perplexity

DSA PDF".

Below is a comprehensive DSA Viva Preparation Manual based on your **DSA Lab Manual (2023–24, Dr. D.Y. Patil Institute of Technology, Pune)**.
Each topic contains 10–15 commonly asked viva questions with concise, oral-style answers for quick revision.

DSA Viva Preparation Manual

(Oral-Style Q&A for Practical Exams)

1. Hashing and Collision Resolution

1. **What is hashing?**
   Hashing is a technique to map data to a fixed-size table using a hash function for fast access.
2. **What is a hash function?**
   A hash function converts a key into a table index.
3. **What is a collision in hashing?**
   A collision occurs when two keys hash to the same table index.
4. **Name two collision resolution strategies.**
   Linear probing and quadratic probing.
5. **How does linear probing work?**
   It searches for the next empty slot linearly when a collision occurs.
6. **How does quadratic probing work?**
   It searches for an empty slot using a quadratic formula: (hash + i^2) mod table size.
7. **What is double hashing?**
   It uses a second hash function to decide the probe sequence after a collision.
8. **What is a bucket in a hash table?**
   A bucket is a slot in the hash table that can store one or more records.
9. **What is open hashing?**
   Open hashing, or separate chaining, stores colliding elements in a linked list at the same index.
10. **What is closed hashing?**
    Closed hashing stores all elements within the hash table itself, using probing to resolve collisions.
11. **When is rehashing needed?**
    Rehashing is needed when the table is too full, to reduce collisions.

12. **What is load factor?**
    Load factor is the ratio of the number of entries to the table size.
13. **How do you handle deletion in open addressing?**
    Mark the slot as deleted or use a special marker.
14. **What is the main advantage of hashing?**
    It provides constant-time average-case search, insert, and delete.
15. **Give an example of a real-world application of hashing.**
    Telephone directories and password storage.

2. Set Abstract Data Type (ADT)

1. **What is a set?**
   A set is a collection of unique, unordered elements.
2. **List basic operations on a set.**
   Add, remove, contains, size.
3. **What is set union?**
   Union combines all unique elements from two sets.
4. **What is set intersection?**
   Intersection contains only elements present in both sets.
5. **What is set difference?**
   Difference contains elements in the first set but not in the second.
6. **What is a subset?**
   A set A is a subset of B if all elements of A are in B.
7. **What is a proper subset?**
   A is a proper subset of B if A is a subset and A ≠ B.
8. **How can sets be implemented in Python?**
   Using curly braces `{}` or the `set()` constructor.
9. **What is the time complexity of set operations in Python?**
   Average-case O(1) for add, remove, and contains.
10. **What is the difference between a set and a list?**
    Sets are unordered and store unique elements, lists are ordered and can have duplicates.
11. **What is the iterator operation in a set?**
    It allows looping through all elements of the set.
12. **How is set ADT useful in real life?**
    For checking membership, removing duplicates, and mathematical set operations.

3. Binary Search Tree (BST) and Operations

1. **What is a binary search tree?**
   A BST is a binary tree where left child < parent < right child.
2. **What is the main property of BST?**
   All left subtree nodes are less than the node, right subtree nodes are greater.
3. **What are the main operations on BST?**
   Insert, search, delete, traverse.
4. **How to find the minimum value in a BST?**
   Go to the leftmost node.
5. **How to find the maximum value in a BST?**
   Go to the rightmost node.
6. **What is in-order traversal?**
   Visit left subtree, node, then right subtree; gives sorted order.
7. **What is pre-order traversal?**
   Visit node, left subtree, then right subtree.
8. **What is post-order traversal?**
   Visit left subtree, right subtree, then node.
9. **How to search for a value in BST?**
   Compare with root; go left if smaller, right if larger, repeat.
10. **What is the time complexity of search in BST?**
    O(h), where h is the height of the tree.
11. **What is the worst-case time for BST operations?**
    O(n), if the tree is skewed.
12. **How to swap left and right pointers at every node?**
    Recursively exchange left and right for each node.
13. **What is the longest path in a BST?**
    It is the height of the tree.
14. **How to delete a node with two children in BST?**
    Replace with in-order successor or predecessor, then delete that node.
15. **Difference between BST and binary tree?**
    BST has ordering property, binary tree does not.

4. Expression Tree

1. **What is an expression tree?**
   A binary tree representing arithmetic expressions.
2. **What are the leaves in an expression tree?**
   Operands (numbers or variables).
3. **What are the internal nodes in an expression tree?**
   Operators (+, -, *, /).

4. **What is prefix expression?**
   Operator comes before operands.
5. **What is infix expression?**
   Operator comes between operands.
6. **What is postfix expression?**
   Operator comes after operands.
7. **How to construct an expression tree from prefix?**
   Scan from right to left, create nodes for operands and operators.
8. **How to traverse an expression tree in post-order?**
   Visit left, right, then node.
9. **What is the use of expression trees?**
   To evaluate or convert expressions.
10. **How to delete an expression tree?**
    Use post-order traversal to delete all nodes.

5. Dictionary using BST

1. **What is a dictionary in data structures?**
   A collection of key-value pairs.
2. **How is a dictionary implemented using BST?**
   Keys are stored in BST nodes, values as data.
3. **How to add a new keyword?**
   Insert a new node with the keyword as key.
4. **How to delete a keyword?**
   Delete the node with that key from the BST.
5. **How to update a value?**
   Search for the key and change its value.
6. **How to display data in ascending order?**
   Perform in-order traversal.
7. **How to display data in descending order?**
   Perform reverse in-order traversal.
8. **How many maximum comparisons for finding a keyword?**
   Equal to the height of the BST.
9. **What is the time complexity for searching in BST?**
   O(h), where h is the height.
10. **What happens if BST becomes unbalanced?**
    Operations become slower, up to O(n).

6. Graph and Traversal

1. **What is a graph?**
   A collection of vertices (nodes) and edges (connections).
2. **Difference between directed and undirected graph?**
   Directed edges have direction, undirected do not.
3. **What is an adjacency matrix?**
   A 2D array indicating edge presence between vertices.
4. **What is an adjacency list?**
   A list where each vertex stores its adjacent vertices.
5. **What is BFS?**
   Breadth First Search, visits level by level using a queue.
6. **What is DFS?**
   Depth First Search, visits as deep as possible using stack or recursion.
7. **How to check if a graph is connected?**
   Perform BFS or DFS; if all nodes are visited, it's connected.
8. **What is the degree of a vertex?**
   Number of edges connected to the vertex.
9. **What is a null graph?**
   A graph with no edges.
10. **What is an acyclic graph?**
    A graph with no cycles.
11. **What is a weighted graph?**
    Edges have weights or costs.
12. **What is a spanning tree?**
    A subgraph that connects all vertices without cycles.
13. **What is the time complexity for BFS/DFS using adjacency list?**
    O(V + E), where V is vertices and E is edges.

7. Minimum Spanning Tree (MST)

1. **What is a minimum spanning tree?**
   A tree connecting all vertices with minimum total edge weight.
2. **Which algorithms find MST?**
   Prim's and Kruskal's algorithms.
3. **What is Prim's algorithm?**
   Starts from a vertex, adds the smallest edge to grow the tree.
4. **What is Kruskal's algorithm?**
   Sorts all edges and adds them one by one, avoiding cycles.
5. **What is the use of MST?**
   Network design, like connecting offices with minimum cost.

6. **What is a spanning tree?**
   A subgraph that connects all vertices without forming cycles.
7. **How many edges in an MST with n vertices?**
   n - 1 edges.
8. **What is the greedy property in MST algorithms?**
   Always choose the smallest available edge.
9. **What is the time complexity of Prim's algorithm?**
   O(E log V) with min-heap.
10. **What is a cycle in a graph?**
    A path that starts and ends at the same vertex.

8. Optimal Binary Search Tree (OBST)

1. **What is an OBST?**
   A BST with minimum expected search cost, given access probabilities.
2. **How is search cost calculated in OBST?**
   Sum of (probability × depth) for all keys.
3. **Which technique is used to build OBST?**
   Dynamic programming.
4. **What is the advantage of OBST?**
   Faster average search if access probabilities are known.
5. **What is the difference between BST and OBST?**
   OBST considers search probabilities; BST does not.
6. **What is the time complexity for OBST construction?**
   $O(n^3)$, where n is number of keys.
7. **What is the root matrix in OBST?**
   Stores the root of each subtree for optimal cost.
8. **What is the weight matrix in OBST?**
   Stores sum of probabilities for subtrees.
9. **What is a dummy key in OBST?**
   Represents unsuccessful search probabilities.
10. **Where is OBST used?**
    Databases and compilers for fast lookup.

9. AVL Tree (Height Balanced Tree)

1. **What is an AVL tree?**
   A self-balancing BST where balance factor is -1, 0, or +1.
2. **What is balance factor?**
   Difference in heights of left and right subtrees.

3. **Why are AVL trees used?**
   To keep search, insert, and delete operations O(log n).
4. **What is a rotation in AVL tree?**
   Operation to restore balance after insertion or deletion.
5. **Types of rotations in AVL trees?**
   Left, right, left-right, and right-left.
6. **What is the height of an AVL tree with n nodes?**
   O(log n).
7. **How does AVL tree differ from BST?**
   AVL tree is always balanced, BST may not be.
8. **What happens if AVL property is violated?**
   Tree is rebalanced using rotations.
9. **What is the time complexity of AVL insert/delete?**
   O(log n).
10. **What is the maximum balance factor allowed?**
    +1, 0, or -1.
11. **How is search performed in AVL tree?**
    Same as BST, but always balanced.
12. **What is the worst-case height of AVL tree?**
    $1.44 \times \log_2(n+2) - 1.328$.

10. Priority Queue

1. **What is a priority queue?**
   A queue where each element has a priority; highest priority is served first.
2. **How can a priority queue be implemented?**
   Using arrays, linked lists, or heaps.
3. **What is the time complexity of insert in heap-based priority queue?**
   O(log n).
4. **What is dequeue operation in priority queue?**
   Removes the element with the highest priority.
5. **What is the use of priority queue in real life?**
   Task scheduling, hospital patient management.
6. **What is the difference between queue and priority queue?**
   Queue is FIFO; priority queue serves by priority.
7. **What is the peek operation?**
   Returns the highest priority element without removing it.
8. **How is hospital patient queue managed?**
   By assigning priorities: serious, non-serious, general checkup.

9. **What is the time complexity of finding max in an unordered array?**
   O(n).
10. **What is the time complexity of finding max in a heap?**
    O(1).

11. File Organization (Sequential & Indexed Sequential)

1. **What is file organization?**
   The way records are arranged on storage media.
2. **What is sequential file organization?**
   Records are stored one after another, usually in order of insertion.
3. **What is indexed sequential file organization?**
   Records are stored sequentially with an index for fast access.
4. **What is a logical file?**
   A complete set of records for a specific application.
5. **What is a physical file?**
   A physical unit like a disk or tape.
6. **How is a record deleted in sequential files?**
   Copy all except the record to a new file, then replace the old file.
7. **What is the advantage of sequential files?**
   Simple and easy to maintain.
8. **What is the disadvantage of sequential files?**
   Slow for random access.
9. **What is the advantage of indexed sequential files?**
   Faster search using the index.
10. **What is a multi-index file?**
    A file with more than one index for different keys.
11. **What is a heap file?**
    Records stored in any order, no sorting.
12. **What is a B+ tree file organization?**
    Uses B+ tree for indexing and fast range queries.

For best results:

- Revise these questions before your viva.
- Practice answering aloud in 1–3 lines for each.
- Focus on clarity and key concepts.

If you need this manual as a downloadable PDF, let me know!

⁂