bytebybyte

Linked List Quickstart Guide

In this mini-workbook, we'll get you up and running with linked lists. These are used as a building block to the more complicated data structures such as trees, graphs, etc., so it is important to deeply understand their inner workings.

In this guide, we will quickly highlight the key concepts that you need to know for your interview. If you're unfamiliar with any of these, I highly recommend taking some time to review them.

Then, I've provided you with a series of daily exercises to help you get up and running quickly. I recommend using these similarly to how you use your daily workbook. Do one set of exercises each day and be sure to log them in your tracker. This will help you ensure that you've reviewed all the most important points.

Key Linked List Terminology

- **Singly Linked List**
  The core data structure.
- **Doubly Linked List**
  A linked list data structure where each node has a pointer to the next node and the previous node.
- **Circular Linked List**
  A doubly linked list where the tail points to the head and the head points to the tail.

Key Linked List Patterns

- **Basic Linked List Operations**
  - Construct a linked list
  - Insert a node
  - Remove a node
  - Find a value

Daily Exercises

Day 1

Implement a `Node` and `LinkedList` class. You can assume that node values are just integers. Implement the following methods for the tree class:

- Constructor (LucidProgramming Video)
- `insert(int n)`
  Inserts the value n into the linked list (LucidProgramming Video)
- `delete(int n)`
  Removes the value n from the linked list (LucidProgramming Video)
- `contains(int n)`
  Return true if the linked list contains the given value
- `length()`
  Return the length of the linked list (LucidProgramming Video)

Day 2

Implement the following fundamental operations on a singly-linked list. I would highly encourage you to attempt solving each both recursively and iteratively. Each has a natural approach that lends itself, but practicing both approaches will help you develop a much deeper understanding.

- Node Swap (LucidProgramming Video)

- Reverse Nodes (LucidProgramming Video)
- Merge Two Sorted Lists (LucidProgramming Video)
- Remove Duplicates (LucidProgramming Video)
- Nth to Last Node (LucidProgramming Video)
- Count Occurrences (LucidProgramming Video)
- Rotate Node (LucidProgramming Video)
- Is Palindrome (LucidProgramming Video)
- Move Tail to Head (LucidProgramming Video)
- Sum Two Lists (LucidProgramming Video)

Day 3

Implement a `Node` and `DoublyLinkedList` class. You can assume that node values are just integers. Implement the following methods for the tree class:

- Constructor
- `insert(int n) and prepend(int n)`
  Inserts the value n into the doubly-linked list (LucidProgramming Video)
- `delete(int n)`
  Removes the value n from the doubly-linked list (LucidProgramming Video)
- `contains(int n)`
  Return true if the doubly-linked list contains the given value
- `length()`
  Return the length of the doubly-linked list

Day 4

Implement the following fundamental operations on a doubly-linked list. I would highly encourage you to attempt solving each both recursively and iteratively. Each has a natural approach that lends itself, but practicing both approaches will help you develop a much deeper understanding.

- Reverse Nodes (LucidProgramming Video)
- Remove Duplicate Nodes (LucidProgramming Video)
- Pairs with Sum (LucidProgramming Video)

Day 5

Implement a `Node` and `CircularLinkedList` class. You can assume that node values are just integers. Implement the following methods for the tree class:

- Constructor
- `insert(int n) and prepend(int n)`
  Inserts the value n into the circular-linked list (LucidProgramming Video)
- `delete(int n)`
  Removes the value n from the circular linked list (LucidProgramming Video)
- `contains(int n)`
  Return true if the circular-linked list contains the given value
- `length()`
  Return the length of the circular-linked list

Day 6

Implement the following fundamental operations on a circular-linked list. I would highly encourage you to attempt solving each both recursively and iteratively. Each has a natural approach that lends itself, but practicing both approaches will help you develop a much deeper understanding.

- Split Circular Linked List (LucidProgramming Video)
- Josephus Problem (LucidProgramming Video)
- Is Circular Linked List (LucidProgramming Video)

Exercises Solutions

Coming soon!