HARD QUESTIONS

1. **Longest Common Subsequence**: Given two strings, find the length of the longest subsequence present in both of them.
2. **Maximum Subarray**: Find the contiguous subarray within an array that has the largest sum.
3. **Merge K Sorted Lists**: Merge k sorted linked lists and return it as one sorted list.
4. **Regular Expression Matching**: Implement regular expression matching with support for '.' and '*'.
5. **Minimum Window Substring**: Given a string S and a string T, find the minimum window in S that contains all the characters in T.
6. **Alien Dictionary**: Given a list of words sorted in a specific order, verify if a given word is valid.
7. **Maximum Product Subarray**: Find the contiguous subarray within an array containing at least one number that has the largest product.
8. **Word Break**: Given a non-empty string s and a dictionary wordDict containing a list of non-empty words, determine if s can be segmented into a space-separated sequence of one or more dictionary words.
9. **Distinct Subsequences**: Given two strings s and t, return the number of distinct subsequences of s which equals t.
10. **Trapping Rain Water**: Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.
11. **Word Search II**: Given an m x n board of characters and a list of words, find all words in the board.
12. **Palindrome Partitioning II**: Given a string s, partition s such that every substring of the partition is a palindrome. Return the minimum cuts needed for a palindrome partitioning of s.
13. **Find Median from Data Stream**: Design a data structure that supports the following two operations: addNum() which adds integers to the data structure, and findMedian() which returns the median of all the elements added so far.
14. **Serialize and Deserialize Binary Tree**: Design an algorithm to serialize and deserialize a binary tree.
15. **Best Time to Buy and Sell Stock III**: Design an algorithm to find the maximum profit.

16. **Substring with Concatenation of All Words**: Given a string s and a list of words, find all starting indices of substring(s) in s that is a concatenation of each word in words exactly once and without any intervening characters.

17. **First Missing Positive**: Given an unsorted integer array, find the smallest missing positive integer.

18. **Maximum Frequency Stack**: Implement FreqStack, a class which simulates the operation of a stack-like data structure.

19. **Jump Game II**: 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. Your goal is to reach the last index in the minimum number of jumps.

20. **Longest Valid Parentheses**: Given a string containing just the characters '(' and ')', find the length of the longest valid (well-formed) parentheses substring.

21. **Binary Tree Maximum Path Sum**: Given a non-empty binary tree, find the maximum path sum.

22. **Unique Paths II**: 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). Now consider if some obstacles are added to the grids. How many unique paths would there be?

23. **Count of Smaller Numbers After Self**: You are given an integer array nums and you have to return a new counts array. The counts array has the property where counts[i] is the number of smaller elements to the right of nums[i].

24. **Meeting Rooms II**: Given an array of meeting time intervals consisting of start and end times [[s1,e1],[s2,e2],...], find the minimum number of conference rooms required.

25. **Number of Islands**: Given an m x n 2D binary grid which represents a map of '1's (land) and '0's (water), return the number of islands.

26. **Shortest Palindrome**: Given a string s, you are allowed to convert it to a palindrome by adding characters in front of it. Find and return the shortest palindrome you can find by performing this transformation.

27. **Count of Range Sum**: Given an integer array nums, return the number of range sums that lie in [lower, upper] inclusive.

28. **Paint House II**: There are a row of n houses, each house can be painted with one of the k colors. The cost of painting each house with a certain color is different. You have to paint all the houses such that no two adjacent houses have the same color.

29. **Wildcard Matching**: Implement wildcard pattern matching with support for '?' and '*'.

30. **Edit Distance**: Given two words word1 and word2, find the minimum number of operations required to convert word1 to word2.

31. **Minimum Window Subsequence**: Given strings S and T, find the minimum (contiguous) substring W of S, so that T is a subsequence of W.

32. **Maximum Length of Repeated Subarray**: Given two integer arrays A and B, return the maximum length of an subarray that appears in both arrays.

33. **Median of Two Sorted Arrays**: There are two sorted arrays nums1 and nums2 of size m and n respectively. Find the median of the two sorted arrays.

34. **Maximum Gap**: Given an unsorted array, find the maximum difference between the successive elements in its sorted form.

35. **Valid Number**: Validate if a given string can be interpreted as a decimal number.

36. **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.

37. **Subsets II**: Given a collection of integers that might contain duplicates, return all possible subsets.

38. **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.

39. **Search in Rotated Sorted Array**: Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand. You are given a target value to search. If found in the array, return its index, otherwise return -1.

40. **Find K Pairs with Smallest Sums**: You are given two integer arrays nums1 and nums2 sorted in ascending order and an integer k. Define a pair (u,v) which consists of one element from the first array and one element from the second array. Find the k pairs (u1,v1),(u2,v2) ...(uk,vk) with the smallest sums.

41. **Split Array Largest Sum**: Given an array nums which consists of non-negative integers and an integer m, you can split the array into m non-empty continuous subarrays.

42. **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).

43. **Longest Substring Without Repeating Characters**: Given a string, find the length of the longest substring without repeating characters.

44. **Max Points on a Line**: Given n points on a 2D plane, find the maximum number of points that lie on the same straight line.

45. **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.
46. **Search a 2D Matrix II**: Write an efficient algorithm that searches for a target value in an m x n integer matrix. The matrix has the following properties:
47. **Insert Interval**: Given a set of non-overlapping intervals, insert a new interval into the intervals (merge if necessary).
48. **Find the Duplicate Number**: Given an array nums containing n + 1 integers where each integer is between 1 and n (inclusive), prove that at least one duplicate number must exist.
49. **Implement Trie (Prefix Tree)**: Implement a trie with insert, search, and startsWith methods.
50. **Longest Increasing Path in a Matrix**: Given an integer matrix, find the length of the longest increasing path.
51. **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).
52. **Regular Expression Matching**: Given an input string (s) and a pattern (p), implement regular expression matching with support for '.' and '*'.
53. **Sudoku Solver**: Write a program to solve a Sudoku puzzle by filling the empty cells.
54. **Kth Smallest Element in a Sorted Matrix**: Given a n x n matrix where each of the rows and columns are sorted in ascending order, find the kth smallest element in the matrix.
55. **Interleaving String**: Given s1, s2, s3, find whether s3 is formed by the interleaving of s1 and s2.
56. **Edit Distance**: Given two words word1 and word2, find the minimum number of operations required to convert word1 to word2.
57. **Distinct Subsequences**: Given two strings s and t, return the number of distinct subsequences of s which equals t.
58. **Palindrome Partitioning II**: Given a string s, partition s such that every substring of the partition is a palindrome. Return the minimum cuts needed for a palindrome partitioning of s.
59. **Best Time to Buy and Sell Stock III**: Design an algorithm to find the maximum profit.
60. **Median of Two Sorted Arrays**: There are two sorted arrays nums1 and nums2 of size m and n respectively. Find the median of the two sorted arrays.
61. **Longest Valid Parentheses**: Given a string containing just the characters '(' and ')', find the length of the longest valid (well-formed) parentheses substring.

62. **Word Break**: Given a non-empty string s and a dictionary wordDict containing a list of non-empty words, determine if s can be segmented into a space-separated sequence of one or more dictionary words.

63. **Distinct Subsequences**: Given two strings s and t, return the number of distinct subsequences of s which equals t.

64. **Shortest Palindrome**: Given a string s, you are allowed to convert it to a palindrome by adding characters in front of it. Find and return the shortest palindrome you can find by performing this transformation.

65. **Word Search II**: Given an m x n board of characters and a list of words, find all words in the board.

66. **Serialize and Deserialize Binary Tree**: Design an algorithm to serialize and deserialize a binary tree.

67. **Word Break II**: Given a non-empty string s and a dictionary wordDict containing a list of non-empty words, add spaces in s to construct a sentence where each word is a valid dictionary word. Return all such possible sentences.

68. **Count of Smaller Numbers After Self**: You are given an integer array nums and you have to return a new counts array. The counts array has the property where counts[i] is the number of smaller elements to the right of nums[i].

69. **Maximum Subarray**: Find the contiguous subarray within an array that has the largest sum.

70. **Maximum Product Subarray**: Find the contiguous subarray within an array containing at least one number that has the largest product.

71. **Meeting Rooms II**: Given an array of meeting time intervals consisting of start and end times [[s1,e1],[s2,e2],...], find the minimum number of conference rooms required.

72. **Merge K Sorted Lists**: Merge k sorted linked lists and return it as one sorted list.

73. **Trapping Rain Water**: Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

74. **Maximum Frequency Stack**: Implement FreqStack, a class which simulates the operation of a stack-like data structure.

75. **Jump Game II**: 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. Your goal is to reach the last index in the minimum number of jumps.

76. **Longest Common Subsequence**: Given two strings, find the length of the longest subsequence present in both of them.

77. **Search in Rotated Sorted Array**: Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand. You are given a target value to search. If found in the array, return its index, otherwise return -1.

78. **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.

79. **Find Median from Data Stream**: Design a data structure that supports the following two operations: addNum() which adds integers to the data structure, and findMedian() which returns the median of all the elements added so far.

80. **Binary Tree Maximum Path Sum**: Given a non-empty binary tree, find the maximum path sum.

81. **Longest Substring with At Least K Repeating Characters**: Given a string s and an integer k, return the length of the longest substring of s such that the frequency of each character in this substring is greater than or equal to k.

82. **Regular Expression Matching**: Implement regular expression matching with support for '.' and '*'.

83. **Palindrome Partitioning II**: Given a string s, partition s such that every substring of the partition is a palindrome. Return the minimum cuts needed for a palindrome partitioning of s.

84. **Longest Increasing Path in a Matrix**: Given an integer matrix, find the length of the longest increasing path.

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

86. **Edit Distance**: Given two words word1 and word2, find the minimum number of operations required to convert word1 to word2.

87. **Count of Smaller Numbers After Self**: You are given an integer array nums and you have to return a new counts array. The counts array has the property where counts[i] is the number of smaller elements to the right of nums[i].

88. **Maximum Subarray**: Find the contiguous subarray within an array that has the largest sum.

89. **Meeting Rooms II**: Given an array of meeting time intervals consisting of start and end times [[s1,e1],[s2,e2],...], find the minimum number of conference rooms required.

90. **Merge K Sorted Lists**: Merge k sorted linked lists and return it as one sorted list.

91. **Trapping Rain Water**: Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

92. **Maximum Frequency Stack**: Implement FreqStack, a class which simulates the operation of a stack-like data structure.

93. **Jump Game II**: 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. Your goal is to reach the last index in the minimum number of jumps.

94. **Longest Common Subsequence**: Given two strings, find the length of the longest subsequence present in both of them.

95. **Search in Rotated Sorted Array**: Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand. You are given a target value to search. If found in the array, return its index, otherwise return -1.

96. **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.

97. **Find Median from Data Stream**: Design a data structure that supports the following two operations: addNum() which adds integers to the data structure, and findMedian() which returns the median of all the elements added so far.

98. **Binary Tree Maximum Path Sum**: Given a non-empty binary tree, find the maximum path sum.

99. **Longest Substring with At Least K Repeating Characters**: Given a string s and an integer k, return the length of the longest substring of s such that the frequency of each character in this substring is greater than or equal to k.

100. **Regular Expression Matching**: Implement regular expression matching with support for '.' and '*'.

101. **Regular Expression Matching**: Implement regular expression matching with support for '.' and '*'.

102. **Distinct Subsequences**: Given two strings s and t, return the number of distinct subsequences of s which equals t.

103. **Shortest Palindrome**: Given a string s, you are allowed to convert it to a palindrome by adding characters in front of it. Find and return the shortest palindrome you can find by performing this transformation.

104. **Word Search II**: Given an m x n board of characters and a list of words, find all words in the board.

105. **Serialize and Deserialize Binary Tree**: Design an algorithm to serialize and deserialize a binary tree.

106. **Word Break II**: Given a non-empty string s and a dictionary wordDict containing a list of non-empty words, add spaces in s to construct a sentence where each word is a valid dictionary word. Return all such possible sentences.

107. **Count of Smaller Numbers After Self**: You are given an integer array nums and you have to return a new counts array. The counts array has the property where counts[i] is the number of smaller elements to the right of nums[i].

108. **Maximum Subarray**: Find the contiguous subarray within an array that has the largest sum.

109. **Maximum Product Subarray**: Find the contiguous subarray within an array containing at least one number that has the largest product.

110. **Meeting Rooms II**: Given an array of meeting time intervals consisting of start and end times [[s1,e1],[s2,e2],...], find the minimum number of conference rooms required.

111. **Merge K Sorted Lists**: Merge k sorted linked lists and return it as one sorted list.

112. **Trapping Rain Water**: Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

113. **Maximum Frequency Stack**: Implement FreqStack, a class which simulates the operation of a stack-like data structure.

114. **Jump Game II**: 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. Your goal is to reach the last index in the minimum number of jumps.

115. **Longest Common Subsequence**: Given two strings, find the length of the longest subsequence present in both of them.

116. **Search in Rotated Sorted Array**: Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand. You are given a target value to search. If found in the array, return its index, otherwise return -1.

117. **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.

118. **Find Median from Data Stream**: Design a data structure that supports the following two operations: addNum() which adds integers to the data structure, and findMedian() which returns the median of all the elements added so far.

119. **Binary Tree Maximum Path Sum**: Given a non-empty binary tree, find the maximum path sum.

120. **Longest Substring with At Least K Repeating Characters**: Given a string s and an integer k, return the length of the longest substring of s such that the frequency of each character in this substring is greater than or equal to k.