Algorithms & Data Structures Questions

by Ultan

Introduction

Some questions from, or relevant to, the course.

These questions are based mainly on content from the course, Algorithms and Data Structures, but also from Leetcode, linked here. Go to Leetcode, where relevant, to view the full question specifications.

Theory Questions

- What are the cost models?
- List the typical orders of growth giving the name, typical code framework and a description
- Explain big theta, big oh and big omega with examples
- Suppose $T(N) = O(N^2 logN)$. Which of the following are also true?
  - T(N) = O(N^3)
  - T(N) = O(N^2)
  - T(N) = \Omega(N)
  - T(N) = \Omega(N^3)
  - T(N) = \Omega(N^2 log N)
  - T(N) = \Theta(N^2 log N)
- Suppose $T(N) = \Omega(N^2 logN)$. Which of the following are also true?
  - T(N) = O(N^3)
  - T(N) = O(N^2)
  - T(N) = \Omega(N)
  - T(N) = \Omega(N^3)
  - T(N) = \Omega(N^2 log N)
  - T(N) = \Theta(N^2 log N)
- Suppose $T(N) = \Theta(N^2 logN)$. Which of the following are also true?
  - T(N) = O(N^3)
  - T(N) = O(N^2)
  - T(N) = \Omega(N)
  - T(N) = \Omega(N^3)
  - T(N) = \Omega(N^2 log N)
  - T(N) = \Theta(N^2 log N)
- Explain the properties of asymptotic notation with reference to simplification, addition and multiplication
- What is an upper bound, lower bound and optimal algorithm?
- What are Java Generics?

- What is autoboxing?
- What is an iterator?
- What is comparable?
- What are the time complexities of the following?
  - Stack using linked list?
  - Stack using resizing array?
  - Queue using linked list?
  - Queue using resizing array?
  - Priority queue unordered array max?
  - Binary search tree?
- What is amortised?
- Implement a stack
- Implement a queue
- Implement a priority queue
- Implement a binary search tree
- What is the time complexity of selection sort, insertion sort, quicksort and binary search?
- What are the use cases of selection sort, insertion sort, quicksort and binary search?
- Implement binary search
- Implement selection sort
- Implement insertion sort
- Implement quicksort
- Explain briefly the following:
  - Brute-force
  - Decrease and conquer
  - Divide and conquer
  - Transform and conquer
  - Greedy
  - Dynamic programming

Practical Questions

1. Remove Duplicates from a Sorted Array: Given a sorted array *nums*, remove the duplicates in place such that each element appears only *once* and return the new length
2. Best time to Buy and Sell Stock II: Say you have an array prices for which the i element is the price of a stock on a given day. Design an algorithm to find the maximum profit. You may complete as many transactions as you like (i.e., buy one and sell one share of the stock multiple times). You may not engage in multiple transactions at the same time (i.e., you must sell the stock before you buy again)
3. Rotate Array: Given an array, rotate the array to the right by k steps, where k is non-negative
4. Contains Duplicate: Given an array of integers, find if the array contains any duplicates

5. Single Number: Given a non-empty array of integers, every element appears twice except for one. Find that single one
6. Intersection of Two Arrays II: Given two arrays, write a function to compute their intersection. Each element in the result should appear as many times as it shows in both arrays. The result can be in any order
7. Plus One: Given a non-empty array of digits representing a non-negative integer, plus one to the integer. The digits are stored such that the most significant digit is at the head of the list, and each element in the array contain a single digit
8. Move Zeroes: Given an array of nums, write a function to move all 0's to the end of it while maintaining the relative order of the non-zero elements
9. Two Sum: Given an array of integers, return indices of the two number 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
10. Valid Sudoku: Determine if a 9*9 Sudoku board is valid. Only the filled cells need to be validated according to the following rules:
    1. Each row must contain the digits 1-9 without repetition
    2. Each column must contain the digits 1-9 without repetition
    3. Each of the 9 3*3 sub-boxes of the grid must contain the digits 1-9 without repetition
11. Rotate Image: You are given an n*n 2D matrix representing an image. Rotate the image by 90 degrees clockwise
12. Reverse String: Write a function that reverses a string. The input string is given as an array of characters
13. Reverse Integer: Given a 32-bit signed integer, reverse the digits of an integer
14. First Unique Character in a String: Given a string, find the first non-repeating character in it and return its index. If it doesn't exits, return -1
15. Valid Anagram: Given two strings s and t, write a function to determine if t is an anagram of s. You may assume the string contains only lowercase alphabets
16. Valid Palindrome: Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases
17. String to Integer (atoi): Convert a string to an integer
18. Implement strStr(): Return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack
19. Count and Say: The count-and-stay sequence is the sequence of integers with the first five terms as shown. Given an integer n where 1 <= n <= 30, generate the nth term of the count-and-say-sequence. You can do so recursively
    1. 1
    2. 11
    3. 21
    4. 1211
    5. 111221
20. Longest Common Prefix: Write a function to find the longest common prefix string amongst an array of strings
21. Delete Node in a Linked List: Write a function to delete a node (except the tail) in a singly linked list, given only access to that node

22. Delete Nth Node From End of List: Given a linked list, remove the nth node from the end of list and return its head
23. Reverse Linked List: Reverse a singly linked list
24. Merge Two Sorted Lists: Merge two sorted linked lists and return it as a new list. The new list should be made by splicing together the nodes of the first two lists
25. Palindrome Linked List: Given a singly linked list, determine if it is a palindrome
26. Linked List Cycle: Given a linked list, determine if it has a cycle in it. To represent a cycle in the given linked list, we use an integer pos which represents the position in the linked list where tail connects to. If pos is -1 then there is no cycle in the linked list
27. Maximum Depth of Binary Tree: Given a binary tree, find its maximum depth
28. Validate Binary Tree: Given a binary tree, determine if it is a valid binary search tree
29. Symmetric Tree: Given a binary tree, check whether it is a mirror of itself (i.e. symmetric around its centre)
30. Binary Tree Level Order Traversal: Given a binary tree, return the level order traversal of its nodes' values (i.e. from left to right, level by level)
31. Convert Sorted Array to Binary Search Tree: Given an array where elements are sorted in ascending order, convert it to a height balanced BST
32. Merge Sorted Array: Given two sorted integer arrays nums1 and nums2, merge nums2 into nums1 as one sorted array
33. First Bad Version: Suppose you have n versions, [1,2,...,n] and you want to find out the first bad one, which causes all the following ones to be bad. You are given an API bool isBadVersion(version) which will return whether version is bad. Implement a function to find the first bad version
34. Climbing Stairs: You are climbing a staircase. It takes n steps to reach to the top. Each time you can either climb 1 or 2 steps. In how may distinct ways can you climb to the top
35. Best Time to Buy and Sell Stocks: Say you have an array for which the *i* th element is the price of a given stock on day *i*. If you were only permitted to complete at most one transaction (i.e., buy one and sell one share of the stock), design an algorithm to find the maximum profit. Note that you cannot sell a stock before you buy one.
36. Maximum Subarray: Given an integer array nums, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum
37. House Robber: Two adjacent houses contact the police. Given a list of non-negative integers representing the amount of money of each house, determine the max amount to money you can rob without alerting the police
38. Shuffle and Array: Shuffle a set of numbers without duplicates
39. Min Stack: Design a stack that supports push, pop, top, and retrieving the min element in constant time
40. Fizz Buzz: Write a program that outputs the string representation of numbers 1 to n. But for multiple of three it should output "Fizz" instead of the number and for multiples of five output "Buzz". For numbers which are multiples of both three and five output "FizzBuzz"
41. Count Primes: Count the number of prime numbers less than a non-negative number, n
42. Power of Three: Given an integer, write a function to determine if it is a power of three
43. Roman to Integer: Convert a roman numeral to an integer. Numerals usually written largest to smallest, left to right. Roman symbols are -

    1. I for 1

2. V for 5
3. X for 10
4. L for 50
5. C for 100
6. D for 500
7. M for 1000

Exceptions:

1. IV for 4
2. IX for 9
3. XL for 40
4. XC for 90
5. CD for 400
6. CM for 900

44. Number of 1 bits: Write a function that takes an unsigned integer and returns the number of 1 bits it has (i.e. the Hamming weight)
45. Hamming Distance: HD is the number of positions between two integers in which the corresponding bits are different. Given two integers x and y, calculate the Hamming distance
46. Reverse Bits: Reverse bits of a 32 bits unsigned integer
47. Pascal's Triangle: Given a non-negative integer numRows, generate the first numRows of Pascal's triangle. In Pascal's triangle, each number if the sum of the two numbers directly above it
48. Valid Parentheses: Given a string containing just the characters (,),{,},[,], determine if the input string is valid
49. Missing Number: Given an array containing n distinct numbers taken from 0,1,2,...,n, find the one that is missing from the array