

Linked List Interview Questions

Basic Questions

1. What is a linked list and how does it work?
2. Compare linked list and array. When would you use one over the other?
3. What are the different types of linked lists?
4. What is the difference between singly and doubly linked lists?
5. What is a circular linked list and where is it used?
6. What is the time complexity of insertion, deletion, and search operations in a linked list?
7. How do you traverse a linked list? What are the challenges?

Intermediate-Level Questions

8. How is memory managed in a linked list?
9. How does garbage collection work in the context of linked lists?
10. What are the disadvantages of using linked lists?
11. What is the concept of a "dummy node" in a linked list?
12. What is the difference between a tail pointer and a head pointer?
13. Can we implement a stack/queue using a linked list? How?
14. What are sentinel nodes and what is their purpose?

Advanced Level Questions

15. How does the "tortoise and hare" algorithm detect a cycle in a linked list?
16. What happens if a linked list contains a cycle? How do you detect and remove it?
17. Explain the slow and fast pointer technique and give scenarios where it's useful.
18. What are the problems in deleting a node without head pointer in a singly linked list?
19. How do you handle overlapping/intersecting linked lists?
20. What's the difference between deep copy and shallow copy in linked lists (esp. with random pointers)?
21. Can a linked list be used to implement memory management systems? How?
22. Explain the real-life use cases of linked lists in operating systems or compilers.

Conceptual and Analytical

23. Why is linked list preferred for dynamic memory allocation over arrays?
24. How do linked lists help in efficient memory usage?
25. Why is insertion faster in a linked list than an array, but access is slower?
26. How does pointer manipulation affect performance in linked lists?

27. What is the overhead involved in a doubly linked list?

28. Why can't we do binary search in a linked list?

Application-Based Design Questions

29. How can you implement an LRU (Least Recently Used) Cache using a linked list?

30. What data structure would you use to design a browser history? How would a linked list help?

31. How would you design a music playlist where songs can be played forward and backward?

32. Which type of linked list is suitable for implementing Undo-Redo functionality and why?

33. In what scenarios would you prefer a circular linked list?

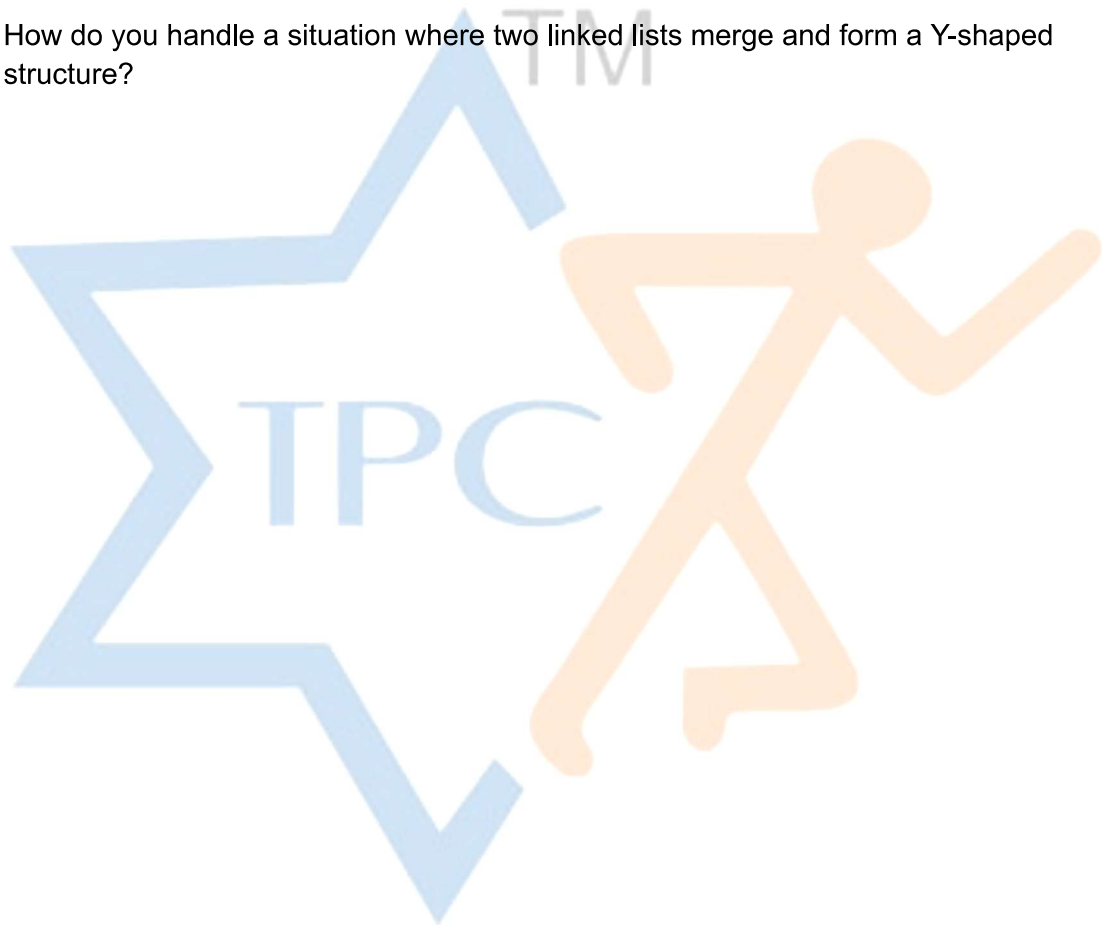
34. What data structure would you use for a memory-efficient queue? Why not array?

Problem-Solving & Edge Case Discussions

35. What issues arise when deleting a node in a singly linked list without the previous pointer?

36. Why is memory leak a concern in linked lists?

37. How can you identify the intersection point of two singly linked lists?
38. What precautions should be taken while manipulating the next pointers during deletion?
39. Explain how you would reverse a linked list without using extra space.
40. How do you handle a situation where two linked lists merge and form a Y-shaped structure?



www.tpcglobal.in