Author: Abhishek Chauhan

Software Engineer at MagpieXYZ
LinkedIn : https://www.linkedin.com/in/abhishek-chauhan-10b4991b8/
GitHub : https://github.com/AbhishekChauhan9036

Time and Space Complexity

1. What is time complexity, and why is it important?
2. What is the difference between worst-case, best-case, and average-case time complexity?
3. Explain the concept of Big-O notation with examples.
4. How is the time complexity of nested loops calculated?
5. What is the time complexity of searching in an array?
6. What is the time complexity of inserting an element in a linked list?
7. Define space complexity and its importance in algorithm analysis.
8. How does recursion affect space complexity?
9. Compare the time complexity of stack operations versus queue operations.
10. Explain the trade-offs between time complexity and space complexity in algorithm design.

Stack

1. What is a stack, and how does it function?
2. Explain the concept of LIFO (Last In, First Out) with examples.
3. What are the basic operations of a stack?
4. Differentiate between static and dynamic stack implementations.
5. How is a stack implemented using an array?
6. How is a stack implemented using a linked list?
7. What is the difference between push and pop operations in a stack?
8. Explain stack overflow and underflow conditions.
9. What are the applications of stacks in real-life scenarios?
10. How is a stack used in function call management (recursion)?
11. What is a circular stack?
12. Explain how a stack can be used to reverse a string.
13. How do you evaluate a postfix expression using a stack?
14. How is a stack used to check for balanced parentheses in an expression?
15. What is the difference between a stack and a queue?

Queue

1. What is a queue, and how does it function?
2. Explain the concept of FIFO (First In, First Out) with examples.
3. What are the basic operations of a queue?
4. How is a queue implemented using an array?
5. How is a queue implemented using a linked list?
6. What is the difference between an enqueue and a dequeue operation?
7. Explain queue overflow and underflow conditions.
8. What is a circular queue, and why is it used?
9. What is a priority queue, and how does it differ from a regular queue?
10. How does a deque (double-ended queue) work?
11. What are some real-life applications of queues?
12. How are queues used in operating systems for process scheduling?
13. Explain how a queue can be implemented using two stacks.
14. What is the difference between a circular queue and a deque?
15. How does a blocking queue work in multithreaded programming?

Linked List

1. What is a linked list, and how is it structured?
2. Differentiate between singly linked lists and doubly linked lists.
3. What is a circular linked list? Provide examples.
4. Explain the advantages of using linked lists over arrays.
5. What are the disadvantages of linked lists compared to arrays?
6. How is memory allocated for nodes in a linked list?
7. How do you insert a new node at the beginning of a singly linked list?
8. How do you delete a node from the end of a doubly linked list?
9. What is the time complexity of searching for an element in a linked list?
10. Explain how a linked list can be used to implement a stack or a queue.

Binary Tree

1. What is a binary tree, and how is it structured?
2. What are the key properties of a binary tree?
3. How does a full binary tree differ from a complete binary tree?
4. Explain the concept of height and depth in a binary tree.
5. List the types of tree traversals and their purposes in binary trees.

Binary Search Tree (BST)

1. What is a binary search tree (BST), and how is it different from a binary tree?
2. Explain the process of inserting a node into a BST.
3. How do you search for an element in a BST?
4. What are the advantages and disadvantages of using a BST?
5. How can you check if a given binary tree is a BST?

General Binary Tree to Binary Tree Conversion

1. What is the process to convert a general binary tree into a binary tree?
2. Explain the left-child, right-sibling representation of general binary trees.
3. How can you convert a general binary tree to a binary search tree (BST)?
4. What are the challenges in converting a general binary tree to a binary tree?
5. Describe a use case where converting a general binary tree to a binary tree is required.

AVL Tree

1. What is an AVL tree, and how does it maintain balance?
2. Explain the rotations used in an AVL tree (LL, RR, LR, RL).
3. Compare the advantages of an AVL tree over a BST.
4. What is the balance factor in an AVL tree?
5. Describe how to insert a node into an AVL tree.

B Tree

1. What is a B-tree, and where is it used?
2. Explain the properties of a B-tree.
3. How is data inserted into a B-tree?
4. Compare B-tree and AVL tree in terms of use cases.
5. What are the advantages of B-trees for disk-based storage?

Graph

1. What is a graph, and what are its main components?
2. Explain the difference between directed and undirected graphs.
3. What are the different types of graphs (e.g., cyclic, acyclic)?
4. What are the applications of graphs in computer science?
5. Compare graphs and trees.

Adjacency Matrix

1. What is an adjacency matrix representation of a graph?
2. Explain the space and time complexity of using an adjacency matrix.
3. How can weighted graphs be represented using an adjacency matrix?
4. What are the advantages of using an adjacency matrix?
5. Compare adjacency matrix and adjacency list for sparse graphs.

Adjacency List

1. What is an adjacency list representation of a graph?
2. Explain how adjacency lists are implemented in memory.
3. How does an adjacency list handle weighted graphs?
4. What are the advantages of using an adjacency list over an adjacency matrix?
5. Compare adjacency list and adjacency matrix for dense graphs.

BFS and DFS

1. What is Breadth-First Search (BFS), and how does it work?
2. Explain Depth-First Search (DFS) and its recursive implementation.
3. Compare BFS and DFS in terms of space and time complexity.
4. What are the practical applications of BFS and DFS?
5. How can BFS and DFS be used to detect cycles in a graph?

Spanning Tree

1. What is a spanning tree of a graph?
2. Explain the properties of a minimum spanning tree (MST).
3. How does Kruskal's algorithm find the MST?
4. Describe Prim's algorithm for finding the MST.
5. Compare Prim's and Kruskal's algorithms in terms of efficiency.

Searching

1. What is the difference between linear search and binary search?
2. Explain the time complexity of linear search.
3. What are the prerequisites for using binary search?
4. What is the time complexity of binary search in the worst case?
5. How can binary search be implemented recursively and iteratively?
6. What is interpolation search, and how is it different from binary search?
7. Explain how exponential search works and where it is used.
8. What is the time complexity of searching in a balanced binary search tree (BST)?
9. How does hashing work for searching, and what are its advantages?
10. What is the difference between searching in an unsorted array and a sorted array?

Sorting

1. What is sorting, and why is it important in computer science?
2. Explain the difference between stable and unstable sorting algorithms.
3. How does bubble sort work, and what is its time complexity?
4. What are the advantages of insertion sort over bubble sort?
5. Explain how selection sort finds the smallest element in each pass.
6. How does merge sort work, and what is its time complexity?
7. Describe the quicksort algorithm and its partitioning process.
8. What is the difference between merge sort and quicksort in terms of time complexity?
9. What is counting sort, and when is it used?
10. Compare the space complexities of different sorting algorithms (e.g., merge sort vs. quicksort).