，并分别将它们的入度都减去一。最后判断所有节点的入度是否为0。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/course-schedule/" \t "_blank)，课程表II：[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/course-schedule-ii/" \t "_blank)
98. **实现 Trie (前缀树)**：创建时，先创建26个孩子节点，并都置为nullptr。insert时，对于某个字母，先判断以其为索引的孩子节点是否存在，不存在需要创建，遍历到最后，需要置结束标识位。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/implement-trie-prefix-tree/" \t "_blank)
99. **单词搜索II**：[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/word-search-ii/" \t "_blank)
100. **存在重复元素**：排序/哈希表[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/contains-duplicate/" \t "_blank)
101. **天际线**：使用一个vector排序，遍历，放到multiset里（右端点放进去，左端点出来），获取最大高度，更新结果。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/the-skyline-problem/" \t "_blank)
102. **二叉树中所有距离为K的结点**：[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/all-nodes-distance-k-in-binary-tree/)
103. **二叉搜索树中第K小的元素**：[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/kth-smallest-element-in-a-bst/)
104. **回文链表**：找到链表中点进行反转后半部分，再比较。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/palindrome-linked-list/)
105. **二叉树的最近公共祖先**：[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/lowest-common-ancestor-of-a-binary-tree/)
106. **删除链表中的节点**：赋值为链表下一个节点的值。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/delete-node-in-a-linked-list/)
107. **除自身以外数组的乘积**：左右乘积列表。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/product-of-array-except-self/)
108. **滑动窗口最大值**：维持一个非严格单调递减队列。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/sliding-window-maximum/)
109. **搜索二维矩阵**：右上->左下搜索，可以用二分加快搜索[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/search-a-2d-matrix-ii/" \t "_blank)
110. **有效的字母异位词**：[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/valid-anagram/)
111. **缺失的数字**：求和，再减。[LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/missing-number/)
112. **完全平方数**：返回和为n的完全平方数的最少数量，求最少这种问题，首先想到dp。  
     dp[n] = min(dp[n - i \* i]). for i = 0...sqrt(n) init dp[n] = i. [LC](https://link.zhihu.com/?target=https%3A//leetcode-cn.com/problems/perfect-squares/)
113. **移动零**：把0移动数组的最左边/最右边，双指针，一个遍历，一个记录0的位置。[LC](https://zhuanlan.zhihu.com/p/386929820/edit)