Data Structures and Algorithms Viva Questions and Answers

Prepared by Rahul

For DSA Exam on 4th July

This document is prepared by Rahul for the DSA viva exam on 4th July.

1 Unit I: Introduction and Arrays

1. **What is a data structure?**
   A data structure is a way to store and organize data in a computer so it can be used easily. Examples are arrays, linked lists, stacks, and queues.

2. **What is an algorithm?**
   An algorithm is a step-by-step way to solve a problem or do a task.

3. **What is time complexity?**
   Time complexity tells us how long an algorithm takes to run based on how much data it has to process. We use Big O notation to show it.

4. **What is space complexity?**
   Space complexity tells us how much memory an algorithm uses based on the data size.

5. **What is Big O notation?**
   Big O notation shows the maximum time an algorithm might take. For example, O(n) means time grows with the data size.

6. **What is an array?**
   An array is a list of items stored next to each other in memory. You can find items using their position number (index).

7. **How do you insert an element into an array?**
   Move the items after your chosen spot to the right, then put the new item in that spot.

8. **What is the time complexity of inserting an element into an array?**
   It's O(n) in the worst case because you might have to move all items.

9. **What is bubble sort?**
   Bubble sort is a simple way to sort a list. It compares items next to each other and swaps them if they're in the wrong order, repeating until sorted.

10. **What is the time complexity of bubble sort?**
    It's $O(n^2)$ because it might need to check every pair of items many times.

11. **What is binary search?**
    Binary search finds an item in a sorted list by cutting the list in half each time and checking which half it's in.

12. **What is the time complexity of binary search?**
    It's O(log n) because it keeps dividing the list in half.

2 Unit II: Linked Lists

1. **What is a linked list?**
   A linked list is a chain of nodes where each node has data and a link to the next node.

2. **What are the advantages of linked lists over arrays?**
   Linked lists can grow or shrink as needed, and adding or removing items doesn't need shifting like in arrays.

3. **What is a singly linked list?**
   It's a linked list where each node only points to the next node.

4. **What is a doubly linked list?**
   It's a linked list where each node points to both the next and previous nodes.

5. **How do you insert a node at the beginning of a linked list?**
   Make a new node, point it to the current first node, and make it the new first node.

6. **What is the time complexity of inserting a node at the beginning of a linked list?**
   It's O(1) because it's super quick no matter how big the list is.

7. **What is a circular linked list?**
   It's a linked list where the last node connects back to the first, making a loop.

8. **What is a header linked list?**
   It's a linked list with a special starting node that might hold extra info or just be a placeholder.

3 Unit III: Stacks and Queues

1. **What is a stack?**
   A stack is a list where you add and remove items from the top only, like a stack of plates (Last In, First Out - LIFO).

2. **What are the basic operations of a stack?**
   Push (add to top), Pop (remove from top), and Peek (look at the top without removing).

3. **How can a stack be implemented?**
   You can use an array or a linked list to make a stack.

4. **What is a queue?**
   A queue is a list where you add items at the back and remove from the front, like a line of people (First In, First Out - FIFO).

5. **What are the basic operations of a queue?**
   Enqueue (add to back), Dequeue (remove from front), and Front (look at the front without removing).

6. **What is a priority queue?**
   It's a queue where items have priorities, and higher priority items get removed first.

7. **What is a deque?**
   A deque (double-ended queue) lets you add or remove items from both ends.

8. **What is Polish notation?**
   It's a way to write math expressions with the operator before the numbers, like + 3 4 instead of 3 + 4.

4 Unit IV: Trees and Recursion

1. **What is a tree in data structures?**
   A tree is a structure with a top node (root) and branches to other nodes, like a family tree.

2. **What is a binary tree?**
   It's a tree where each node has up to two kids (left and right).

3. **What is a complete binary tree?**
   A binary tree where all levels are full except maybe the last, which fills left to right.

4. **What is a binary search tree (BST)?**
   It's a binary tree where left kids are smaller and right kids are bigger than their parent.

5. **How do you insert a node into a BST?**
   Start at the top, compare the new value, go left or right until you find an empty spot, then add it there.

6. **What is tree traversal?**
   It's visiting every node in a tree in a certain order.

7. **What are the types of tree traversals?**
   In-order (left, root, right), Pre-order (root, left, right), Post-order (left, right, root).

8. **What is recursion?**
   Recursion is when a function calls itself to solve a smaller problem until it's simple enough.

9. **What is the Tower of Hanoi problem?**
   It's a puzzle where you move disks from one peg to another using recursion, following rules.

10. **What is merge sort?**
    It's a sorting method that splits a list in half, sorts each half, then combines them.

5 Unit V: AVL Trees and Heaps

1. **What is an AVL tree?**
   An AVL tree is a balanced BST where the height difference between left and right sides is at most 1.

2. **Why is balancing important in BSTs?**
   It keeps the tree short so operations stay fast instead of turning into a slow list.

3. **How do you insert a node into an AVL tree?**
   Add it like a BST, then check heights and rotate nodes if it's unbalanced.

4. **What are rotations in AVL trees?**
   Rotations fix the balance by moving nodes around (left, right, or both ways).

5. **What is a heap?**
   A heap is a tree where parents are bigger (max-heap) or smaller (min-heap) than their kids.

6. **What is heap sort?**
   It's a sorting method that builds a heap, then keeps taking the top item to sort the list.

7. **What is a B-tree?**
   A B-tree is a balanced tree for storing lots of data, used in databases for quick access.

8. **What is a B+ tree?**
   It's like a B-tree, but all data is in the bottom nodes, and upper nodes help find it.

6 Unit VI: Graphs and Hashing

1. **What is a graph?**
   A graph is a set of points (vertices) connected by lines (edges), like a map.

2. **What is Warshall's algorithm?**
   It finds if there's a path between any two points in a graph.

3. **What is BFS?**
   Breadth-First Search checks all points at one level before going deeper in a graph.

4. **What is DFS?**
   Depth-First Search goes as far as it can down one path before turning back.

5. **What is the Floyd-Warshall algorithm?**
   It finds the shortest paths between all pairs of points in a graph.

6. **What is hashing?**
   Hashing turns data into a number to store it in a table for fast finding.

7. **What is a hash function?**
   A hash function takes data and makes a number to decide where it goes in a table.

8. **What is a hash table?**
   It's a structure that uses hashing to store and find data quickly.

9. **What is open hashing (separate chaining)?**
   It puts items that clash in a linked list at the same spot in the table.

10. **What is closed hashing (open addressing)?**
    It finds a new spot in the table for clashing items using methods like probing.

7 Practical Experiments

1. **How do you implement insertion and deletion in an array?**
   For insertion, move items right from the spot and add the new one. For deletion, move items left to fill the gap.

2. **What is the difference between linear search and binary search?**
   Linear search checks every item one by one, while binary search cuts a sorted list in half each time.

3. **How do you implement a stack using an array?**
   Use an array and a top pointer. Push adds an item and moves the pointer up, pop removes it and moves it down.

4. **What is the recursive solution for the Tower of Hanoi problem?**
   Move smaller stacks to a spare peg, shift the biggest disk, then move the stack on top of it.

5. **How do you create and traverse a binary tree recursively?**
   Make nodes with left and right kids, then use recursive calls to visit them in order.

8 General Questions

1. **Why is it important to analyze the complexity of algorithms?**
   It tells us how fast or slow an algorithm will be with big data.

2. **What is the difference between a stack and a queue?**
   Stack is last in, first out. Queue is first in, first out.

3. **When would you use a linked list instead of an array?**
   When you need to add or remove items a lot or don't know the size ahead of time.

4. **What is the significance of tree data structures?**
   Trees organize data like a hierarchy and make searching or sorting faster.

5. **How does hashing improve search efficiency?**
   It finds data super fast, usually in O(1), by jumping straight to its spot.