Questions - Group 1

Arrays:

1. Find the element that appears once in a sorted array where all other elements appear twice one after another. Find that element in 0(logn) complexity.

Input: arr[] = {1, 1, 3, 3, 4, 5, 5, 7, 7, 8, 8}

Output: 4

1. A magic index in an array A[0…n-1] is defined to be an index such that A[i] = i. Given a sorted array of distinct integers, write a method to find a magic index if one exists, in an array A. FOLLOW UP: What if the values are not distinct?
2. Given a sorted array of n integers that has been rotated an unknown number of times, write code to find an element in the array. You may assume that the array was originally sorted in increasing order.
3. Given an array that contains numbers from 1 to n-1 and exactly 1 duplicate, find that duplicate.
4. Search an element in an array where difference between adjacent elements is 1.

For example: arr[] = {8, 7, 6, 7, 6, 5, 4, 3, 2, 3, 4, 3}

Search for 3 and Output: Found at index 7

1. Given an array of numbers, split the array into two where one array contains the sum of n-1 numbers and the other array with all the n-1 elements.

Stack:

1. Find missing parenthesis in a given expression – 2 \* ( 3 + 5(sasdfasdfasd)
2. Evaluate an expression given only single digits and only 2 operators \* and +.
3. Reverse a stack and put the reversed value back in the same stack. You can use only one other stack and a temp variable.

Strings:

1. Implement an algorithm to determine if a string has all unique characters. What if you cannot use additional data structures?
2. Given two strings, write a method to decide if one is a permutation of the other?
3. Write a method to replace all spaces in a string with ‘%20’.
4. Implement a method to perform a basic string compression using the counts of repeated characters. For example, the string aabccccaaa would become a2b1c4a3. If the compressed string would not become smaller than the original string, your method should return the original string.
5. Write an algorithm such that if an element in an MxN matrix is 0, its entire row and column are set to 0.
6. Given two sequences, print longest common subsequence

LCS for input Sequences “ABCDGH” and “AEDFHR” is “ADH” of length 3.  
LCS for input Sequences “AGGTAB” and “GXTXAYB” is “GTAB” of length 4.

1. Given two string str1 and str2, find the shortest string that has both str1 and str2 as subsequences.

Examples:

Input: str1 = “geek”, str2=”eke”

Output: “geeke”

Input: str1 = “AGGTAB”, str2=”GXTXAYB”

Output: “AGXGTXAYB”

1. Remove spaces from a given string in O(n) running time and only one traversal of a string.

Input: “I love ice cream”

Output: “Iloveicecream”

1. Find all distinct palindromic sub-strings of a given string

Input: str = "abaaa"

Output: Below are 5 palindrome sub-strings

a

aa

aaa

aba

b

Linked lists:

1. Write an algorithm to determine if a linkedlist is a palindrome
2. Write an algorithm to determine if a linkedlist is circular. FOLLOW UP: Determine where the circle meets.
3. Clone a linked list with a random pointer.
4. Write code to remove duplicates from an unsorted linked list. Follow up: How would you solve it if temporary buffer is not allowed?
5. Implement an algorithm to find the kth to the last element of a singly linked list
6. Implement an algorithm to delete a node a singly linked list, given only access to that node.
7. Write code to partition a linkedlist around a value x, such that all nodes less than x come before all nodes great than or equal to x.
8. Reverse a linked list – iteratively and recursively
9. Given a singly linked list which has data sorted in ascending order, construct a balanced BST.

Trees:

1. Given a BST, create a linkedlist of all the nodes at each depth
2. Convert a BST into a doubly linkedlist.
3. Determine if a binary tree is balanced
4. Given a sorted array, create a binary search tree with minimal height
5. Implement a function to check if a binary tree is a BST
6. Write an algorithm to find the next node (i.e in-order successor) of a given node in a binary search tree. You may assume that each node has a link to its parent
7. Design an algorithm and write code to find the first common ancestor of two nodes in a binary tree. Avoid storing additional nodes in a data structure. NOTE: This is not necessarily a BST
8. You have 2 very large binary trees: T1 with millions of nodes, and T2 with hundreds of nodes. Create an algorithm to decide if T2 is a subtree of T1. A tree T2 is a subtree of T1 if there exists a node n in T1 such that the subtree of n is identical to T2. That is, if you cut off the tree at node n, the two trees would be identical.
9. You are given a binary tree in which each node contains a value. Design an algorithm to print all paths which sum to a given value. Note that a path can start or end anywhere in the tree.

Graph:

1. Determine if a path exists from a source node to a destination node
2. Print all paths from a given source to a destination

Recursion and Dynamic programming:

1. Write an algorithm to print all ways of arranging eight queens on an 8x8 chess board so that none of them share the same row, column or diagonal. In this case, “diagonal” means all diagonals, not just the two that bisect the board.

Bit manipulation:

1. Given an array that contains numbers from 1 to n-1 and exactly 1 duplicate, find that duplicate using bit operations.
2. Find the number of on(1) bits in a given number. For example, numbr 5 has 2 bits that are 1. Number 7 has 3 bits that are on.
3. Subtract two numbers without using arithmetic operators
4. How to turn off a particular bit in a number?
5. Swap two nibbles in a byte.
6. Swap all odd and even bits.
7. Binary representation of a given number
8. Given an array where every element occurs three times, except one element which occurs only once. Find the element that occurs once. Expected time complexity is O(n) and O(1) extra space.  
   Examples:

Input: arr[] = {12, 1, 12, 3, 12, 1, 1, 2, 3, 3}

Output: 2

Other:

1. Given any integer, print an English phrase that describes the integer (e.g., “One Thousand, Two Hundred Thirty Four”)
2. How to determine if a number is a power of 2.
3. Find a specific row and a column value in a Pascal’s triangle