DSA Strong Dataset – University Level

Q1. Explain time and space complexity analysis with examples.

Q2. Design an algorithm to reverse a linked list using recursion.

Q3. Explain the difference between binary tree, BST, and AVL tree.

Q4. Write pseudocode for merge sort and explain its working.

Q5. Design a stack using two queues and analyze its complexity.

Q6. Explain hash table collision and how chaining resolves it.

Q7. Implement Dijkstra ’ s algorithm and explain with an example graph.

Q8. What is the difference between BFS and DFS traversal?

Q9. Explain dynamic programming using the Fibonacci example.

Q10. Explain the divide and conquer strategy with Merge Sort.

Q11. Write an algorithm to find the intersection of two linked lists.

Q12. Design a queue using two stacks and discuss complexity.

Q13. Explain recursion depth and stack overflow with examples.

Q14. Implement binary search and explain its best and worst cases.

Q15. Describe heap operations and use cases of priority queues.

Q16. Explain topological sorting and its real-life applications.

Q17. What is a spanning tree? Explain Kruskal ’ s algorithm.

Q18. Explain adjacency matrix vs adjacency list representation.

Q19. What is memoization and how does it improve performance?

Q20. Describe Floyd-Warshall algorithm for all pairs shortest path.

Q21. Explain the working of quick sort and its average complexity.

Q22. Explain graph coloring and its applications.

Q23. Discuss the importance of Big-O notation in algorithm design.

Q24. What is a circular linked list and its advantages?

Q25. Explain the concept of recursion tree and time complexity.

Q26. Discuss Bellman-Ford algorithm and how it differs from Dijkstra.

Q27. Explain why dynamic programming avoids repeated subproblems.

Q28. Explain how binary search tree insertion and deletion work.

Q29. Design an algorithm to find all leaf nodes in a binary tree.

Q30. Discuss how to detect cycles in a directed graph.

Q31. Explain different tree traversal techniques with examples.

Q32. Implement quick sort and discuss space complexity.

Q33. Explain greedy algorithms with examples.

Q34. What are the benefits and limitations of recursion?

Q35. Explain Huffman coding algorithm for data compression.

Q36. What are red-black trees and why are they used?

Q37. Explain graph traversal order using BFS and DFS on a sample graph.

Q38. Describe Prim ’ s algorithm with example.

Q39. Explain binary heap structure and operations.

Q40. Write a function to check if a string is a palindrome using stack.

Q41. What is hashing? Explain open addressing with example.

Q42. Design an LRU cache using a doubly linked list and hash map.

Q43. Explain how quicksort works with partitioning logic.

Q44. Discuss how recursion is implemented internally using call stack.

Q45. Explain the use of adjacency list for sparse graphs.

Q46. What is the difference between a min-heap and a max-heap?

Q47. Explain divide and conquer with an example other than sorting.

Q48. Design an algorithm to detect loop in a linked list.

Q49. Write an algorithm to check if a graph is bipartite.

Q50. Explain stack vs heap memory management.

Q51. Discuss the working of AVL tree rotations.

Q52. Rephrase: Describe divide and conquer with an example other than sorting.

Q53. Rephrase: Analyze Bellman-Ford algorithm and how it differs from Dijkstra.

Q54. Rephrase: Describe recursion depth and stack overflow with examples.

Q55. Rephrase: What are the benefits and limitations of recursion?

Q56. Rephrase: Describe the divide and conquer strategy with Merge Sort.

Q57. Rephrase: Implement Dijkstra ’ s algorithm and explain with an example graph.

Q58. Rephrase: Write an algorithm to find the intersection of two linked lists.

Q59. Rephrase: Describe why dynamic programming avoids repeated subproblems.

Q60. Rephrase: What is the difference between BFS and DFS traversal?

Q61. Rephrase: Describe Huffman coding algorithm for data compression.

Q62. Rephrase: Design a stack using two queues and analyze its complexity.

Q63. Rephrase: What is hashing? Describe open addressing with example.

Q64. Rephrase: Analyze the importance of Big-O notation in algorithm design.

Q65. Rephrase: Analyze the working of AVL tree rotations.

Q66. Rephrase: What is a spanning tree? Describe Kruskal ’ s algorithm.

Q67. Rephrase: Write a function to check if a string is a palindrome using stack.

Q68. Rephrase: Describe stack vs heap memory management.

Q69. Rephrase: Describe Prim ’ s algorithm with example.

Q70. Rephrase: Analyze how recursion is implemented internally using call stack.

Q71. Rephrase: Describe topological sorting and its real-life applications.