100 Data Structures MCQs - Comprehensive Test

ArrayDeque, Deque, Priority Queue, Queue, and Stack

# Instructions

This comprehensive test contains 100 multiple choice questions covering five fundamental data structures. Each section contains 20 questions focusing on operations, time complexity, applications, and implementation details. Choose the best answer for each question.

**Question Distribution:**• Questions 1-20: Stack  
• Questions 21-40: Queue  
• Questions 41-60: ArrayDeque  
• Questions 61-80: Deque  
• Questions 81-100: Priority Queue

# STACK QUESTIONS (1-20)

## Question 1

What is the time complexity of the push operation in a stack implemented using a dynamic array?

A) O(1) always  
B) O(1) amortized  
C) O(n) always  
D) O(log n)

**Answer: B) O(1) amortized**

## Question 2

Which principle does a stack follow?

A) FIFO (First In First Out)  
B) LIFO (Last In First Out)  
C) Random Access  
D) Priority Based

**Answer: B) LIFO (Last In First Out)**

## Question 3

In postfix expression evaluation, what happens when an operator is encountered?

A) Push it onto the stack  
B) Pop two operands, apply operator, push result  
C) Pop one operand and apply operator  
D) Ignore the operator

**Answer: B) Pop two operands, apply operator, push result**

## Question 4

What exception is thrown when pop() is called on an empty Stack in Java?

A) NullPointerException  
B) EmptyStackException  
C) IndexOutOfBoundsException  
D) IllegalStateException

**Answer: B) EmptyStackException**

## Question 5

Which data structure is best for implementing function call management?

A) Queue  
B) Stack  
C) Priority Queue  
D) Deque

**Answer: B) Stack**

## Question 6

In balanced parentheses checking, when should you return false?

A) When stack is empty after processing all characters  
B) When encountering opening bracket  
C) When stack is empty but closing bracket is encountered  
D) When stack size exceeds array length

**Answer: C) When stack is empty but closing bracket is encountered**

## Question 7

Which algorithm uses stack for traversal?

A) Breadth-First Search  
B) Depth-First Search  
C) Dijkstra's Algorithm  
D) Binary Search

**Answer: B) Depth-First Search**

## Question 8

What is the space complexity of a stack with n elements?

A) O(1)  
B) O(log n)  
C) O(n)  
D) O(n²)

**Answer: C) O(n)**

## Question 9

In infix to postfix conversion, operators with higher precedence are:

A) Pushed to stack immediately  
B) Popped from stack before pushing new operator  
C) Ignored completely  
D) Added to output immediately

**Answer: B) Popped from stack before pushing new operator**

## Question 10

Which operation is NOT typically provided by a stack?

A) push()  
B) pop()  
C) peek()  
D) get(index)

**Answer: D) get(index)**

## Question 11

What happens in stack overflow?

A) Stack becomes empty  
B) Stack exceeds maximum capacity  
C) Stack operations become slow  
D) Stack gets corrupted

**Answer: B) Stack exceeds maximum capacity**

## Question 12

Which approach is used to reverse a string using stack?

A) Push all characters, then pop all  
B) Push alternate characters  
C) Use two stacks  
D) Sort the characters

**Answer: A) Push all characters, then pop all**

## Question 13

In expression evaluation, parentheses have:

A) Lowest precedence  
B) Highest precedence  
C) Medium precedence  
D) No precedence

**Answer: B) Highest precedence**

## Question 14

Stack can be used to check:

A) Palindrome strings  
B) Balanced parentheses  
C) Sorted arrays  
D) Prime numbers

**Answer: B) Balanced parentheses**

## Question 15

The peek() operation in stack:

A) Removes top element  
B) Returns top element without removing  
C) Adds new element  
D) Clears the stack

**Answer: B) Returns top element without removing**

## Question 16

Which is true about stack implementation using linked list?

A) Has fixed size limit  
B) No size limit theoretically  
C) Slower than array implementation  
D) Cannot implement peek()

**Answer: B) No size limit theoretically**

## Question 17

In tower of Hanoi problem, minimum number of moves for n disks is:

A) n  
B) n²  
C) 2ⁿ - 1  
D) 2ⁿ

**Answer: C) 2ⁿ - 1**

## Question 18

Stack is used in:

A) Undo operations  
B) Browser back button  
C) Recursion implementation  
D) All of the above

**Answer: D) All of the above**

## Question 19

What is the result of postfix expression '2 3 + 4 \*'?

A) 14  
B) 20  
C) 24  
D) 32

**Answer: B) 20**

## Question 20

Which stack operation has O(1) time complexity?

A) Only push  
B) Only pop  
C) Both push and pop  
D) Neither push nor pop

**Answer: C) Both push and pop**

# QUEUE QUESTIONS (21-40)

## Question 21

What principle does a queue follow?

A) LIFO  
B) FIFO  
C) Random Access  
D) Priority Based

**Answer: B) FIFO**

## Question 22

In circular queue implementation, when is the queue considered full?

A) front == rear  
B) (rear + 1) % capacity == front  
C) rear == capacity - 1  
D) front == 0

**Answer: B) (rear + 1) % capacity == front**

## Question 23

Which algorithm commonly uses queue data structure?

A) Depth-First Search  
B) Breadth-First Search  
C) Binary Search  
D) Quick Sort

**Answer: B) Breadth-First Search**

## Question 24

What is the time complexity of enqueue operation in a queue?

A) O(1)  
B) O(log n)  
C) O(n)  
D) O(n²)

**Answer: A) O(1)**

## Question 25

In a circular queue, when is it empty?

A) front == rear  
B) front == -1  
C) rear == -1  
D) front > rear

**Answer: A) front == rear**

## Question 26

Which Java class implements Queue interface with thread safety?

A) LinkedList  
B) ArrayDeque  
C) ConcurrentLinkedQueue  
D) ArrayList

**Answer: C) ConcurrentLinkedQueue**

## Question 27

Queue is used in:

A) CPU scheduling  
B) Handling requests in web servers  
C) BFS traversal  
D) All of the above

**Answer: D) All of the above**

## Question 28

What happens when dequeue is called on empty queue?

A) Returns null  
B) Throws exception  
C) Returns -1  
D) Depends on implementation

**Answer: D) Depends on implementation**

## Question 29

Which is NOT a queue operation?

A) enqueue  
B) dequeue  
C) push  
D) front

**Answer: C) push**

## Question 30

In level order traversal of binary tree, which data structure is used?

A) Stack  
B) Queue  
C) Priority Queue  
D) Hash Table

**Answer: B) Queue**

## Question 31

Circular queue avoids:

A) Memory wastage  
B) Overflow condition  
C) Underflow condition  
D) Time complexity issues

**Answer: A) Memory wastage**

## Question 32

Queue can be implemented using:

A) Array only  
B) Linked List only  
C) Both array and linked list  
D) Stack only

**Answer: C) Both array and linked list**

## Question 33

In producer-consumer problem, which data structure is commonly used?

A) Stack  
B) Queue  
C) Tree  
D) Graph

**Answer: B) Queue**

## Question 34

Priority of elements in a simple queue is based on:

A) Value of elements  
B) Size of elements  
C) Order of insertion  
D) Alphabetical order

**Answer: C) Order of insertion**

## Question 35

Double-ended queue is also known as:

A) Deque  
B) Stack  
C) Priority Queue  
D) Circular Queue

**Answer: A) Deque**

## Question 36

Queue overflow occurs when:

A) Queue is empty  
B) Queue is full and new element is to be added  
C) Front pointer exceeds rear pointer  
D) Queue contains only one element

**Answer: B) Queue is full and new element is to be added**

## Question 37

In array implementation of queue, which pointers are maintained?

A) Only front  
B) Only rear  
C) Both front and rear  
D) Top pointer

**Answer: C) Both front and rear**

## Question 38

Queue follows which memory allocation?

A) LIFO  
B) FIFO  
C) Random  
D) Priority based

**Answer: B) FIFO**

## Question 39

Breadth-first search uses queue to:

A) Store visited nodes  
B) Store nodes to be visited  
C) Store path information  
D) Store distances

**Answer: B) Store nodes to be visited**

## Question 40

Which operation views the front element without removing it?

A) enqueue  
B) dequeue  
C) peek/front  
D) rear

**Answer: C) peek/front**

# ARRAYDEQUE QUESTIONS (41-60)

## Question 41

ArrayDeque in Java is backed by:

A) Fixed size array  
B) Resizable array  
C) Linked List  
D) Hash Table

**Answer: B) Resizable array**

## Question 42

What is the default initial capacity of ArrayDeque?

A) 10  
B) 16  
C) 8  
D) 32

**Answer: B) 16**

## Question 43

ArrayDeque allows:

A) null elements  
B) duplicate elements  
C) Both null and duplicate elements  
D) Only duplicate elements

**Answer: B) duplicate elements**

## Question 44

Which method adds element to the front of ArrayDeque?

A) addLast()  
B) addFirst()  
C) add()  
D) offer()

**Answer: B) addFirst()**

## Question 45

ArrayDeque implements which interfaces?

A) Queue only  
B) Deque only  
C) Both Queue and Deque  
D) List interface

**Answer: C) Both Queue and Deque**

## Question 46

How does ArrayDeque handle capacity expansion?

A) Increases by 10 elements  
B) Doubles the capacity  
C) Increases by 50%  
D) Asks user for new size

**Answer: B) Doubles the capacity**

## Question 47

ArrayDeque is preferred over Stack because:

A) It's faster  
B) No capacity restrictions  
C) More consistent and complete interface  
D) All of the above

**Answer: D) All of the above**

## Question 48

Which operation is O(n) in ArrayDeque?

A) addFirst()  
B) removeFirst()  
C) contains()  
D) addLast()

**Answer: C) contains()**

## Question 49

ArrayDeque is thread-safe:

A) True  
B) False  
C) Only for read operations  
D) Only for write operations

**Answer: B) False**

## Question 50

Which is true about ArrayDeque iterator?

A) It's bidirectional  
B) It's fail-fast  
C) It allows modification during iteration  
D) It's thread-safe

**Answer: B) It's fail-fast**

## Question 51

ArrayDeque uses which strategy for array management?

A) Linear array  
B) Circular array  
C) Jagged array  
D) Multi-dimensional array

**Answer: B) Circular array**

## Question 52

Space complexity of ArrayDeque with n elements is:

A) O(1)  
B) O(n)  
C) O(log n)  
D) O(n²)

**Answer: B) O(n)**

## Question 53

ArrayDeque can be used as:

A) Stack only  
B) Queue only  
C) Both stack and queue  
D) Priority queue

**Answer: C) Both stack and queue**

## Question 54

Which method removes and returns the last element?

A) removeLast()  
B) pollLast()  
C) Both A and B  
D) getLast()

**Answer: C) Both A and B**

## Question 55

ArrayDeque memory overhead compared to LinkedList:

A) Higher  
B) Lower  
C) Same  
D) Depends on number of elements

**Answer: B) Lower**

## Question 56

Which method throws exception when ArrayDeque is empty?

A) poll()  
B) peek()  
C) remove()  
D) offer()

**Answer: C) remove()**

## Question 57

ArrayDeque provides random access:

A) Yes, through interface  
B) No, not through interface  
C) Only for reading  
D) Only for writing

**Answer: B) No, not through interface**

## Question 58

Best case time complexity for adding element to ArrayDeque:

A) O(1)  
B) O(log n)  
C) O(n)  
D) O(n²)

**Answer: A) O(1)**

## Question 59

ArrayDeque maintains insertion order:

A) Always  
B) Never  
C) Only when used as queue  
D) Only when used as stack

**Answer: A) Always**

## Question 60

Which is the most efficient way to iterate through ArrayDeque?

A) for loop with get(i)  
B) Iterator  
C) Enhanced for loop  
D) Both B and C

**Answer: D) Both B and C**

# DEQUE QUESTIONS (61-80)

## Question 61

Deque stands for:

A) Double-ended queue  
B) Duplicate queue  
C) Dynamic queue  
D) Distributed queue

**Answer: A) Double-ended queue**

## Question 62

Deque allows insertion and deletion at:

A) Front only  
B) Rear only  
C) Both front and rear  
D) Middle only

**Answer: C) Both front and rear**

## Question 63

Which problem is efficiently solved using deque?

A) Finding shortest path  
B) Sliding window maximum  
C) Binary search  
D) Sorting array

**Answer: B) Sliding window maximum**

## Question 64

In sliding window maximum problem, deque stores:

A) Array values  
B) Array indices  
C) Maximum values  
D) Window sizes

**Answer: B) Array indices**

## Question 65

Deque can simulate:

A) Stack only  
B) Queue only  
C) Both stack and queue  
D) Priority queue only

**Answer: C) Both stack and queue**

## Question 66

Which operation is NOT supported by deque?

A) addFirst()  
B) addLast()  
C) addMiddle()  
D) removeFirst()

**Answer: C) addMiddle()**

## Question 67

Deque interface extends:

A) Collection interface  
B) Queue interface  
C) List interface  
D) Set interface

**Answer: B) Queue interface**

## Question 68

Time complexity of addFirst() in deque:

A) O(1)  
B) O(log n)  
C) O(n)  
D) O(n²)

**Answer: A) O(1)**

## Question 69

Which of these implements Deque interface?

A) ArrayDeque  
B) LinkedList  
C) Both A and B  
D) ArrayList

**Answer: C) Both A and B**

## Question 70

Deque is useful for:

A) Palindrome checking  
B) Undo/Redo operations  
C) BFS with 0-1 weights  
D) All of the above

**Answer: D) All of the above**

## Question 71

In palindrome checking using deque, we compare:

A) Front and rear elements  
B) All elements with first  
C) Adjacent elements  
D) Random elements

**Answer: A) Front and rear elements**

## Question 72

Restricted deque allows:

A) Insertion at one end only  
B) Deletion at one end only  
C) Either insertion or deletion restriction at one end  
D) No restrictions

**Answer: C) Either insertion or deletion restriction at one end**

## Question 73

Output-restricted deque allows deletion only at:

A) Front  
B) Rear  
C) One specific end  
D) Both ends with restrictions

**Answer: C) One specific end**

## Question 74

Input-restricted deque allows insertion only at:

A) Front  
B) Rear  
C) One specific end  
D) Middle

**Answer: C) One specific end**

## Question 75

Deque can be implemented using:

A) Circular array  
B) Doubly linked list  
C) Both A and B  
D) Binary tree

**Answer: C) Both A and B**

## Question 76

Which method returns but doesn't remove front element?

A) removeFirst()  
B) peekFirst()  
C) pollFirst()  
D) addFirst()

**Answer: B) peekFirst()**

## Question 77

Deque supports iteration in:

A) Forward direction only  
B) Backward direction only  
C) Both directions  
D) No iteration support

**Answer: C) Both directions**

## Question 78

Maximum elements that can be stored in deque:

A) Fixed limit  
B) Integer.MAX\_VALUE  
C) Depends on implementation  
D) 1024

**Answer: C) Depends on implementation**

## Question 79

Which is true about deque performance?

A) O(1) for both end operations  
B) O(n) for middle operations  
C) O(log n) for search  
D) Both A and B

**Answer: D) Both A and B**

## Question 80

Deque can replace:

A) Stack in all applications  
B) Queue in all applications  
C) Both stack and queue in most applications  
D) Priority queue in all applications

**Answer: C) Both stack and queue in most applications**

# PRIORITY QUEUE QUESTIONS (81-100)

## Question 81

Priority Queue is typically implemented using:

A) Array  
B) Linked List  
C) Binary Heap  
D) Hash Table

**Answer: C) Binary Heap**

## Question 82

Time complexity of insertion in binary heap-based priority queue:

A) O(1)  
B) O(log n)  
C) O(n)  
D) O(n log n)

**Answer: B) O(log n)**

## Question 83

Java's PriorityQueue is:

A) Min-heap by default  
B) Max-heap by default  
C) Can be both depending on comparator  
D) Both A and C

**Answer: D) Both A and C**

## Question 84

Which algorithm uses priority queue?

A) Dijkstra's shortest path  
B) Prim's MST  
C) Huffman coding  
D) All of the above

**Answer: D) All of the above**

## Question 85

Priority queue allows:

A) Null elements  
B) Duplicate elements  
C) Both null and duplicate  
D) Neither null nor duplicate

**Answer: B) Duplicate elements**

## Question 86

Time complexity of removing arbitrary element from priority queue:

A) O(1)  
B) O(log n)  
C) O(n)  
D) O(n log n)

**Answer: C) O(n)**

## Question 87

Building heap from unsorted array takes:

A) O(n log n)  
B) O(n)  
C) O(log n)  
D) O(n²)

**Answer: B) O(n)**

## Question 88

Priority queue is used in:

A) Operating system scheduling  
B) Event simulation  
C) Graph algorithms  
D) All of the above

**Answer: D) All of the above**

## Question 89

What happens with equal priority elements in PriorityQueue?

A) FIFO order maintained  
B) LIFO order maintained  
C) Order is not guaranteed  
D) Exception is thrown

**Answer: C) Order is not guaranteed**

## Question 90

Which method extracts minimum element from min-heap?

A) poll()  
B) peek()  
C) remove()  
D) Both A and C

**Answer: D) Both A and C**

## Question 91

Priority queue iteration order is:

A) Sorted order  
B) Insertion order  
C) Priority order  
D) Undefined/arbitrary

**Answer: D) Undefined/arbitrary**

## Question 92

Heap property in min-heap states:

A) Parent ≤ children  
B) Parent ≥ children  
C) Left child ≤ right child  
D) All leaves are at same level

**Answer: A) Parent ≤ children**

## Question 93

Space complexity of priority queue with n elements:

A) O(1)  
B) O(log n)  
C) O(n)  
D) O(n²)

**Answer: C) O(n)**

## Question 94

PriorityQueue in Java is:

A) Thread-safe  
B) Not thread-safe  
C) Partially thread-safe  
D) Thread-safe for reads only

**Answer: B) Not thread-safe**

## Question 95

Which operation maintains heap property after insertion?

A) Heapify down  
B) Heapify up  
C) Both A and B  
D) No operation needed

**Answer: B) Heapify up**

## Question 96

Maximum height of binary heap with n nodes:

A) log n  
B) n  
C) √n  
D) n/2

**Answer: A) log n**

## Question 97

Priority queue can implement:

A) Heap sort  
B) Job scheduling  
C) Graph algorithms  
D) All of the above

**Answer: D) All of the above**

## Question 98

Which is true about complete binary tree in heap?

A) All levels filled except possibly last  
B) Last level filled left to right  
C) Both A and B  
D) Balanced tree

**Answer: C) Both A and B**

## Question 99

Custom comparator in PriorityQueue is used for:

A) Changing default ordering  
B) Handling custom objects  
C) Implementing max-heap  
D) All of the above

**Answer: D) All of the above**

## Question 100

Time complexity of peek() operation in priority queue:

A) O(1)  
B) O(log n)  
C) O(n)  
D) O(n log n)

**Answer: A) O(1)**

# COMPLETE ANSWER KEY

**STACK (1-20):**1. B 2. B 3. B 4. B 5. B 6. C 7. B 8. C 9. B 10. D   
11. B 12. A 13. B 14. B 15. B 16. B 17. C 18. D 19. B 20. C  **QUEUE (21-40):**  
21. B 22. B 23. B 24. A 25. A 26. C 27. D 28. D 29. C 30. B   
31. A 32. C 33. B 34. C 35. A 36. B 37. C 38. B 39. B 40. C  **ARRAYDEQUE (41-60):**  
41. B 42. B 43. B 44. B 45. C 46. B 47. D 48. C 49. B 50. B   
51. B 52. B 53. C 54. C 55. B 56. C 57. B 58. A 59. A 60. D  **DEQUE (61-80):**  
61. A 62. C 63. B 64. B 65. C 66. C 67. B 68. A 69. C 70. D   
71. A 72. C 73. C 74. C 75. C 76. B 77. C 78. C 79. D 80. C  **PRIORITY QUEUE (81-100):**  
81. C 82. B 83. D 84. D 85. B 86. C 87. B 88. D 89. C 90. D   
91. D 92. A 93. C 94. B 95. B 96. A 97. D 98. C 99. D 100. A

# SCORING GUIDE

**Excellent (90-100):** Outstanding mastery of all data structures  
**Very Good (80-89):** Strong understanding with minor gaps  
**Good (70-79):** Solid knowledge, some areas need review  
**Fair (60-69):** Basic understanding, significant study needed  
**Below 60:** Requires comprehensive review of all topics  
  
**Topic-wise Analysis:**• Stack (Q1-20): LIFO operations, expression evaluation, applications  
• Queue (Q21-40): FIFO operations, BFS, circular implementation  
• ArrayDeque (Q41-60): Resizable arrays, performance, dual functionality  
• Deque (Q61-80): Double-ended operations, sliding window problems  
• Priority Queue (Q81-100): Heap operations, algorithms, complexity

# COMPREHENSIVE STUDY GUIDE

**STACK - Key Concepts to Master:**• LIFO principle and basic operations (push, pop, peek)  
• Expression evaluation (infix, postfix, prefix)  
• Balanced parentheses checking algorithm  
• Stack applications: DFS, function calls, undo operations  
• Implementation using arrays and linked lists  
• Time complexity analysis and space considerations  
  
**QUEUE - Essential Knowledge:**• FIFO principle and core operations (enqueue, dequeue)  
• Circular queue implementation and advantages  
• BFS algorithm and level-order tree traversal  
• Producer-consumer problem and applications  
• Thread-safe implementations (ConcurrentLinkedQueue)  
• Queue overflow and underflow conditions  
  
**ARRAYDEQUE - Important Points:**• Resizable circular array implementation  
• Capacity expansion strategy (doubling)  
• Null element restrictions and fail-fast iterators  
• Performance comparison with ArrayList and LinkedList  
• Dual functionality as stack and queue  
• Memory efficiency and cache locality benefits  
  
**DEQUE - Core Understanding:**• Double-ended queue concept and operations  
• Sliding window maximum/minimum algorithms  
• Palindrome checking and string processing  
• Input/output restricted deque variants  
• Implementation using circular arrays and doubly linked lists  
• Bidirectional iteration capabilities  
  
**PRIORITY QUEUE - Advanced Topics:**• Binary heap properties (min-heap and max-heap)  
• Heapify operations (up and down)  
• Custom comparators and natural ordering  
• Graph algorithms: Dijkstra's, Prim's MST  
• Heap construction from unsorted array  
• Applications in scheduling and simulation

# TIME COMPLEXITY SUMMARY

**Operation Complexity Comparison:  
  
STACK:**• Push: O(1) amortized  
• Pop: O(1)  
• Peek: O(1)  
• Search: O(n)  
  
**QUEUE:**• Enqueue: O(1)  
• Dequeue: O(1)  
• Front: O(1)  
• Search: O(n)  
  
**ARRAYDEQUE:**• Add (both ends): O(1) amortized  
• Remove (both ends): O(1)  
• Random access: Not supported  
• Contains: O(n)  
  
**DEQUE:**• Add/Remove (both ends): O(1)  
• Middle operations: O(n)  
• Search: O(n)  
• Iteration: O(n)  
  
**PRIORITY QUEUE:**• Insert: O(log n)  
• Extract min/max: O(log n)  
• Peek: O(1)  
• Remove arbitrary: O(n)  
• Build heap: O(n)

# RECOMMENDED PRACTICE PROBLEMS

**BEGINNER LEVEL:**1. Implement stack using arrays and linked lists  
2. Evaluate simple postfix expressions  
3. Check for balanced parentheses  
4. Implement queue using two stacks  
5. Reverse a string using stack  
  
**INTERMEDIATE LEVEL:**1. Next greater element using stack  
2. Sliding window maximum using deque  
3. Design circular queue  
4. Implement min stack (getMin in O(1))  
5. Level order traversal using queue  
  
**ADVANCED LEVEL:**1. Largest rectangle in histogram  
2. Implement LRU cache using deque  
3. Merge k sorted lists using priority queue  
4. Design browser history with undo/redo  
5. Implement Dijkstra's algorithm

# ADDITIONAL RESOURCES

Online Platforms:  
• LeetCode: Stack, Queue, Heap problem sets  
• HackerRank: Data Structures track  
• GeeksforGeeks: Comprehensive articles and problems  
• Codeforces: Advanced algorithmic problems  
  
Visualization Tools:  
• VisuAlgo: Interactive data structure animations  
• Data Structure Visualizations (USF)  
• Algorithm Visualizer  
  
Books:  
• 'Introduction to Algorithms' by CLRS  
• 'Data Structures and Algorithms in Java' by Goodrich  
• 'Algorithms' by Robert Sedgewick  
  
Java Documentation:  
• Official Java Collections Framework documentation  
• Oracle Java tutorials on collections  
• OpenJDK source code for implementation details