UNIT IV – TREE ADT – QUESTION BANK

Part A – 2 Marks Questions

1. Define Tree ADT.
2. What is a binary tree?
3. Differentiate between full binary tree and complete binary tree.
4. What is tree traversal? Name the types.
5. What is the time complexity of in-order traversal?
6. Define backtracking with an example use case.
7. What is a Binary Search Tree (BST)?
8. What are the applications of AVL Trees?
9. Define max-heap and min-heap.
10. What is a multi-way search tree?
11. Explain the term ‘balance factor’ in AVL Trees.
12. Define the 0/1 Knapsack problem.
13. What is Branch and Bound method?
14. Write any two properties of heap.
15. What is the difference between DFS and BFS in tree traversal?

Part B – 13 Marks Questions

1. Explain the representation of Binary Trees using arrays and linked structures.
2. Write and explain the algorithms for in-order, pre-order, and post-order traversal of a binary tree.
3. Discuss the concept of backtracking. Implement the N-Queens problem using backtracking.
4. Construct a binary search tree (BST) using the following elements: 50, 30, 20, 40, 70, 60, 80. Perform in-order traversal.
5. Explain insertion and deletion operations in Binary Search Tree with examples.
6. Describe the AVL tree. Explain insertion operation with appropriate rotations.
7. Write an algorithm for building a max-heap and explain heapify procedure.
8. Differentiate between Binary Tree, BST, AVL Tree, and Heap with examples.
9. Explain the working of multi-way search trees. How are they different from binary trees?
10. Solve the 0/1 Knapsack problem using the Branch and Bound technique.
11. Explain any two applications of trees in real-world software systems.
12. Compare Backtracking and Branch & Bound techniques with appropriate examples.
13. Write a C/C++/Python function to insert an element into an AVL tree and balance it.

Part C – 15 Marks Case Study-Based Questions

1. **Case Study – Memory Efficient Storage**:
   A software product needs to implement a symbol table where data needs to be stored

in a memory-efficient and sorted manner with frequent insertions and deletions. Which tree data structure will you choose? Justify with operations and analyze time complexity. Implement the insertion and balancing logic.

2. **Case Study – Task Scheduling using Heaps**:
   A task manager needs to prioritize jobs based on deadlines and penalties. How can a heap structure help? Illustrate how max-heap/min-heap is used in this scenario with insertion and deletion of jobs. Provide simulation and complexity analysis.
3. **Case Study – Real-Time Routing in Navigation Systems**:
   Navigation systems use trees for route optimization and quick lookup. Propose a suitable tree-based data structure and explain how traversal can help in optimal path finding. Include edge cases like dynamic route changes.
4. **Case Study – Decision Trees in AI**:
   Explain how multi-way trees or binary trees can be used to simulate decision-making in AI systems such as chatbots or recommendation engines. Include a sample decision tree and how it is traversed for output.
5. **Case Study – Secure Data Storage using AVL Trees**:
   A banking system needs to ensure secure, consistent, and quick access to customer data. Describe how AVL trees can be implemented to manage this efficiently. Provide code snippets and rotation cases.
6. **Case Study – Resource Allocation Using Knapsack Problem**:
   A hospital is trying to allocate medical kits to departments with limited supply and differing utility values. Model this as a 0/1 Knapsack problem and solve it using Branch and Bound. Explain pruning of nodes and calculation of bounds.