MCA 2nd Semester

KCA205: DATA STRUCTURES & ANALYSIS OF ALGORITHMS

Important Questions

Unit-I

1. Define the terms "data," "entity," and "information" in the context of data structures.
2. Define abstract data type (ADT) and explain its significance in data structures. Provide an example of an ADT and its associated operations.
3. What is a data structure? Discuss the importance of data structures in programming and distinguish between linear and non-linear data structures. Provide examples of each type and explain their characteristics.
4. Define an algorithm and differentiate it from a program. Discuss the key properties of an algorithm. Explain the algorithm design techniques such as divide and conquer, dynamic programming, greedy algorithms and backtracking. Provide examples to illustrate each technique.
5. What do you mean by Analysis of Algorithms? Discuss the Asymptotic notations used in analysis of algorithms.
6. Define an array and discuss its applications in programming. Explain the concepts of single-dimensional and multidimensional arrays. Discuss the representation of arrays

in row major order and column major order. Also discuss the address formulae for 1-D and 2-D arrays.

7. Explain the concept of sparse matrices and their representation. Discuss how sparse matrices can be stored efficiently in memory.
8. Describe the array implementation and pointer implementation of singly linked lists. Explain the operations (insertion, deletion, traversal) on a linked list.
9. Discuss the concept of a doubly linked list. Explain its advantages over a singly linked list and describe the operations that can be performed on a doubly linked list.
10. Explain the concept of a circularly linked list. Discuss its characteristics and the benefits it offers compared to linear linked lists.
11. Describe the operations (insertion, deletion, traversal) that can be performed on linked lists.
12. Explain the polynomial representation of a linked list. Discuss how addition, subtraction, and multiplication of single-variable polynomials can be performed using linked lists.

Unit - II

Stacks:

1. Define the abstract data type (ADT) "stack" and explain its characteristics. Discuss the importance of stack data structure in programming.

2. Explain the primitive stack operations: push, pop and peek using array and Linked list.
3. Describe the algorithms of converting an infix expression to postfix and prefix expressions using an example.
4. With the help of an example, explain the step-by-step process of evaluating a postfix expression using a stack.
5. What is recursion? Discuss the different types of recursion. Explain the concept of tail recursion and its significance in recursive algorithms. Provide an example to illustrate the use of tail recursion.
6. Explain the problem of Tower of Hanoi. Discuss the recursive algorithm's steps and complexity.

Queues:

1. What is a Queue? Describe the operations of insertion, deletion and display in a queue checking if the queue is full, and checking if the queue is empty.
2. Explain the concept of circular queues and their advantages over linear queues. Discuss the implementation of circular queues using arrays and linked list.
3. What is a Double-ended queue? Discuss enqueue and dequeue operations in a double ended queue.
4. Explain the concept of a priority queue and its applications. Discuss the implementation of a priority queue using arrays or linked list.

Searching:

1. Define the concept of searching and its importance in data structures. Discuss sequential search and its implementation.
2. Explain Binary search using a suitable example. Provide step-by-step explanation for each step and also discuss its time complexity.
3. What is Hashing? Discuss the various Collision Resolution Techniques used in Hashing with necessary examples.

<u>Unit - III</u>

Sorting:

1. Explain the difference between the following:
    1. Internal and External Sorting
    2. In place and out-of-place sorting
    3. Stable and Unstable sorting
    4. Adaptive and Non-adaptive sorting
2. Explain the working of following sorting algorithms with suitable examples and also discuss their time complexity:
    1. Bubble Sort
    2. Selection Sort
    3. Insertion Sort
3. Explain the following linear time sorting algorithms: counting sort, radix sort and bucket sort. Discuss their steps, time complexity, and provide examples to illustrate their working.

4. Discuss the divide and conquer approach to problem-solving. Explain the step by step working of merge sort' and quick sort algorithms with suitable examples and discuss their time complexities.

Graphs:

1. Explain the following terminologies used with Graphs:
    a) Directed Graph
    b) Undirected Graph
    c) Weighted Graph
    d) Connected Graph
    e) Simple Graph
    f) Complete Graph
    g) Spanning Tree
2. Discuss the different ways of Graph representations- Adjacency matrix, Incidence matrix, Adjacency List.
3. Explain the difference between Breadth First Search (BFS) and Depth First Search (DFS) using suitable example.
4. Explain the concept of connected components in a graph.

<u>Unit - IV</u>

Trees:

1. Explain the following terms related to Trees:
    a) Degree, Internal, External Nodes
    b) Height and Depth of a Tree
    c) Binary Tree
    d) Strict Binary Tree

   e) Perfect Binary Tree
   f) Complete Binary Tree
   g) Extended Binary Tree
   h) Binary Search Tree
   i) Threaded Binary Tree
   j) AVL Tree
   k) B Tree
2. Discuss the representation of trees in memory using Arrays and Linked lists using a suitable example.
3. Explain the following traversals in context of trees using examples: Inorder, Preorder, Postorder, Levelorder.
4. Discuss the process of constructing a binary tree from a given set of tree traversal sequences.
5. Explain the operations of searching, insertion, modification and deletion of data in a binary search tree using examples.
6. Discuss the various rotations performed in an AVL Tree.
7. Explain the operations of searching, insertion, modification and deletion of data in an AVL tree using examples.
8. Compare a BST and an AVL tree generated from first 10 natural numbers.
9. Illustrate how Huffman coding uses binary trees for efficient data compression.
10. Discuss the advantages of Threaded binary trees and AVL trees.
11. Explain the concept of B-trees. Discuss the characteristics and properties of B-trees. Also discuss the advantages of B-trees over other data structures for storing and managing large amounts of data.

12. Create B trees of order 3, 4 and 5 with first 15 natural numbers.

Unit - V

1. Explain the following concepts using appropriate examples:
    a) Divide and Conquer
    b) Greedy Programming
    c) Dynamic Programming
2. Describe Strassen's algorithm of Matrix Multiplication. Also discuss the time complexity of this method.
3. Explain the concept of Minimum Spanning Tree (MST). Explain Kruskal's and Prim's algorithm using an appropriate example to generate MST from a given graph. Also write the complexities of both.
4. Describe Dijkstra's algorithm of finding Single source shortest path using examples of directed and undirected graph. Also discuss its time and space complexity.
5. Describe Bellman Ford algorithm of finding Single source shortest path using examples of directed and undirected graph. Also discuss its time and space complexity.
6. Compare(similarities) and Contrast(differences) Dijkstra and Bellman Ford algorithm.
7. Explain the working of Floyd Warshall All-pair shortest path algorithm using a suitable example.
8. What is Longest Common Subsequence (LCS)? Describe the Dynamic programming approach to find LCS from a given sequence of characters.