Program for Interview Preparation

Week-2 - Linked Lists, Stacks, Queues

Note: Make sure to practice the implementation of all three data structures since they can be asked in the interview

Linked Lists

- A linked list is a linear data structure where each element is a separate object.
- Each element of such a list needs to comprise of two items - the data and a reference to the next node.
- The object that holds the data and refers to the next element in the list is called a node.
- Data components may consist of several fields.
- For example, the data could be a student's name and Roll number in a linked list of students.
- The head points to the first node of the Linked List
- The last node of the Linked List always points to NULL.

Video Link for an easy explanation: https://www.youtube.com/watch?v=R9PTBwOzceo

Important Difference between a Linked List and an Array:

1. Unlike arrays, Linked Lists have dynamic memory allocation.
2. Arrays are always stored in the memory in a continuous manner, whereas that is not necessary for a Linked List.
3. The cost of accessing an element is higher for a Linked List as we only store the access to the root node, and for every other node, we must traverse the list, whereas we can directly access elements in an Array.
4. Array elements are independent of each other, whereas, in a Linked List, elements depend on each other since each node contains the address to the next element.

Link for detailed answer: https://www.javatpoint.com/ds-array-vs-linked-list

Important basic operation on a linked list:
Finding the middle node of a Linked List:
https://www.geeksforgeeks.org/write-a-c-function-to-print-the-middle-of-the-linked-list/
Deletion of a Node in a Linked List:
https://www.geeksforgeeks.org/linked-list-set-3-deleting-node/
Insertion of a Node in a Linked List:
https://www.geeksforgeeks.org/linked-list-set-2-inserting-a-node/

Doubly Linked Lists

- Permits traversal of the list in both directions
- Useful where navigation in both directions needed
- Each node points to its prev node and its next node. Header node only points to the next node.

Video Link for an easy Explanation: https://www.youtube.com/watch?v=e9NG_a6Z0mg

Practice Question: (Random Order)

Linked List Palindrome
Reverse a Linked List
Reverse Adjacent Nodes in a Linked List
Remove Duplicates in a Sorted Linked List
Detect a loop in a Linked List
LRU Cache (VERY IMPORTANT)
Copy a list with a random pointer
Add two numbers given in the form of Linked Lists
MergeSort / QuickSort in a Linked List
The starting point of a loop in a Linked List

Stacks

Stack is a collection of elements that follows the LIFO (Last In First Out) order. A good real life example of a stack is a stack of books. A new book is added to the top of the stack and while removing, the top most book is removed first.
Link for easy explanation : https://www.youtube.com/watch?v=F1F2imiOJfk&t=1s

There are mainly three operations in the stack:

- Push: Inserts an element to the top of the stack
- Pop: Removes the topmost element of the stack
- Seek: Returns the top element of the stack.

A stack can be implemented using an array or a linked list. The standard library also provides a built-in stack.
Link to implementation of stack:
https://www.geeksforgeeks.org/stack-data-structure-introduction-program/

The following are common applications of stack:

- Evaluating prefix and postfix notations
- Saving local variables when one function calls another, and this one calls another, and so on. Therefore stack also handles recursion.
- Implementing the compiler's syntax check for matching braces
- Storing all text changes for "undo" mechanism in text editors
- Backtracking while solving a maze
- Reversing the order of a list of elements or to reverse a word

Practice Questions (In order of difficulty):

1. Min Stack
2. Next Greater Element
3. Implement Queue using Stacks
4. Valid Parentheses
5. Decode String
6. Asteroid Collision
7. Minimum Cost Tree From Leaf Values
8. Daily Temperatures
9. Trapping Rain Water
10. Maximal Rectangle

Queues

Queue is a collection of elements that follows the FIFO (First In First Out) order. A good real life example of a queue is a line of people waiting for a bus at the bus stand. The person who comes first will be the first one to enter the bus.
Link to easy explanation: https://www.youtube.com/watch?v=XuCbpw6Bj1U

There are mainly two operations in the Queue:

- Enqueue: Inserts an element to the back of the queue
- Dequeue: Removes the element at the front of the queue

A queue can be implemented using an array or a linked list. The standard library also provides a built-in queue.
Link to implementation of Queue:
https://www.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/

Double-ended queues (Deque) are a special case of queues in which insertion and deletion operations are possible at both the ends. The standard library also provides a built-in deque.
Link to implementation of doubly queue:
https://www.geeksforgeeks.org/implementation-deque-using-doubly-linked-list/

Common Applications of Queue:

- Breadth-First Search is implemented with a Queue
- While scheduling jobs on the CPU the OS uses a queue

Practice Questions (In order of difficulty):

1. Moving Average from Data Stream
2. Implement Stack using Queues
3. Task Scheduler
4. Design Snake Game
5. Design Circular Queue
6. Max Sum of Rectangle No Larger Than K