Must-Know Data Structures and Algorithms

educative

Why DSA Matters?

- DSA forms the backbone of Computer Science.
- Knowing DSA is the secret sauce to success in coding interviews.
- DSA isn't just for software engineering interviews; it's a lifelong tool that supports career growth, including specialized roles.

What is Big O Notation and Why Does it Matter?

- Big O Notation is a way to measure how efficient an algorithm is.
- It measures both time complexity (how fast or slow something runs) and space complexity (how much memory it takes up).
- Understanding Big O helps to select optimal solutions, specifically when the input size is large.

Key Data Structures

- **Array:** Dynamic list, versatile in data storage.
- **Linked List:** A list with pointers, used for efficient insertions and deletions.
- **Hash Table:** A data structure known for fast data lookups and quick data retrieval.
- **Stack:** A collection that follows the Last in, first out (LIFO) operations, where the last item added is the first one removed.
- **Queue:** A collection that follows the First In, First Out (FIFO) operations, where the first item added is the first one removed.
- **Heap:** A structure that stores data based on priority, and allows priority-based data handling.
- **Binary Search Tree:** A tree that makes searching for data faster.

Important Algorithms

- **Sorting:** It's about arranging data in a specific order.
  - Insertion Sort
  - Selection Sort
  - Merge Sort
  - Quick Sort
- **Binary Search:** It's about fast data lookup in sorted arrays.
- **Tree Algorithms:**
  - Tree traversals: Pre-order, In-order, Post-order, and Level-order.
  - Breadth-First Search (BFS)
  - Depth-First Search (DFS)
- **Recursion:** It's about breaking problems into smaller sub-problems.
  - Factorial Calculation
  - Fibonacci Sequence
  - Tower of Hanoi

Practice Some Must-Know DSA Exercises

- Reversing a linked list
- Finding a middle value in a linked list
- Detect cycle in a linked list
- Merge two sorted linked lists in place
- Two-sum, where it's about finding two values in an array that equals the given target
- Finding the maximum or minimum value in an array
- Right rotate an array by one index
- Performing an inorder traversal in a binary search tree

Tips for Selecting DSA

- **Data Storage:** Look for how data needs to be organized for quick access and manipulation. For example, arrays store elements in contiguous memory.
- **Insertion/Deletion:** Look for how easily data can be added or removed. For example, linked lists allow
- efficient insertions/deletions at both ends.
- **Search Operations:** Consider the speed of searching for data. For example, hash tables offer constant-time lookups.
- **Traversal:** Look for how data is accessed or visited. For example, trees can be traversed in various orders, such as in-order, pre-order, and post-order.
- **Memory Usage:** Look for how much space a data structure consumes. For example, arrays use contiguous memory, while linked lists use nodes that are scattered.
- **Time Complexity:** Look for how fast operations like insertion, deletion, and search can be. For example, binary search in sorted arrays has `O(log n)` time complexity.

Study and Practice Tips for DSA

- Revise all the core data structures and algorithms.
- Understand the key features and functions of each data structure so that you know how to apply them in different scenarios.
- Practice coding problems that focus on data structures like lists, arrays, and trees to strengthen your confidence for DSA.

Interview Tips for DSA Questions

- **Clarify the Problem:** Before coding, ensure you understand the question fully.
- **Think Aloud:** Explain your thought process, especially when optimizing for time or space.
- **Edge Cases:** Identify and handle potential edge cases (e.g., empty inputs, large inputs, duplicates).