Department :CSE, AIML, AI & DS

Year & Semester :II YEAR & III SEM

Sub Code & Sub Name : 23CSE231& ADVANCED DATA STRUCTURES AND ALGORITHM ANALYSIS

Unit-I

S.No	Part-A Questions
1.	What is an AVL Tree?
2.	Define Algorithm Analysis?
3.	Define time complexity?
4.	Define space complexity?
5.	What is the difference between time complexity and space complexity?
6.	State the purpose of Asymptotic Notations in algorithm analysis?
7.	Define the balance factor in an AVL Tree.
8.	What is a B-Tree?
9.	Mention two applications of AVL Trees.
10.	Mention two applications of B-Trees.
11.	What is the significance of self-balancing in AVL Trees?
12.	What are the three common Asymptotic Notations?
13.	Why are rotations needed in AVL trees?
14.	Mention two key properties of B-Trees?
15.	Mention the types of AVL rotations?

S.No	Part-B Questions
1.	Explain the concepts of Time and Space Complexity analysis with suitable examples. Discuss the importance of analyzing algorithms.
2.	Describe the various Asymptotic Notations (Big O, Big Omega, Big Theta) used in algorithm analysis. Illustrate with examples how each notation represents the growth rate of an algorithm.
3.	Explain the creation and insertion operations in an AVL Tree. Detail the different rotation types (LL, RR, LR, RL) required to maintain balance during insertion.
4.	Describe the deletion operation in an AVL Tree. Explain how balance is maintained after deletion, including the necessary rotations.

5.	Explain the creation and insertion operations in a B-Tree. Discuss how the properties of a B-Tree are maintained during insertion, including node splitting.
6.	Construct an AVL Tree for the following data 21, 26, 30, 9, 4, 14, 28, 18, 15, 10, 2, 3, 7. Write insertion algorithm for AVL Tree.
7.	Compare and contrast AVL Trees and B-Trees. Discuss their respective advantages, disadvantages, and typical application areas.
8.	Illustrate the steps to insert the following keys into an AVL Tree: 50, 30, 70, 60, 80, 90, 85. Show each rotation clearly and draw the final AVL Tree.
9.	Discuss the applications of B-Trees in detail, providing specific scenarios where they are advantageous. Explain why their structure is well-suited for these applications.
10.	Given a sequence of elements, demonstrate the step-by-step process of constructing an AVL Tree by performing insertions and necessary rotations.
11.	Explain the concepts of Time and Space Complexity analysis with suitable examples. Discuss the importance of analyzing algorithms.
12.	Describe the various Asymptotic Notations (Big O, Big Omega, Big Theta) used in algorithm analysis. Illustrate with examples how each notation represents the growth rate of an algorithm.
13.	Consider inserting the following keys 12, 22, 23, 15, 80, 34, 56, 78, 39, 8, 44, 74, 35, 53, 82, 77, 66, 90, 54, 18, 7, 64, 60 in order = 5.
14.	Insert the keys 78, 52, 81, 40, 33, 90, 85, 20 and 38 in this order in an initially empty B-Tree of order = 3.
15.	Describe the concept of B-Tree insertion and deletion of elements by writing an algorithm for both.

Unit-II

S.No	Part-A Questions
1.	Define a Min Heap.
2.	Define a Max Heap.
3.	State two properties of a Heap.
4.	What is the time complexity of insertion in a heap?
5.	Which data structure is used to implement a priority queue?
6.	Mention one real-life application of heaps.
7.	Write the height of a heap with *n* nodes.
8.	Differentiate between complete binary tree and heap.
9.	Define a biconnected component in a graph.
10.	What is an articulation point?
11.	State one difference between connected and biconnected graphs.
12.	Give an example of a biconnected graph.
13.	What is the divide-and-conquer technique?
14.	Give two examples of algorithms that use divide and conquer.
15.	Write the general steps of the divide-and-conquer approach.
16.	Name one advantage of divide and conquer.
17.	Define merge sort.
18.	Write the best-case and worst-case time complexity of merge sort.
19.	Is merge sort stable? Why?
20.	Which sorting technique is based on divide and conquer: quick sort or merge sort?

S.No	Part-B Questions
1.	Construct a Max Heap for the sequence: 10, 20, 5, 6, 1, 8, 9. Show all steps.
2.	Construct a Min Heap for the sequence: 40, 30, 35, 15, 10, 25. Show step-by-step.
3.	Explain biconnected components with an example graph.
4.	Explain the role of depth-first search (DFS) in finding biconnected components.
5.	Discuss applications of biconnected components in network design.
6.	Explain the divide-and-conquer paradigm with suitable examples.
7.	Discuss the time complexity analysis of divide-and-conquer algorithms
8.	Apply Quick Sort on **[40, 35, 20, 50, 10, 70, 45]** with the first element as pivot. Show partitioning steps clearly
9.	Sort the array **[38, 27, 43, 3, 9, 82, 10]** using Merge Sort. Show all merging steps.
10.	Explain the merge sort algorithm with an example

Unit-III

S.No	Part-A Questions
1.	What is the greedy method?
2.	State one advantage and one limitation of the greedy approach
3.	What is the objective of the Job Sequencing with Deadlines problem?
4.	What data structures are typically used in job sequencing problems?
5.	What is the difference between 0/1 Knapsack and Fractional Knapsack?
6.	In what case does the greedy method give an optimal solution for the knapsack problem?
7.	Define a spanning tree.
8.	Name two algorithms used to find the minimum cost spanning tree.
9.	When does Dijkstra's algorithm fail to produce correct results?
10.	What data structure is commonly used to improve the efficiency of Dijkstra's algorithm?
11.	What are the two main properties required for dynamic programming?
12.	Define overlapping subproblems with an example.
13.	Which algorithm is used for finding all pairs shortest paths in a weighted graph?
14.	Can Floyd-Warshall algorithm handle negative edge weights?
15.	How does Bellman-Ford handle negative edge weights?
16.	What is the time complexity of Bellman-Ford algorithm?
17.	What is an Optimal Binary Search Tree (OBST)?
18.	What is the cost metric used in OBST?
19.	State the recurrence relation for 0/1 Knapsack using dynamic programming.
20.	How is 0/1 knapsack different from fractional knapsack?
21.	What operations are allowed in computing edit distance?
22.	Write the recurrence relation used for edit distance calculation.
23.	Is TSP solvable in polynomial time using dynamic programming?
24.	What is the time complexity of TSP using dynamic programming?

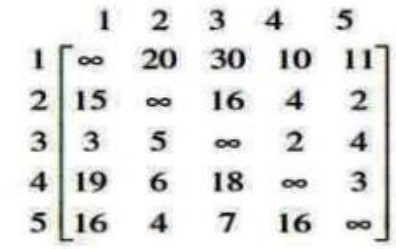
S.No	Part-B Questions
1.	Explain the general method of greedy algorithms with a suitable example.
2.	Compare greedy strategy with dynamic programming. Provide examples where greedy fails but dynamic programming works.
3.	Solve the Job Sequencing with Deadlines problem for the following jobs with profits and deadlines. Show all steps clearly.
4.	Explain the greedy solution to the Job Sequencing with Deadlines problem. Analyze its time complexity
5.	Solve the Job Sequencing with Deadlines problem for the following jobs with profits and deadlines. Show all steps clearly.

6.	Explain the greedy solution to the Job Sequencing with Deadlines problem. Analyze its time complexity.
7.	Explain Prim's algorithm for finding the Minimum Cost Spanning Tree. Trace it on a graph.
8.	Discuss Kruskal's algorithm. Show its working with a graph example.
9.	Explain Dijkstra's algorithm in detail. Apply it to a given graph and show the shortest paths.
10.	Compare Dijkstra's and Bellman-Ford algorithms for single source shortest paths.
11.	Describe the general method of solving problems using dynamic programming with a flowchart or pseudocode.
12.	Differentiate between divide and conquer and dynamic programming with examples.
13.	Explain the Floyd-Warshall algorithm for all pairs shortest paths. Illustrate with a step-by-step example.Write the dynamic programming formulation for all-pairs shortest paths.
14.	Explain the working of Bellman-Ford algorithm. Apply it to a given graph and detect negative cycles.
15.	Write the algorithm and discuss how it differs from Dijkstra's algorithm.
16.	Derive the dynamic programming solution for Optimal Binary Search Trees. Use a sample input to construct OBST.
17.	Discuss the significance of OBST and explain the time complexity of constructing it.
18.	Solve the 0/1 knapsack problem using dynamic programming. Include the table used for computation.
19.	Explain the bottom-up and top-down approaches for solving 0/1 knapsack.
20.	Explain the dynamic programming solution to the edit distance problem. Show an example.
21.	Derive the algorithm to convert one string into another using insert, delete, and replace operations.
22.	Formulate the TSP using dynamic programming and solve it for a given cost matrix.
23.	Explain the Held-Karp algorithm for solving TSP using DP. Illustrate with an example.

Unit-IV

S.No	Part-A Questions
1.	Define backtracking and give one example of a problem solved using it.
2.	What is the state-space tree in the context of backtracking?
3.	Give any 3 applications of back tracking?
4.	What is the objective of the 8-Queens problem?
5.	State the constraints for placing queens in the 8-Queens problem.
6.	Define the Sum of Subsets problem.
7.	What is meant by a feasible solution in the context of backtracking?
8.	Explain the concept of graph coloring.
9.	What is the significance of the chromatic number in graph coloring?
10.	Define the 0/1 Knapsack problem.

11.	Differentiate between explicit and implicit constraints in backtracking.
12.	What is the role of bounding functions in backtracking?
13.	Define the Branch and Bound technique.
14.	What is the difference between backtracking and Branch and Bound?
15.	What is the cost function in the context of the Travelling Salesperson Problem (TSP)?
16.	State the difference between a feasible and an optimal solution in Branch and Bound.

S.No	Part-B Questions
1.	Explain the general method of backtracking with a suitable example. Illustrate the steps using a state-space tree
2.	Describe the algorithm for solving the 8-Queens problem using backtracking. Provide a step-by-step solution for a 4x4 chessboard.
3.	Explain the graph coloring problem and describe the backtracking algorithm to solve it. Illustrate with an example graph.
4.	Solve the 0/1 Knapsack problem using backtracking for items with weights {2, 3, 5}, values {30, 40, 50}, and capacity W = 6
5.	Write the backtracking algorithm for the Sum of Subsets problem and explain how it generates all possible subsets
6.	Construct the State space tree for the profits={3,5,6,10} and weights={2,3,4,5},n=4 and m=8 (Capacity). Apply the backtracking for 0/1 Knapsack and also find the Maximum profit.
7.	Explain the general method of Branch and Bound with an example. Discuss how it differs from backtracking.
8.	Solve the 0/1 Knapsack problem using Branch and Bound for items with weights {2, 3, 4, 5}, values {20, 30, 40, 50}, and capacity W = 8
9.	Find the LC branch and bound solution for the traveling sale person problem whose cost matrix is as follows: 
10.	Explain the principles of FIFO branch and bound. Explain the principles of LIFO branch and bound.

Unit-V

S.No	Part-A Questions
1.	Define P class and NP Class.
2.	What are NP complete and NP Hard?
3.	What is Chromatic Number?

4.	What is deterministic algorithm?
5.	What is non- deterministic algorithm?
6.	State Cook's theorem?
7.	What is NP-hard problems?
8.	How to prove a problem is NP and NP-Hard?
9.	What is CDP?
10.	Define the Chromatic Number Decision Problem (CNDP).
11.	State the Traveling Salesperson Decision Problem (TSP).
12.	What is meant by Scheduling Identical Processors problem?
13.	Explain the Job Shop Scheduling problem briefly.
14.	Differentiate between decision and optimization problems in the context of NP Hard problems.
15.	What is the role of nondeterministic algorithms in defining NP problems?

S.No	Part-B Questions
1.	What are the differences between Deterministic and Non-Deterministic Algorithms?
2.	How do you prove a problem is Np-Hard? Explain using example.
3.	Explain Cook's theorem in detail, including its proof outline and significance in establishing NP Completeness.
4.	Describe the Clique Decision Problem (CDP) and prove that it is NP Complete.
5.	Discuss the Chromatic Number Decision Problem (CNDP) and explain why it is NP Hard. Provide an example.
6.	Explain the Traveling Salesperson Decision Problem (TSP) and demonstrate its NP Completeness using a polynomial-time reduction.
7.	Discuss the Scheduling Identical Processors problem and explain why it is NP Hard. Illustrate with a small instance.
8.	Describe the Job Shop Scheduling problem in detail and explain its NP Hard nature. Provide an example with 2 jobs and 2 machines.
9.	Compare and contrast NP Hard and NP Complete problems with examples from graph and scheduling problems.
10.	Explain how polynomial-time reduction is used to prove NP Completeness. Illustrate with a reduction from SAT to the Clique Decision Problem.
11.	Discuss the challenges in solving NP Hard graph problems like CDP and CNDP. Suggest possible approximation approaches.
12.	Explain the relationship between decision and optimization versions of the Traveling Salesperson Problem and discuss their complexity.
13.	Explain Cook's theorem in detail, including its proof outline and significance in establishing NP Completeness.