DSA with JavaScript MCQ Test (50+ Questions)

1. What is a Data Structure?

a) A method to execute algorithms

b) A way to organize and store data efficiently

c) A type of programming language

d) A hardware component

Answer: b) A way to organize and store data efficiently

---

2. What does time complexity analysis measure?

a) The amount of memory used

b) The time taken by an algorithm as a function of input size

c) The number of variables in a program

d) The speed of the processor

Answer: b) The time taken by an algorithm as a function of input size

---

3. What is the worst-case time complexity of an algorithm?

a) The minimum time it takes

b) The maximum time it takes

c) The average time it takes

d) The best possible time

Answer: b) The maximum time it takes

---

4. What is an algorithm?

a) A data storage technique

b) A step-by-step procedure to solve a problem

c) A type of data structure

d) A programming language

Answer: b) A step-by-step procedure to solve a problem

---

5. What is the time complexity of array indexing in JavaScript?

a) O(1)

b) O(n)

c) O(log n)

d) O(n^2)

Answer: a) O(1)

---

6. How do you traverse an array in JavaScript?

a) Using a for loop

b) Using a stack

c) Using recursion only

d) Using a queue

Answer: a) Using a for loop

---

7. What is the time complexity of inserting an element at the beginning of an array in JavaScript?

a) O(1)

b) O(n)

c) O(log n)

d) O(n^2)

Answer: b) O(n)

---

8. What is the time complexity of Linear Search in the average case?

a) O(1)

b) O(n)

c) O(log n)

d) O(n^2)

Answer: b) O(n)

---

9. How do you modify an array element in JavaScript?

a) arr[index] = value

b) arr.push(value)

c) arr.add(index, value)

d) arr.modify(index, value)

Answer: a) arr[index] = value

---

10. Which JavaScript method concatenates two arrays?

a) arr1 + arr2

b) arr1.concat(arr2)

c) arr1.join(arr2)

d) arr1.merge(arr2)

Answer: b) arr1.concat(arr2)

---

11. What is the time complexity of Selection Sort in the best case?

a) O(n)

b) O(n log n)

c) $O(n^2)$

d) O(log n)

Answer: c) $O(n^2)$

---

12. What is the main idea behind Bubble Sort?

a) Divide and conquer

b) Repeatedly swap adjacent elements if they are in the wrong order

c) Select the minimum element and place it

d) Use a pivot to partition

Answer: b) Repeatedly swap adjacent elements if they are in the wrong order

---

13. What is the average-case time complexity of Merge Sort?

a) O(n)

b) O(n log n)

c) $O(n^2)$

d) O(log n)

Answer: b) O(n log n)

---

14. What is the worst-case time complexity of Quick Sort?

a) O(n)

b) O(n log n)

c) $O(n^2)$

d) O(log n)

Answer: c) $O(n^2)$

---

15. How is a 2D array represented in JavaScript?

a) An array of arrays

b) A single flat array

c) A linked list

d) A stack

Answer: a) An array of arrays

---

16. What is the time complexity of accessing an element in a 2D array?

a) O(1)

b) O(n)

c) O(log n)

d) $O(n^2)$

Answer: a) O(1)

---

17. How do you modify an element in a 2D array in JavaScript?

a) arr[i, j] = value

b) arr[i][j] = value

c) arr(i)(j) = value

d) arr.modify(i, j, value)

Answer: b) arr[i][j] = value

---

18. What is a singly linked list?

a) A list where each node points to the previous node

b) A list where each node points to the next node

c) A list with two pointers per node

d) A circular list

Answer: b) A list where each node points to the next node

---

19. What is the time complexity of traversing a singly linked list?

a) O(1)

b) O(n)

c) O(log n)

d) O(n^2)

Answer: b) O(n)

---

20. What is the time complexity of inserting a node at the end of a singly linked list (with tail pointer)?

a) O(1)

b) O(n)

c) O(log n)

d) O(n^2)

Answer: a) O(1)

---

21. How can you reverse a singly linked list?

a) Using a stack

b) Changing the direction of pointers iteratively or recursively

c) Using a queue

d) Sorting the list

Answer: b) Changing the direction of pointers iteratively or recursively

---

22. What is the time complexity of deleting a node without a head pointer in a linked list?

a) O(1)

b) O(n)

c) O(log n)

d) O(n^2)

Answer: a) O(1)

---

23. What is the time complexity of Binary Search?

a) O(n)

b) O(log n)

c) O(n log n)

d) O(n^2)

Answer: b) O(log n)

---

24. What does the Two Pointer Technique typically optimize?

a) Space complexity

b) Time complexity for array or list problems

c) Recursion depth

d) Sorting efficiency

Answer: b) Time complexity for array or list problems

---

25. What does Kadane's Algorithm compute?

a) The maximum sum subarray

b) The minimum element

c) The median of an array

d) The sorted array

Answer: a) The maximum sum subarray

---

26. What is recursion?

a) A loop that repeats indefinitely

b) A function that calls itself to solve a problem

c) A method to traverse arrays

d) A sorting technique

Answer: b) A function that calls itself to solve a problem

---

27. Which is generally more space-efficient: recursive or iterative solutions?

a) Recursive

b) Iterative

c) Both are equal

d) Depends on the compiler

Answer: b) Iterative

---

28. What is the base condition in a recursive Binary Search?

a) When the array is empty

b) When the start index exceeds the end index

c) When the element is found

d) When the array is sorted

Answer: b) When the start index exceeds the end index

---

29. What is the key difference between Binary Search and Linear Search?

a) Binary Search requires a sorted array

b) Linear Search is faster

c) Binary Search uses more memory

d) Linear Search requires recursion

Answer: a) Binary Search requires a sorted array

---

30. What is the time complexity of finding the lower bound in a sorted array using Binary Search?

a) O(n)

b) O(log n)

c) O(n log n)

d) $O(n^2)$

Answer: b) O(log n)

---

31. How do you find the number of occurrences of an element in a sorted array in O(log n)?

a) Use Linear Search

b) Use Binary Search to find the first and last occurrence

c) Traverse the array

d) Use a stack

Answer: b) Use Binary Search to find the first and last occurrence

---

32. Which problem involves searching in a rotated sorted array?

a) Finding the maximum element

b) Finding a target in a cyclically shifted sorted array

c) Reversing an array

d) Merging two arrays

Answer: b) Finding a target in a cyclically shifted sorted array

---

33. What is a Set in JavaScript?

a) A collection of key-value pairs

b) A collection of unique values

c) A sorted array

d) A stack implementation

Answer: b) A collection of unique values

---

34. Which method adds an element to a JavaScript Map?

a) map.add(key, value)

b) map.set(key, value)

c) map.push(key, value)

d) map.insert(key, value)

Answer: b) map.set(key, value)

---

35. What is the key difference between Set and WeakSet in JavaScript?

a) WeakSet allows primitive values only

b) WeakSet holds weak references to objects

c) Set is slower than WeakSet

d) WeakSet allows duplicates

Answer: b) WeakSet holds weak references to objects

---

36. What is a Stack?

a) A Last-In-First-Out (LIFO) data structure

b) A First-In-First-Out (FIFO) data structure

c) A sorted list

d) A random access structure

Answer: a) A Last-In-First-Out (LIFO) data structure

---

37. What is the time complexity of push and pop operations in a Stack?

a) O(1)

b) O(n)

c) O(log n)

d) $O(n^2)$

Answer: a) O(1)

---

38. Which problem uses a Stack to find the "Next Greater Element"?

a) Reversing a string

b) Finding the maximum element in an array

c) Finding the next larger element for each array element

d) Balancing parentheses

Answer: c) Finding the next larger element for each array element

---

39. How do you check for balanced parentheses using a Stack?

a) Push opening brackets, pop on matching closing brackets

b) Count the number of brackets

c) Use a queue to store brackets

d) Sort the brackets

Answer: a) Push opening brackets, pop on matching closing brackets

---

40. How can you reverse a string using a Stack in JavaScript?

a) Split the string, push to stack, pop all elements

b) Use string.reverse()

c) Use a queue

d) Concatenate the string backwards

Answer: a) Split the string, push to stack, pop all elements

---

41. What is a Queue?

a) A Last-In-First-Out (LIFO) structure

b) A First-In-First-Out (FIFO) structure

c) A random access structure

d) A sorted list

Answer: b) A First-In-First-Out (FIFO) structure

---

42. What is the time complexity of enqueue in a Queue?

a) O(1)

b) O(n)

c) O(log n)

d) $O(n^2)$

Answer: a) O(1)

---

43. How can you implement a Stack using two Queues?

a) Use one queue for pushing, another for popping

b) Make push costly by moving elements between queues

c) Use a single queue

d) Reverse the queues

Answer: b) Make push costly by moving elements between queues

---

44. How can you implement a Queue using two Stacks?

a) Use one stack for enqueue, another for dequeue

b) Make dequeue costly by transferring elements

c) Use a single stack

d) Sort the stacks

Answer: b) Make dequeue costly by transferring elements

---

45. What is the space complexity of Merge Sort?

a) O(1)

b) O(n)

c) O(n log n)

d) O(n^2)

Answer: b) O(n)

---

46. What is the time complexity of updating an element in a linked list?

a) O(1) if position known, O(n) otherwise

b) O(n) always

c) O(log n)

d) O(n^2)

Answer: a) O(1) if position known, O(n) otherwise

---

47. Which sorting algorithm is stable?

a) Quick Sort

b) Selection Sort

c) Merge Sort

d) Bubble Sort

Answer: c) Merge Sort

---

48. What is the time complexity of finding the upper bound in a sorted array?

a) O(n)

b) O(log n)

c) O(n log n)

d) O(n^2)

Answer: b) O(log n)

---

49. Which JavaScript method removes duplicates from an array efficiently?

a) arr.filter()

b) new Set(arr)

c) arr.sort()

d) arr.reduce()

Answer: b) new Set(arr)

---

50. What is the time complexity of searching in a Map in JavaScript?

a) O(1)

b) O(n)

c) O(log n)

d) O(n^2)

Answer: a) O(1)

---

51. Which algorithm is used to find the maximum subarray sum?

a) Binary Search

b) Kadane's Algorithm

c) Quick Sort

d) Two Pointer Technique

Answer: b) Kadane's Algorithm

---

52. What is the time complexity of reversing a linked list iteratively?

a) O(1)

b) O(n)

c) O(log n)

d) O(n^2)

Answer: b) O(n)
