Advanced Algorithmic Problem Solving (R1UC601B)

PRACTICE QUESTIONS FOR ETE

1. Analyse how different hashing methods can be applied to optimize search and sort operations.
2. Analyse the performance and application scenarios of binomial and Fibonacci heaps.
3. Investigate the disjoint set union data structure and its operations, evaluating its efficiency in different applications.
4. Compare and contrast Depth-First Search (DFS) and Breadth-First Search (BFS) in various contexts.
5. Evaluate shortest path algorithms, minimum spanning tree algorithms, and their applications in real-world problems.
6 Investigate the significance of articulation points and bridges in network design and reliability.
7 Examine Strasson's matrix multiplication algorithm and compare it with conventional methods.
8 Analyse the time complexity and efficiency of algorithms based on divide and conquer, such as counting inversions and finding the closest pair of points.
9 Critically evaluate the effectiveness of universal hashing in various scenarios.
10 Judge the suitability of hashing methods for optimization problems and justify the chosen method.
11 Judge the effectiveness of shortest path algorithms and minimum spanning tree algorithms in solving real-world problems.
11 Justify the choice of graph algorithms based on problem constraints and requirements.
12 Create a system that dynamically selects the appropriate search method based on the dataset characteristics.
12 Design a robust hashing system that minimizes collisions and optimizes search and sort operations.
13 Design greedy algorithms tailored to solve specific optimization problems like activity selection and task scheduling.
14 Apply the binary search technique to find the first occurrence of a number in a sorted array.
15 Apply the greedy technique to solve the activity selection problem.
16 How would you apply stack operations to evaluate a postfix expression? Write the function in C/C++/Java/Python.
17 Apply binary search tree operations to insert and find an element. Write the function in C/C++/Java/Python.
18 Use BFS to implement a level order traversal of a binary tree. Write the function in C/C++/Java/Python.
19 Write a program that uses divide and conquer to find the closest pair of points in a 2D plane.
20 Write a program to implement dynamic programming to solve the 0/1 knapsack problem and analyse the memory usage.
21 Evaluate the efficiency of using a sliding window technique for a given dataset of temperature readings over brute force methods.
22 Given a number n, find sum of first *n* natural numbers. To calculate the sum, we will use a recursive function recur_sum().

Advanced Algorithmic Problem Solving (R1UC601B)

23 Implement a recursive algorithm to solve the Tower of Hanoi problem. Find its complexity also.

24 Given a Binary Search Tree and a node value X, find if the node with value X is present in the BST or not.

25 Given two Binary Search Trees. Find the nodes that are common in both of them, ie- find the intersection of the two BSTs.

26 Design and implement a version of quicksort that randomly chooses pivot elements. Calculate the time and space complexity of algorithm.

27 Design and implement an efficient algorithm to merge k sorted arrays.

28 Given two strings str1 & str 2 of length n & m respectively, find the length of the longest subsequence present in both. A subsequence is a sequence that can be derived from the given string by deleting some or no elements without changing the order of the remaining elements. For example, "abe" is a subsequence of "abcde". Example :Input: n = 6, str1 = ABCDGH and m = 6, str2 = AEDFHR Output: 3 Explanation: LCS for input strings “ABCDGH” and “AEDFHR” is “ADH” of length 3.

29 You are given an amount denoted by **value**. You are also given an array of coins. The **array** contains the **denominations** of the given coins. You need to find the **minimum number of coins** to make the change for **value** using the coins of given denominations. Also, keep in mind that you have **infinite supply** of the coins. Input:
value = 10
numberOfCoins = 4
coins[] = {2 5 3 6}
Output: 2

30 Hashing is very useful to keep track of the frequency of the elements in a list. You are given an array of integers. You need to print the count of non-repeated elements in the array. Example : Input:1 1 2 2 3 3 4 5 6 7 Output:4

31 Given two arrays a[] and b[] of size n and m respectively. The task is to find the number of elements in the union between these two arrays. Union of the two arrays can be defined as the set containing distinct elements from both the arrays. If there are repetitions, then only one occurrence of element should be printed in the union. Input:1 2 3 4 5 1 2 3 Output: 5

32 Inorder traversal means traversing through the tree in a Left, Node, Right manner. We first traverse left, then print the current node, and then traverse right. This is done recursively for each node. Given a BST, find its in-order traversal.

Advanced Algorithmic Problem Solving (R1UC601B)

PRACTICE QUESTIONS FOR ETE

These problems were given for practice for MTE, you are advised to practice these for ETE also.

1. What is meant by time complexity and space complexity? Explain in detail.
2. What are asymptotic notations? Define Theta, Omega, big O, small omega, and small o.
3. Explain the meaning of O(2^n), O(n^2), O(nlgn), O(lg n). Give one example of each.
4. Explain sliding window protocol.
5. Explain Naive String-Matching algorithm. Discuss its time and space complexity.
6. Explain Rabin Karp String-Matching algorithm. Discuss its time and space complexity.
7. Explain Knuth Morris and Pratt String-Matching algorithm. Discuss its time and space complexity.
8. What is a sliding window? Where this technique is used to solve programming problems.
9. What are bit manipulation operators? Explain and, or, not, ex-or bit-wise operators.
10. Why are the benefits for linked list over arrays.
11. Implement singly linked list.
11. Implement stack with singly linked list.
12. Implement queue with singly linked list.
12. Implement doubly linked list.
13. Implement circular linked list.
14. Implement circular queue with linked list.
15. What is recursion? What is tail recursion?
16. What is the tower of Hanoi problem? Write a program to implement the Tower of Hanoi problem. Find the time and space complexity of the program.
17. What is backtracking in algorithms? What kind of problems are solved with this technique?
18. Implement N-Queens problem. Find the time and space complexity.
19. What is subset sum problem? Write a recursive function to solve the subset sum problem?
20. Implement a function that uses the sliding window technique to find the maximum sum of any contiguous subarray of size K.
21. Write a recursive function to generate all possible subsets of a given set.
22. Write a program to find the first occurrence of repeating character in a given string.
23. Write a program to print all the LEADERS in the array. An element is a leader if it is greater than all the elements to its right side. And the rightmost element is always a leader.
24. Write a program to find the majority element in the array. A majority element in an array A[] of size n is an element that appears more than n/2 times.
25. Given an integer k and a queue of integers, write a program to reverse the order of the first k elements of the queue, leaving the other elements in the same relative order.
26. Write a program to implement a stack using queues.
27. Write a program to implement queue using stacks.
28. Given a string S of lowercase alphabets, write a program to check if string is

isogram or not. An Isogram is a string in which no letter occurs more than once.

29 Given a sorted array, arr[] consisting of N integers, write a program to find the frequencies of each array element.
30 Write a program to delete middle element from stack.
31 Write a program to remove consecutive duplicates from string.
33 Write a program to display next greater element of all element given in array.
34 Write a program to evaluate a postfix expression.
35 Write a program to get MIN at pop from stack.
36 Write a program to swap k^{th} node from ends in given single linked list.
37 Write a program to detect loop in linked list
38 Write a program to find Intersection point in Y shaped Linked list.
39 Write a program to merge two sorted linked list.
40 Write a program to find max and second max of array.
41 Write a program to find Smallest Positive missing number. You are given an array arr[] of N integers. The task is to find the smallest positive number missing from the array. Positive number starts from 1.
42 Given a non-negative integer N. The task is to check if N is a power of 2. More formally, check if N can be expressed as 2x for some integer x. Return true if N is power of 2 else return false.
43 Write a program to Count Total Digits in a Number using recursion. You are given a number n. You need to find the count of digits in n.
44 Check whether K-th bit is set or not. Given a number **N** and a bit number **K**, check if **Kth** index bit of **N** is set or not. A bit is called set if it is 1. Position of set bit '1' should be indexed starting with 0 from LSB side in binary representation of the number. Index is starting from 0. You just need to return **true** or **false**.
45 Write a program to print 1 To N without loop.