Data Structures and Algorithms (DSA) Topics and Questions

Core Concepts

Theory

- What is a data structure?
- What are the types of data structures?
- What is the difference between linear and non-linear data structures?
- What is a hierarchical data structure?
- What is the difference between contiguous and non-contiguous data structures?
- What is an algorithm?
- Why do we need algorithms?
- What is a dynamically typed language?
- What are primitive vs. non-primitive data types?
- What is the difference between mutable and immutable data in JavaScript?
- Is a string mutable or immutable in JavaScript?
- Why is a string immutable in JavaScript (or Python)?
- What is memory allocation?
- What are the types of memory allocation?
- What is static memory allocation?
- What are the advantages of static memory allocation?
- What is dynamic memory allocation?
- What are the drawbacks of dynamic memory allocation?
- What is the difference between static and dynamic memory allocation?
- What is stack memory?
- What is heap memory?
- What is the difference between stack and heap memory?
- What is virtual memory?
- What is the relationship between virtual memory and data structures?
- What are the security benefits of virtual memory?
- What is a memory pool?
- What is memory deallocation?
- What is a memory leak?
- How to prevent memory leakage in JavaScript?

- What is a garbage collector and how does it work?
- What is stack overflow?
- What is heap overflow?
- What is memory padding?
- What is a circular reference?
- What is a byte?
- What is the difference between kilobyte and kibibyte?
- What is a gigabyte?
- What is a nibble vs. a byte?
- What is character encoding?
- What is UTF-8?
- What is ASCII?
- What are escape sequences?
- What are control characters?
- What is string encoding?
- How do computers interpret strings?
- How much memory is used by strings?
- What is string sanitization?
- What is string validation?

Complexity Analysis

Theory

- What is complexity analysis?
- What is time complexity?
- What is space complexity?
- What is asymptotic analysis?
- What is Big O notation?
- What is Big Theta notation?
- What is Big Omega notation?
- How does Big O notation differ from Big Theta and Big Omega?
- What does O(1) complexity mean?
- What is O(log n) vs. O(n log n)?
- What is the best-case, average-case, and worst-case time complexity?
- When would you choose space complexity over time complexity?
- What are the time complexities for common array operations (e.g., accessing, inserting, deleting)?
- What is the time complexity of adding an element to a linked list?
- What is the time complexity of accessing an element in a linked list when the index is known?
- What is the time complexity of searching an element in a linked list?
- What is the time complexity of deleting an element from the middle of a linked list?
- What is the time complexity of binary search?
- What is the worst-case time complexity of binary search?

- What is the best-case time complexity of linear search?
- What is the time complexity for four nested loops?
- What are logarithmic functions and values?
- What are asymptotic notations for quadratic equations?
- What is the space complexity of recursion?

Arrays

Theory

- What is an array in data structures?
- What is the difference between traditional arrays and JavaScript arrays?
- What is a homogeneous vs. heterogeneous array?
- What is a jagged array?
- What is a sparse array?
- What is a multidimensional array?
- What is a subarray?
- What are typed arrays in JavaScript?
- What are efficient operations in arrays?
- What are the applications of arrays in real-world problems?
- What is array amortization?
- What is the difference between linked lists and arrays?
- What are the time complexities for common array operations (e.g., accessing, inserting, deleting)?
- What is memory allocation in arrays?

Practical

- Find the minimum and maximum number from an array.
- Find the second largest element in an array.
- Find the second smallest element in an array.
- Find the third largest element in an array without sorting.
- Find the kth largest element in an array.
- Find the frequency of occurrence of each number in an array.
- Create a function to find the average of even numbers in an array.
- Find the common element between arrays.
- Find the last occurrence of an element in a sorted array with duplicate values.
- Find the minimum from a rotated sorted array (using binary search).
- Find a subarray with the maximum number of elements in continuously increasing order.
- Find a combination of two numbers in an array that sum to a target value (e.g., 4 or 5).
- Find two numbers that sum to a target value from a sorted array in a single iteration.
- Remove the smallest number from an array.
- Remove a specific element from an array.
- Reverse an array.

- Flatten a multidimensional array.
- Remove a subarray containing the largest number from a 2D array.
- Find the sum of a column in a 2D array and add it as the last column.
- Check if a target is in a 2D array and return its index (e.g., `target = 8`, `output = [1,2]`).
- Remove duplicates from an array.
- Implement array operations without using built-in methods.
- Append and prepend elements to an array.

Strings

Theory

- What are the concepts of strings?
- What is the difference between a character and a string?
- What is a character array?
- Why are strings immutable in JavaScript (or Python)?
- What are string permutations?
- How to extract a substring from a string?
- What are control characters?

Practical

- Find the first non-repeating character in a string.
- Find the last non-repeating character in a string.
- Check if two strings are anagrams of each other.
- Reverse a string without using built-in methods.
- Reverse each word in a string without using built-in methods (e.g., "HELLO WORLD" → "OLLEH DLROW").
- Remove extra whitespaces between words in a string.
- Convert a string to Title Case.
- Ensure a string begins with uppercase and ends with a period.
- Find the longest substring palindrome in a given string.
- Check if parentheses are balanced in a string (e.g., `areBracesBalanced("{{}}{}{}")` → `true`).
- Find the longest substring without vowels from a string.
- Find the longest consecutive repeating characters in a string.
- Find the shortest word in a string.
- Extract digits from a string.
- Remove all occurrences of a specific character from a string (e.g., hide "l" from "hello").
- Remove a character from a string using recursion.
- Count words in a sentence without using built-in functions.
- Find the palindromic prefix in a string.
- Implement string permutations.
- Find the longest substring without repeating characters.
- Convert PascalCase to snake_case.

- Reverse the letters of all words in a string without using built-in functions.
- Implement a product-except-itself solution for an array.
- Solve the valid parentheses problem.

Linked Lists

Theory

- What is a linked list?
- Why is a linked list considered a linear data structure?
- What are the advantages of linked lists?
- What are the disadvantages of linked lists?
- What are the applications of linked lists?
- What is the difference between a singly linked list and a doubly linked list?
- What is a circular linked list?
- What is a circular doubly linked list?
- What is a multilinked list?
- What is the time complexity of linked list operations (e.g., insertion, deletion)?
- How does a doubly linked list differ in traversal compared to a singly linked list?
- What is memory allocation in linked lists?
- What is polynomial representation using a linked list?
- What is the head in a linked list?
- What is a tail in a linked list?

Practical

- Implement a singly linked list (all operations: set, insert, delete, init, get).
- Implement a doubly linked list (all operations).
- Implement a circular linked list.
- Convert a singly linked list to a doubly linked list.
- Convert a linked list to a circular linked list and validate.
- Convert an array `[1, 2, 3, 4, 5]` to a linked list.
- Insert a node at a specific position in a singly linked list.
- Insert a node after a node with particular data in a doubly linked list.
- Delete an element from a specific position in a singly linked list.
- Delete a specific node from a doubly linked list.
- Delete the front and back nodes where `data == data` in a doubly linked list.
- Delete the kth element from the end of a linked list.
- Delete the nth node from the last of a singly linked list (using two pointers).
- Delete the middle element from a linked list (in O(n) time).
- Delete by value in a doubly linked list.
- Remove all nodes that hold a specific value in a singly linked list.
- Remove the last instance of a value from a singly linked list.

- Remove odd element nodes from a linked list.
- Remove duplicates from a linked list without using other data structures.
- Reverse a singly linked list.
- Reverse a doubly linked list.
- Reverse print a doubly linked list.
- Sort nodes in a linked list.
- Merge two sorted linked lists.
- Detect a cycle/loop in a singly linked list (Floyd's cycle detection algorithm).
- Find the middle of a linked list in a single iteration (fast and slow pointer).
- Traverse a doubly linked list.
- Implement a singly linked list with proper encapsulation (using classes).
- Implement a singly linked list with a condition that only one duplicate is allowed.
- Swap the first and last nodes of a singly linked list.
- Print all elements of a linked list in order and reverse order.

Recursion

Theory

- What is recursion?
- What is a base case in recursion?
- What is the difference between direct and indirect recursion?
- What is tail recursion vs. head recursion?
- What is binary recursion?
- What is mutual recursion?
- What are the advantages of recursion?
- What are the disadvantages of recursion?
- What are the applications of recursion?
- What is the space complexity of recursion?
- What is the time complexity of recursion?
- What are recursive base conditions?
- What is the difference between recursion and loops?
- What are the limitations of recursion?
- What is backtracking?

Practical

- Implement a recursive function to find the sum of elements in an array.
- Implement a recursive function to find the largest element in an array.
- Print the first 10 elements of the Fibonacci series using recursion.
- Implement factorial using recursion.
- Implement a recursive function to reverse a string.
- Remove a character from a string using recursion.

- Remove even numbers from an array using recursion.
- Implement binary search using recursion.
- Find the sum of even numbers using recursion.
- Find the sum of digits using recursion.
- Recursively remove a character from a string (e.g., hide "l" from "hello").
- Implement a recursion that recurses only 5 times.
- Implement a palindrome check using recursion.
- Print the Fibonacci series under a limit using recursion.
- Remove duplicates from a linked list using recursion.

Binary Search

Theory

- What is binary search?
- What is the difference between binary search and linear search?
- What is the complexity of binary search?
- What is the limitation of binary search?
- What is the best-case time complexity of binary search?
- What is the worst-case time complexity of binary search?

Practical

- Implement binary search.
- Implement binary search using recursion.
- Implement binary search and replace a number with 0.
- Find the minimum from a rotated sorted array using binary search.
- Implement binary search on an array of strings.

Sliding Window and Two-Pointer Techniques

Theory

- What is the sliding window algorithm?
- What is the two-pointer technique?
- What is the fast and slow pointer approach?

Practical

- Implement the "Buy and Sell Stock" problem using the sliding window approach.

- Find the middle of a linked list in a single iteration using fast and slow pointers.
- Detect a cycle in a linked list using fast and slow pointers (Floyd's algorithm).
- Remove the nth node from the last of a linked list using the two-pointer technique.
- Find a subarray with the maximum number of elements in continuously increasing order.

Additional Topics

Theory

- What is a polynomial representation using a linked list?
- What is a generator function?

Practical

- Practice problems from Blind 75 LeetCode and learn optimal solutions.
- Refer to neetcode.io/practice for questions and practice them on LeetCode.
- Learn brute force and optimal solutions for problems (search YouTube for solutions).
- Solve problems like:
    - Two Sum (LeetCode).
    - Merge Two Sorted Lists.
    - Valid Parentheses.
    - Longest Substring Without Repeating Characters.
    - Product Except Self.
- Practice string problems without using built-in methods.
- Practice linked list, subarray, and array problems.
- Create a generator function to yield numbers from a 2D list.
- Flatten a nested list.
- Avoid overwriting built-in names in code.

Guidelines

- Get familiar with practicing problems on LeetCode, HackerRank, and GeeksforGeeks.
- Practice medium-level questions with time constraints.
- Practice explaining solutions.
- Avoid using AI tools; use blogs and videos to build DSA knowledge.
- Refer to FreeCodeCamp YouTube channel for recursion videos.
- Improve coding speed and logic-building skills.
- Solve application-level programs to strengthen understanding.