Linked List Practice Questions

Question 1: Reverse a Linked List

Write a function to reverse a singly linked list. You must reverse it **in-place** without using extra space for another list.

Example:
Input: 1 → 2 → 3 → 4 → NULL
Output: 4 → 3 → 2 → 1 → NULL

Time Complexity: O(n), **Space Complexity:** O(1)

Question 2: Detect Cycle in Linked List

Given a linked list, determine if it contains a cycle. A cycle occurs when a node's next pointer points back to a previous node in the list.

Example:
Input: 1 → 2 → 3 → 4 → 2 (points back to node 2)
Output: true (cycle exists)

Hint: Use the Floyd's cycle detection algorithm (slow and fast pointers).

Time Complexity: O(n), **Space Complexity:** O(1)

Question 3: Merge Two Sorted Linked Lists

Given two sorted singly linked lists, merge them into a single sorted linked list and return the head of the merged list.

Example:
Input: List1: 1 → 3 → 5 → NULL, List2: 2 → 4 → 6 → NULL
Output: 1 → 2 → 3 → 4 → 5 → 6 → NULL

Hint: Use two pointers to traverse both lists and compare values.

Time Complexity: O(n + m), **Space Complexity:** O(1)

Question 4: Find Middle of Linked List

Write a function to find the middle node of a singly linked list. If the list has an even number of nodes, return the second middle node.

Example:
Input: 1 → 2 → 3 → 4 → 5 → NULL
Output: 3 (the middle node)

Hint: Use slow and fast pointers (fast moves 2 steps, slow moves 1 step).

Time Complexity: O(n), **Space Complexity:** O(1)

Question 5: Remove Duplicates from Sorted Linked List

Given a sorted singly linked list, remove duplicate nodes such that each value appears only once.

Example:
Input: 1 → 1 → 2 → 2 → 3 → NULL
Output: 1 → 2 → 3 → NULL

Time Complexity: O(n), **Space Complexity:** O(1)

Question 6: Palindrome Linked List

Determine if a singly linked list is a palindrome (reads the same forward and backward).

Example:
Input: 1 → 2 → 3 → 2 → 1 → NULL
Output: true

Input: 1 → 2 → 3 → NULL
Output: false

Time Complexity: O(n), **Space Complexity:** O(1) if using slow/fast pointers and reversal

---

Question 7: Remove Nth Node From End of List

Given a singly linked list and an integer n, remove the nth node from the end of the list and return the head.

Example:
Input: 1 → 2 → 3 → 4 → 5 → NULL, n = 2
Output: 1 → 2 → 3 → 5 → NULL (removed node with value 4)

Hint: Use a dummy node to handle edge cases (like removing the head).

Time Complexity: O(n), **Space Complexity:** O(1)

Question 8: Intersection of Two Linked Lists

Given two singly linked lists, find the node at which they intersect. If they do not intersect, return NULL.

Example:
List1: 4 → 1 → 8 → 4 → 5 → NULL
List2: 5 → 0 → 1 → 8 → 4 → 5 → NULL
(They intersect at node with value 8)

Hint: Calculate lengths and use two pointers.

Time Complexity: O(n + m), **Space Complexity:** O(1)

Question 9: Partition List Around a Value

Given a singly linked list and a value x, partition the list such that all nodes with values less than x come before all nodes with values greater than or equal to x. Preserve the original relative order of nodes.

Example:
Input: 1 → 4 → 3 → 2 → 5 → 2 → NULL, x = 3
Output: 1 → 2 → 2 → 4 → 3 → 5 → NULL

Time Complexity: O(n), **Space Complexity:** O(1)

Question 10: Reverse Nodes in K-Group

Given a linked list, reverse the nodes in groups of k (where k is a given positive integer). If the number of nodes is not a multiple of k, then left-out nodes should remain as-is.

Example:
Input: 1 → 2 → 3 → 4 → 5 → NULL, k = 2
Output: 2 → 1 → 4 → 3 → 5 → NULL

Input: 1 → 2 → 3 → 4 → 5 → NULL, k = 3
Output: 3 → 2 → 1 → 4 → 5 → NULL

Time Complexity: O(n), **Space Complexity:** O(1) excluding recursion stack

---

Study Tips

1. **Master pointer manipulation:** Understand how to update pointers correctly to avoid losing references to nodes.[web:51]
2. **Use dummy nodes:** A dummy node pointing to the head simplifies edge case handling when modifying the list structure.[web:49]
3. **Slow and fast pointers:** This technique is essential for cycle detection, finding the middle, and other problems.[web:49][web:50]
4. **Practice recursion:** Many linked list problems can be solved recursively, which helps develop deeper understanding.[web:53]
5. **Draw diagrams:** Visualizing pointer movements prevents mistakes during implementation.[web:51]

References

[1] GeeksforGeeks. Top 50 Linked List Interview Questions. https://www.geeksforgeeks.org/dsa/top-50-linked-list-interview-question/

[2] Tech Interview Handbook. Linked List Cheatsheet for Coding Interviews. https://www.techinterviewhandbook.org/algorithms/linked-list/

[3] Final Round AI. 25 Linked List Interview Questions You Need to Know. https://www.finalroundai.com/blog/linked-list-interview-questions

[4] Kunal Kushwaha. Linked List Interview Questions - Google, Facebook, Amazon, Microsoft. https://www.youtube.com/watch?v=70tx7KcMROc

[5] Shiksha Online. Practice Problems of Linked List. https://www.shiksha.com/online-courses/articles/practice-problems-of-linked-list/