DSA Questions(BackTracking)

**Very Easy:**

1. **Find All Paths in a Maze**

You are given a maze represented as a 2D grid of integers. The maze contains cells that are either open (1) or blocked (0). Your task is to find all possible paths from the start at the top-left corner (0,0)(0, 0)(0,0) to the end at the bottom-right corner (n−1,m−1)(n-1, m-1)(n−1,m−1), moving only through open cells.

A valid path is one where:

You start at the top-left corner and end at the bottom-right corner.

You move only through cells containing 1.

You move only up, down, left, or right (no diagonal moves).

You do not visit any cell more than once in a single path.

If there is no valid path from start to end, return a message indicating that no path exists.  
  
**Example 1:  
Input:**

**Maze:**

**1 0 1**

**1 1 1**

**0 1 1**

**Output:**

**All Paths:**

**(0,0) (1,0) (1,1) (1,2) (2,2)**

**(0,0) (1,0) (1,1) (2,1) (2,2)**

**Example 2:  
Input:**

**Maze:**

**1 0 0**

**0 0 1**

**1 1 0**

**Output:**

**No path exists!**

**Example 3:  
Input:**

**Maze:**

**1 1 1**

**0 0 1**

**1 1 1**

**Output:**

**All Paths:**

**(0,0) (0,1) (0,2) (1,2) (2,2)**

**Constraints**

**Grid Size: 1≤n,m≤121 \leq n, m \leq 121≤n,m≤12 (e.g., 12×1212 \times 1212×12 is the largest allowed maze).**

**Start and End Validity:**

**Both (0,0)(0,0)(0,0) and (n−1,m−1)(n-1,m-1)(n−1,m−1) must be 1 for any path to exist.**

**Cell Visits: Each cell can be visited only once in a single path.**

**Output Paths:**

**The order of paths does not matter.**

1. **Combination of Dice Rolls**

Given n dice and k faces, generate all possible ways to roll the dice such that the total sum equals a target value.

**Example 1:  
Input: n = 2, target = 5 Output: [[1, 4], [2, 3], [3, 2], [4, 1]]**

**Example 2:  
Input: n = 2, target = 0 Output: []  
  
Constraints:**

**n=1: Single die; combinations are all possible face values less than or equal to target.**

**target<ntarget < ntarget<n: No combinations possible because the minimum achievable sum is nnn.**

**target>n×ktarget > n \times ktarget>n×k: No combinations possible because the maximum achievable sum is n×kn \times kn×k.**

**Large n and small k: Many combinations to explore, require efficient pruning.**

1. **Paths in a Grid**

Given a grid of size n×nn \times nn×n, find all possible paths from the top-left corner to the bottom-right corner, where at each step, you can only move right or down. You need to return all possible paths that you can take

**Example 1:**

**Input:  
For a 2×2 grid:**

**n = 2  
Output:["RRDD", "RDRD", "RDDR", "DRRD", "DRDR", "DDRR"]**

**Explanation:  
Each path consists of exactly nnn moves down and nnn moves right. The number of paths for an n×nn \times nn×n grid is the number of ways you can arrange these moves, which is (2nn)\binom{2n}{n}(n2n​).**

**Constraints:**

**Grid size (n):  
1≤n≤101 \leq n \leq 101≤n≤10**

**Output size:  
The total number of paths grows exponentially with nnn. For n=10n = 10n=10, the number of paths will be (2010)\binom{20}{10}(1020​), which is manageable for typical constraints in competitive programming.**

1. **Generate Numbers with a Given Sum**

Generate all numbers of length n whose digits sum up to a target value sum, The digits of the number will be between 0 and 9, and we will generate combinations of digits such that their sum equals the target.

**Example 1:**

**Input: n = 2 and sum = 5**

**Output: 14 23 32 41 50  
  
Example 2:  
Input: n = 3 and sum = 5**

**Output: 104 113 122 131 140 203 212 221 230 302 311 320 401 410 500**

**Constraints:**

**1 <= n <= 9: The number of digits must be between 1 and 9.**

**1 <= sum <= 100: The sum of the digits must be between 1 and 100.**

**The first digit cannot be zero if n > 1.**

1. **Generate Binary Strings**

A binary string consists of only 0s and 1s, and our goal is to generate all possible combinations of these digits for a string of length n.  
  
**Example 1:**

**Input: n = 2**

**Output: 00 01 10 11  
  
Example 2:  
Input: n = 3**

**Output: 000 001 010 011 100 101 110 111**

**Constraints:**

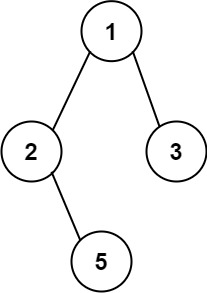
**1 <= n <= 10: The length of the binary string can range from 1 to 10.**

**The number of binary strings for length n is 2^n. For example, for n = 10, there are 1024 binary strings.**

**Easy:   
  
1. Binary Tree Paths**Given the root of a binary tree, return all root-to-leaf paths in any order.

A leaf is a node with no children.

**Example 1:**

****

**Input: root = [1,2,3,null,5]Output: ["1->2->5","1->3"]**

**Example 2:**

**Input: root = [1]Output: ["1"]**

**Constraints:**

**The number of nodes in the tree is in the range [1, 100].**

**-100 <= Node.val <= 100**

**Reference: <https://leetcode.com/problems/binary-tree-paths/description/?envType=problem-list-v2&envId=backtracking>**

1. **Sum of All Subset XOR Totals**

The XOR total of an array is defined as the bitwise XOR of all its elements, or 0 if the array is empty.

For example, the XOR total of the array [2,5,6] is 2 XOR 5 XOR 6 = 1.

Given an array nums, return the sum of all XOR totals for every subset of nums.

Note: Subsets with the same elements should be counted multiple times.

An array a is a subset of an array b if a can be obtained from b by deleting some (possibly zero) elements of b.

**Example 1:**

**Input: nums = [1,3]**

**Output: 6**

**Explanation: The 4 subsets of [1,3] are:**

**- The empty subset has an XOR total of 0.**

**- [1] has an XOR total of 1.**

**- [3] has an XOR total of 3.**

**- [1,3] has an XOR total of 1 XOR 3 = 2.**

**0 + 1 + 3 + 2 = 6**

**Example 2:**

**Input: nums = [5,1,6]**

**Output: 28**

**Explanation: The 8 subsets of [5,1,6] are:**

**- The empty subset has an XOR total of 0.**

**- [5] has an XOR total of 5.**

**- [1] has an XOR total of 1.**

**- [6] has an XOR total of 6.**

**- [5,1] has an XOR total of 5 XOR 1 = 4.**

**- [5,6] has an XOR total of 5 XOR 6 = 3.**

**- [1,6] has an XOR total of 1 XOR 6 = 7.**

**- [5,1,6] has an XOR total of 5 XOR 1 XOR 6 = 2.**

**0 + 5 + 1 + 6 + 4 + 3 + 7 + 2 = 28**

**Example 3:**

**Input: nums = [3,4,5,6,7,8]**

**Output: 480**

**Explanation: The sum of all XOR totals for every subset is 480.**

**Constraints:**

**1 <= nums.length <= 12**

**1 <= nums[i] <= 20  
  
Reference: <https://leetcode.com/problems/sum-of-all-subset-xor-totals/description/?envType=problem-list-v2&envId=backtracking>**

1. **Binary Watch**

A binary watch has 4 LEDs on the top to represent the hours (0-11), and 6 LEDs on the bottom to represent the minutes (0-59). Each LED represents a zero or one, with the least significant bit on the right.

For example, the below binary watch reads "4:51".



Given an integer turnedOn which represents the number of LEDs that are currently on (ignoring the PM), return all possible times the watch could represent. You may return the answer in any order.

The hour must not contain a leading zero.

For example, "01:00" is not valid. It should be "1:00".

The minute must consist of two digits and may contain a leading zero.

For example, "10:2" is not valid. It should be "10:02".

**Example 1:**

**Input: turnedOn = 1Output: ["0:01","0:02","0:04","0:08","0:16","0:32","1:00","2:00","4:00","8:00"]**

**Example 2:**

**Input: turnedOn = 9Output: []**

**Constraints:**

**0 <= turnedOn <= 10**

**Reference: <https://leetcode.com/problems/binary-watch/description/?envType=problem-list-v2&envId=backtracking>**

1. **Permutations**

Given an array nums of distinct integers, return all the possible

permutations

. You can return the answer in any order.

**Example 1:**

**Input: nums = [1,2,3]**

**Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]**

**Example 2:**

**Input: nums = [0,1]**

**Output: [[0,1],[1,0]]**

**Example 3:**

**Input: nums = [1]**

**Output: [[1]]**

**Constraints:**

**1 <= nums.length <= 6**

**-10 <= nums[i] <= 10**

**All the integers of nums are unique.**

**Reference: <https://leetcode.com/problems/permutations/description/?envType=study-plan-v2&envId=top-interview-150>**

1. **Subsets**

Given an integer array nums of unique elements, return all possible

subsets

(the power set).

The solution set must not contain duplicate subsets. Return the solution in any order.

**Example 1:**

**Input: nums = [1,2,3]**

**Output: [[],[1],[2],[1,2],[3],[1,3],[2,3],[1,2,3]]**

**Example 2:**

**Input: nums = [0]**

**Output: [[],[0]]**

**Constraints:**

**1 <= nums.length <= 10**

**-10 <= nums[i] <= 10**

**All the numbers of nums are unique.**

**Reference: <https://leetcode.com/problems/subsets/description/>**

**Medium:**

1. **Letter Combinations of a Phone Number**

Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.



**Example 1:**

**Input: digits = "23"**

**Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]**

**Example 2:**

**Input: digits = ""**

**Output: []**

**Example 3:**

**Input: digits = "2"**

**Output: ["a","b","c"]**

**Constraints:**

**0 <= digits.length <= 4**

**digits[i] is a digit in the range ['2', '9'].**

**Reference: <https://leetcode.com/problems/letter-combinations-of-a-phone-number/description/?envType=study-plan-v2&envId=top-interview-150>**

1. **Combinations**

Given two integers n and k, return all possible combinations of k numbers chosen from the range [1, n].

You may return the answer in any order.

**Example 1:**

**Input: n = 4, k = 2**

**Output: [[1,2],[1,3],[1,4],[2,3],[2,4],[3,4]]**

**Explanation: There are 4 choose 2 = 6 total combinations.**

**Note that combinations are unordered, i.e., [1,2] and [2,1] are considered to be the same combination.**

**Example 2:**

**Input: n = 1, k = 1**

**Output: [[1]]**

**Explanation: There is 1 choose 1 = 1 total combination.**

**Constraints:**

**1 <= n <= 20**

**1 <= k <= n**

**Reference: <https://leetcode.com/problems/combinations/description/?envType=study-plan-v2&envId=top-interview-150>**

1. **[Combination Sum](https://leetcode.com/problems/combination-sum/)**

Given an array of distinct integers candidates and a target integer target, return a list of all unique combinations of candidates where the chosen numbers sum to target. You may return the combinations in any order.

The same number may be chosen from candidates an unlimited number of times. Two combinations are unique if the

frequency

 of at least one of the chosen numbers is different.

The test cases are generated such that the number of unique combinations that sum up to target is less than 150 combinations for the given input.

**Example 1:**

**Input: candidates = [2,3,6,7], target = 7**

**Output: [[2,2,3],[7]]Explanation:**

**2 and 3 are candidates, and 2 + 2 + 3 = 7. Note that 2 can be used multiple times.**

**7 is a candidate, and 7 = 7.**

**These are the only two combinations.**

**Example 2:**

**Input: candidates = [2,3,5], target = 8**

**Output: [[2,2,2,2],[2,3,3],[3,5]]**

**Example 3:**

**Input: candidates = [2], target = 1**

**Output: []**

**Constraints:**

**1 <= candidates.length <= 30**

**2 <= candidates[i] <= 40**

**All elements of candidates are distinct.**

**1 <= target <= 40**

**Reference: <https://leetcode.com/problems/combination-sum/description/?envType=study-plan-v2&envId=top-interview-150>**

1. **Generate Parentheses**

Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses.

**Example 1:**

**Input: n = 3Output: ["((()))","(()())","(())()","()(())","()()()"]**

**Example 2:**

**Input: n = 1Output: ["()"]**

**Constraints:**

**1 <= n <= 8**

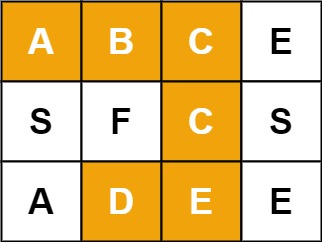
**Reference: <https://leetcode.com/problems/generate-parentheses/description/?envType=study-plan-v2&envId=top-interview-150>**

1. **Word Search**

Given an m x n grid of characters board and a string word, return true if word exists in the grid.

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

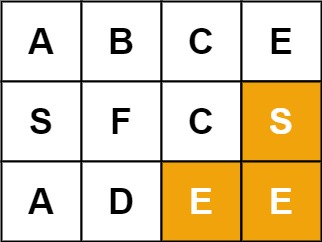
**Example 1:**

****

**Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCCED"**

**Output: true**

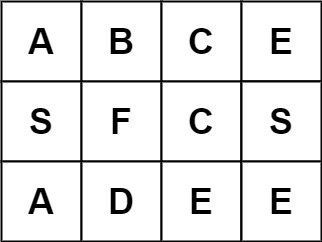
**Example 2:**

****

**Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "SEE"**

**Output: true**

**Example 3:**

****

**Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCB"**

**Output: false**

**Constraints:**

**m == board.length**

**n = board[i].length**

**1 <= m, n <= 6**

**1 <= word.length <= 15**

**board and word consists of only lowercase and uppercase English letters.**

**Reference: <https://leetcode.com/problems/word-search/description/?envType=study-plan-v2&envId=top-interview-150>**

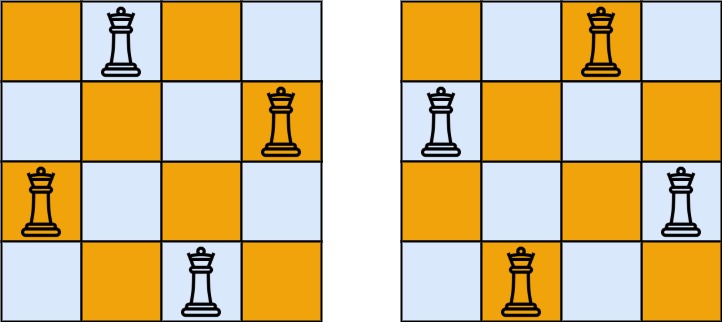
Hard:

1. **[N-Queens II](https://leetcode.com/problems/n-queens-ii/)**

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

Given an integer n, return the number of distinct solutions to the n-queens puzzle.

**Example 1:**

****

**Input: n = 4**

**Output: 2**

**Explanation: There are two distinct solutions to the 4-queens puzzle as shown.**

**Example 2:**

**Input: n = 1**

**Output: 1**

**Constraints:**

**1 <= n <= 9**

**Reference: <https://leetcode.com/problems/n-queens-ii/description/?envType=study-plan-v2&envId=top-interview-150>**

1. **[Combination Sum III](https://leetcode.com/problems/combination-sum-iii/)**

Find all valid combinations of k numbers that sum up to n such that the following conditions are true:

Only numbers 1 through 9 are used.

Each number is used at most once.

Return a list of all possible valid combinations. The list must not contain the same combination twice, and the combinations may be returned in any order.

**Example 1:**

**Input: k = 3, n = 7**

**Output: [[1,2,4]]**

**Explanation:**

**1 + 2 + 4 = 7**

**There are no other valid combinations.**

**Example 2:**

**Input: k = 3, n = 9**

**Output: [[1,2,6],[1,3,5],[2,3,4]]**

**Explanation:**

**1 + 2 + 6 = 9**

**1 + 3 + 5 = 9**

**2 + 3 + 4 = 9**

**There are no other valid combinations.**

**Example 3:**

**Input: k = 4, n = 1**

**Output: []**

**Explanation: There are no valid combinations.**

**Using 4 different numbers in the range [1,9], the smallest sum we can get is 1+2+3+4 = 10 and since 10 > 1, there are no valid combination.**

**Constraints:**

**2 <= k <= 9**

**1 <= n <= 60**

**Reference:<https://leetcode.com/problems/combination-sum-iii/description/?envType=study-plan-v2&envId=leetcode-75>**

1. **[Gray Code](https://leetcode.com/problems/gray-code/)**

An n-bit gray code sequence is a sequence of 2n integers where:

Every integer is in the inclusive range [0, 2n - 1],

The first integer is 0,

An integer appears no more than once in the sequence,

The binary representation of every pair of adjacent integers differs by exactly one bit, and

The binary representation of the first and last integers differs by exactly one bit.

Given an integer n, return any valid n-bit gray code sequence.

**Example 1:**

**Input: n = 2**

**Output: [0,1,3,2]**

**Explanation:**

**The binary representation of [0,1,3,2] is [00,01,11,10].**

**- 00 and 01 differ by one bit**

**- 01 and 11 differ by one bit**

**- 11 and 10 differ by one bit**

**- 10 and 00 differ by one bit**

**[0,2,3,1] is also a valid gray code sequence, whose binary representation is [00,10,11,01].**

**- 00 and 10 differ by one bit**

**- 10 and 11 differ by one bit**

**- 11 and 01 differ by one bit**

**- 01 and 00 differ by one bit**

**Example 2:**

**Input: n = 1**

**Output: [0,1]**

**Constraints:**

**1 <= n <= 16**

**Reference: <https://leetcode.com/problems/gray-code/description/?envType=problem-list-v2&envId=backtracking>**

1. **[Restore IP Addresses](https://leetcode.com/problems/restore-ip-addresses/)**

A valid IP address consists of exactly four integers separated by single dots. Each integer is between 0 and 255 (inclusive) and cannot have leading zeros.

For example, "0.1.2.201" and "192.168.1.1" are valid IP addresses, but "0.011.255.245", "192.168.1.312" and "192.168@1.1" are invalid IP addresses.

Given a string s containing only digits, return all possible valid IP addresses that can be formed by inserting dots into s. You are not allowed to reorder or remove any digits in s. You may return the valid IP addresses in any order.

**Example 1:**

**Input: s = "25525511135"**

**Output: ["255.255.11.135","255.255.111.35"]**

**Example 2:**

**Input: s = "0000"**

**Output: ["0.0.0.0"]**

**Example 3:**

**Input: s = "101023"**

**Output: ["1.0.10.23","1.0.102.3","10.1.0.23","10.10.2.3","101.0.2.3"]**

**Constraints:**

**1 <= s.length <= 20**

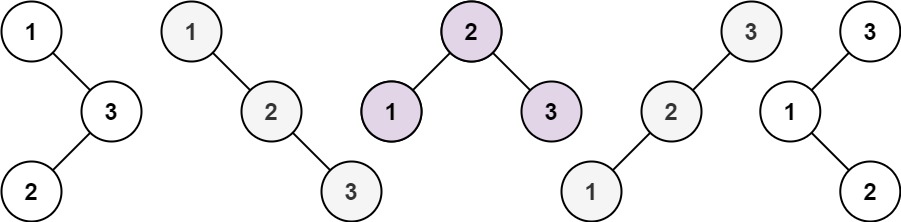
**s consists of digits only.**

**Reference: <https://leetcode.com/problems/restore-ip-addresses/description/?envType=problem-list-v2&envId=backtracking>**

1. **[Unique Binary Search Trees II](https://leetcode.com/problems/unique-binary-search-trees-ii/)**

Given an integer n, return all the structurally unique BST's (binary search trees), which has exactly n nodes of unique values from 1 to n. Return the answer in any order.

**Example 1:**

****

**Input: n = 3**

**Output: [[1,null,2,null,3],[1,null,3,2],[2,1,3],[3,1,null,null,2],[3,2,null,1]]**

**Example 2:**

**Input: n = 1**

**Output: [[1]]**

**Constraints:**

**1 <= n <= 8**

**Reference: <https://leetcode.com/problems/unique-binary-search-trees-ii/description/?envType=problem-list-v2&envId=backtracking>**

**Very Hard:**

1. **[Sudoku Solver](https://leetcode.com/problems/sudoku-solver/)**

Write a program to solve a Sudoku puzzle by filling the empty cells.

A sudoku solution must satisfy all of the following rules:

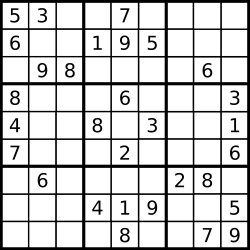
Each of the digits 1-9 must occur exactly once in each row.

Each of the digits 1-9 must occur exactly once in each column.

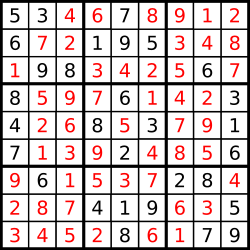
Each of the digits 1-9 must occur exactly once in each of the 9 3x3 sub-boxes of the grid.

The '.' character indicates empty cells.

**Example 1:**

****

**Input: board = [["5","3",".",".","7",".",".",".","."],["6",".",".","1","9","5",".",".","."],[".","9","8",".",".",".",".","6","."],["8",".",".",".","6",".",".",".","3"],["4",".",".","8",".","3",".",".","1"],["7",".",".",".","2",".",".",".","6"],[".","6",".",".",".",".","2","8","."],[".",".",".","4","1","9",".",".","5"],[".",".",".",".","8",".",".","7","9"]]Output: [["5","3","4","6","7","8","9","1","2"],["6","7","2","1","9","5","3","4","8"],["1","9","8","3","4","2","5","6","7"],["8","5","9","7","6","1","4","2","3"],["4","2","6","8","5","3","7","9","1"],["7","1","3","9","2","4","8","5","6"],["9","6","1","5","3","7","2","8","4"],["2","8","7","4","1","9","6","3","5"],["3","4","5","2","8","6","1","7","9"]]Explanation: The input board is shown above and the only valid solution is shown below:**

****

**Constraints:**

**board.length == 9**

**board[i].length == 9**

**board[i][j] is a digit or '.'.**

**It is guaranteed that the input board has only one solution.**

**Reference: <https://leetcode.com/problems/sudoku-solver/description/?envType=problem-list-v2&envId=backtracking>**

**2. [Word Ladder II](https://leetcode.com/problems/word-ladder-ii/)**

A transformation sequence from word beginWord to word endWord using a dictionary wordList is a sequence of words beginWord -> s1 -> s2 -> ... -> sk such that:

Every adjacent pair of words differs by a single letter.

Every si for 1 <= i <= k is in wordList. Note that beginWord does not need to be in wordList.

sk == endWord

Given two words, beginWord and endWord, and a dictionary wordList, return all the shortest transformation sequences from beginWord to endWord, or an empty list if no such sequence exists. Each sequence should be returned as a list of the words [beginWord, s1, s2, ..., sk].

**Example 1:**

**Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log","cog"]**

**Output: [["hit","hot","dot","dog","cog"],["hit","hot","lot","log","cog"]]**

**Explanation: There are 2 shortest transformation sequences:**

**"hit" -> "hot" -> "dot" -> "dog" -> "cog"**

**"hit" -> "hot" -> "lot" -> "log" -> "cog"**

**Example 2:**

**Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log"]**

**Output: []**

**Explanation: The endWord "cog" is not in wordList, therefore there is no valid transformation sequence.**

**Constraints:**

**1 <= beginWord.length <= 5**

**endWord.length == beginWord.length**

**1 <= wordList.length <= 500**

**wordList[i].length == beginWord.length**

**beginWord, endWord, and wordList[i] consist of lowercase English letters.**

**beginWord != endWord**

**All the words in wordList are unique.**

**The sum of all shortest transformation sequences does not exceed 105.**

**Reference[:https://leetcode.com/problems/word-ladder-ii/description/?envType=problem-list-v2&envId=backtracking](https://leetcode.com/problems/word-ladder-ii/description/?envType=problem-list-v2&envId=backtracking)**

**3.[Word Break II](https://leetcode.com/problems/word-break-ii/)**

Given a string s and a dictionary of strings wordDict, add spaces in s to construct a sentence where each word is a valid dictionary word. Return all such possible sentences in any order.

Note that the same word in the dictionary may be reused multiple times in the segmentation.

**Example 1:**

**Input: s = "catsanddog", wordDict = ["cat","cats","and","sand","dog"]**

**Output: ["cats and dog","cat sand dog"]**

**Example 2:**

**Input: s = "pineapplepenapple", wordDict = ["apple","pen","applepen","pine","pineapple"]**

**Output: ["pine apple pen apple","pineapple pen apple","pine applepen apple"]**

**Explanation: Note that you are allowed to reuse a dictionary word.**

**Example 3:**

**Input: s = "catsandog", wordDict = ["cats","dog","sand","and","cat"]**

**Output: []**

**Constraints:**

**1 <= s.length <= 20**

**1 <= wordDict.length <= 1000**

**1 <= wordDict[i].length <= 10**

**s and wordDict[i] consist of only lowercase English letters.**

**All the strings of wordDict are unique.**

**Input is generated in a way that the length of the answer doesn't exceed 105.**

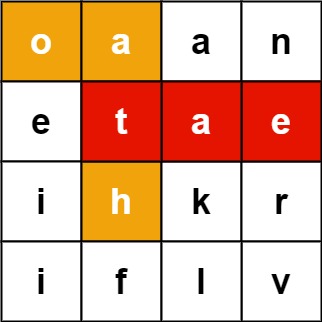
**Reference: <https://leetcode.com/problems/word-break-ii/description/?envType=problem-list-v2&envId=backtracking>**

1. **[Word Search II](https://leetcode.com/problems/word-search-ii/)**

Given an m x n board of characters and a list of strings words, return all words on the board.

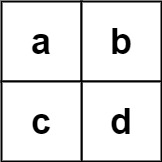
Each word must be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or vertically neighboring. The same letter cell may not be used more than once in a word.

**Example 1:**

****

**Input: board = [["o","a","a","n"],["e","t","a","e"],["i","h","k","r"],["i","f","l","v"]], words = ["oath","pea","eat","rain"]Output: ["eat","oath"]**

**Example 2:**

****

**Input: board = [["a","b"],["c","d"]], words = ["abcb"]Output: []**

**Constraints:**

**m == board.length**

**n == board[i].length**

**1 <= m, n <= 12**

**board[i][j] is a lowercase English letter.**

**1 <= words.length <= 3 \* 104**

**1 <= words[i].length <= 10**

**words[i] consists of lowercase English letters.**

**All the strings of words are unique.**

**Reference: <https://leetcode.com/problems/word-search-ii/description/?envType=problem-list-v2&envId=backtracking>**

1. **[Remove Invalid Parentheses](https://leetcode.com/problems/remove-invalid-parentheses/)**

Given a string s that contains parentheses and letters, remove the minimum number of invalid parentheses to make the input string valid.

Return a list of unique strings that are valid with the minimum number of removals. You may return the answer in any order.

**Example 1:**

**Input: s = "()())()"**

**Output: ["(())()","()()()"]**

**Example 2:**

**Input: s = "(a)())()"**

**Output: ["(a())()","(a)()()"]**

**Example 3:**

**Input: s = ")("**

**Output: [""]**

**Constraints:**

**1 <= s.length <= 25**

**s consists of lowercase English letters and parentheses '(' and ')'.**

**There will be at most 20 parentheses in s.**

**Reference:<https://leetcode.com/problems/remove-invalid-parentheses/description/?envType=problem-list-v2&envId=backtracking>**