Trees

1. What is the top node in a tree called?

a) Leaf b) Root c) Sibling d) Child

2. A node with no children is called:

a) Root b) Interior node c) Leaf d) Parent

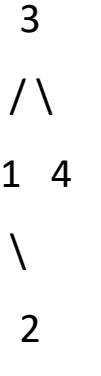
3. In a tree, nodes sharing the same parent are called:

a) Ancestors b) Descendants c) Siblings d) Leaves

4. Which traversal visits nodes in the order: Left, Root, Right?

a) Pre-order b) In-order c) Post-order d) Level-order

5. What is the output of an in-order traversal for this BST?



a) 1, 2, 3, 4 b) 3, 1, 2, 4 c) 1, 3, 2, 4 d) 3, 4, 1, 2

6. Which traversal is used for deleting a tree?

a) Pre-order b) In-order c) Post-order d) Breadth-first

7. The maximum depth of a tree is its: a) Size b) Height c) Level d) Degree

8. A collection of trees is called a: a) Forest b) Graph c) Heap d) Array

9. In a binary tree, each node has at most:

a) 1 child b) 2 children c) 3 children d) Unlimited children

10. Which traversal uses a queue?

a) Pre-order b) In-order c) Post-order d) Breadth-first

11. What is the height of a tree with only a root node?

a) 0 b) 1 c) 2 d) Undefined

12. The node directly above another node is its:

a) Child b) Sibling c) Parent d) Ancestor

13. Which traversal visits the root first?

a) Pre-order b) In-order c) Post-order d) Level-order

14. A node with at least one child is called:

a) Leaf b) Interior node c) Root d) Sibling

15. In-order traversal of a BST returns elements in:

a) Random order b) Descending order

c) Ascending order d) Level order

16. What is the time complexity to calculate the height of a tree?

a) O(1) b) O(log n) c) O(n) d) $O(n^2)$

17. Which of the following is **not** a tree application?

a) Syntax parsing b) Database indexing

c) Linear search d) File systems

18. A tree where each node has 0 or 2 children is called:

a) Full binary tree b) Complete binary tree

c) Perfect binary tree d) Balanced tree

19. Recursive algorithms are commonly used for:

a) Arrays b) Trees c) Linked lists d) Queues

20. The ancestors of a node include:

a) Only its parent b) Parent and siblings

c) Parent, grandparent, etc. d) Only its children

Arrays (15 Questions)

21. Arrays provide _____ time access to any element.

a) Linear b) Constant c) Logarithmic d) Quadratic

22. What is the formula for row-major order in a 2D array?

a) `array_addr + elem_size × (i + j)`

b) `array_addr + elem_size × (i × cols + j)`

c) `array_addr + elem_size × (j × rows + i)`

d) `array_addr + elem_size × (i × j)`

23. Adding an element at the end of an array takes:

a) O(1) b) O(n) c) O(log n) d) $O(n^2)$

24. Which operation is O(n) in arrays?

a) Accessing the first element b) Inserting at the beginning

c) Updating the last element d) Deleting the last element

25. In column-major order, elements are stored by:

a) Rows b) Columns c) Diagonals d) Randomly

26. The memory allocation for arrays is:

a) Non-contiguous b) Contiguous c) Linked d) Hashed

27. What is the worst-case time to insert in the middle of an array?

a) O(1) b) O(n) c) O(log n) d) $O(n^2)$

28. Which is **not** a limitation of arrays?

a) Fixed size b) Slow insertion at the beginning

c) Fast access by index d) Costly resizing

29. For a 3x4 array, how many elements are there?

a) 7 b) 12 c) 10 d) 5

30. The index of the first element in an array is typically:

a) 0 b) 1 c) -1 d) User-defined

32. Deleting the last element of an array is:

a) O(1) b) O(n) c) O(log n) d) $O(n^2)$

33. Multi-dimensional arrays are stored as:

a) Linked lists b) Contiguous blocks c) Graphs d) Trees

34. The size of an array is determined:

a) At runtime b) During compilation

c) After execution d) By the OS

35. Which operation is O(1) in arrays?

a) Linear search b) Binary search

c) Access by index d) Insertion at the middle

Searching/Sorting (15 Questions)

36. Linear search works on:

a) Only sorted arrays b) Only unsorted arrays

c) Both sorted and unsorted arrays d) Only trees

37. Binary search requires the data to be:

a) Unsorted b) Sorted c) Random d) Hashed

38. The worst-case time complexity of linear search is:

a) O(1) b) O(log n) c) O(n) d) $O(n^2)$

39. Binary search reduces the search space by:

a) 1/4 each step b) 1/2 each step c) 1/3 each step d) 2/3 each step

40. The best-case time for binary search is:

a) O(1) b) O(log n) c) O(n) d) $O(n^2)$

41. Bubble sort compares:

a) Distant elements b) Adjacent elements

c) Random elements d) Every third element

42. The worst-case time of bubble sort is:

a) O(n) b) O(n log n) c) $O(n^2)$ d) O(1)

43. Insertion sort is efficient for:

a) Large unsorted arrays b) Small or nearly sorted arrays

c) Reverse-sorted arrays d) Linked lists

44. Which algorithm uses a pivot element?

a) Bubble sort b) Quick sort c) Insertion sort d) Merge sort

45. The best-case time of insertion sort is:

a) O(n) b) $O(n^2)$ c) O(log n) d) O(1)

46. Which is **not** a comparison-based sort?

a) Bubble sort b) Quick sort c) Counting sort d) Merge sort

47. Binary search is an example of:

a) Divide-and-conquer b) Greedy algorithm

c) Dynamic programming d) Backtracking

48. The first step in insertion sort is to:

a) Swap adjacent elements b) Choose a pivot

c) Mark the first element as sorted d) Split the array

49. How many passes does bubble sort need for an n-element array?

a) 1 b) n c) n^2 d) log n

50. Which sorting algorithm is adaptive?

a) Selection sort b) Insertion sort c) Merge sort d) Heap sort

Answers:

1. b	2. c	3. c	4. b	5. a	6. c	7. b	8. a	9. b	10. d
11. b	12. c	13. a	14. b	15. c	16. c	17. c	18. a	19. b	20. c
21. b	22. b	23. a	24. b	25. b	26. b	27. b	28. c	29. b	30. a
31. *	32. a	33. b	34. b	35. c	36. c	37. b	38. c	39. b	40. a
41. b	42. c	43. b	44. b	45. a	46. c	47. a	48. c	49. b	50. b

Linked list :

1. **What is the time complexity of `PushFront` in a singly-linked list?**

   a) O(n) b) O(1) c) O(log n) d) O(n^2)

2. **Which operation in a singly-linked list without a tail pointer has a time complexity of O(n)?**

   a) `PushFront` b) `PushBack` c) `PopFront` d) `TopFront`

3. **What is the purpose of the `tail` pointer in a singly-linked list?**

   a) To reduce the time complexity of `PushBack` to O(1)

   b) To reduce the time complexity of `PopFront` to O(1)

   c) To make `Find` operation O(1) d) To reverse the list

4. **In a singly-linked list with a tail pointer, what is the time complexity of `TopBack`?**

   a) O(n) b) O(1) c) O(log n) d) O(n^2)

5. **What is the time complexity of `PopBack` in a singly-linked list (with or without a tail)?**

   a) O(1) b) O(n) c) O(log n) d) O(n^2)

6. **Which of the following operations is O(1) in a singly-linked list?**

   a) `Find` b) `PushBack` (without tail) c) `PopFront` d) `Erase`

7. **What is the time complexity of `AddAfter(Node, Key)` in a singly-linked list?**

   a) O(n) b) O(1) c) O(log n) d) O(n^2)

8. **What is the time complexity of `AddBefore(Node, Key)` in a singly-linked list?**

   a) O(n) b) O(1) c) O(log n) d) O(n^2)

9. **In a doubly-linked list, what additional pointer does each node have compared to a singly-linked list?**

a) `head` pointer b) `tail` pointer c) `prev` pointer d) `parent` pointer

10. **What is the time complexity of `PopBack` in a doubly-linked list with a tail pointer?**

a) O(n) b) O(1) c) O(log n) d) $O(n^2)$

11. **Which operation in a doubly-linked list is O(1) due to the `prev` pointer?**

a) `Find` b) `AddBefore` c) `PushFront` d) `Empty`

12. **What is the time complexity of `PushBack` in a doubly-linked list with a tail pointer?**

a) O(n) b) O(1) c) O(log n) d) $O(n^2)$

13. **In a singly-linked list, which operation requires traversing the entire list in the worst case?**

a) `TopFront` b) `PopFront` c) `Find` d) `PushFront`

14. **What is the time complexity of `Erase(Key)` in a singly-linked list?**

a) O(1) b) O(n) c) O(log n) d) $O(n^2)$

15. **Which of the following is true about a doubly-linked list?**

a) `AddBefore` is O(n) b) `PopBack` is O(n) even with a tail pointer

c) `AddAfter` is O(1) d) All of the above

16. **What is the time complexity of `Empty()` in both singly and doubly-linked lists?**

a) O(n) b) O(1) c) O(log n) d) $O(n^2)$

17. **In a singly-linked list, what happens during `PopFront` if the list is empty?**

a) It returns `nil` b) It throws an error c) It adds a new node d) It does nothing

18. **What is the time complexity of `TopFront()` in a singly-linked list?**

a) O(n) b) O(1) c) O(log n) d) $O(n^2)$

19. **Which of the following operations is O(n) in a doubly-linked list without a tail pointer?**

a) `PushFront` b) `PushBack` c) `PopFront` d) `TopFront`

20. **What is the key advantage of a doubly-linked list over a singly-linked list?**

a) Faster `Find` operation b) Constant-time `AddBefore` and `AddAfter`

c) No need for a `head` pointer d) Uses less memory

21. **In a singly-linked list, what is the first step of the `PushFront(Key)` operation?**

a) Traverse the list to find the tail b) Create a new node with the given key

c) Update the `tail` pointer d) Delete the `head` node

22. **What is the time complexity of `Find(Key)` in both singly and doubly-linked lists?**

a) O(1) b) O(n) c) O(log n) d) $O(n^2)$

23. **Which pointer is updated during `PopBack` in a doubly-linked list?**

a) `head` b) `tail.prev` c) `head.next` d) `tail.next`

24. **What is the purpose of the `Empty()` function in the List API?**

a) To check if the list is empty b) To delete all nodes in the list

c) To reverse the list d) To sort the list

25. **In a doubly-linked list, what is the time complexity of `AddBefore(Node, Key)`?**

a) O(n) b) O(1) c) O(log n) d) O(n^2)

26. **Which of the following is NOT a property of a singly-linked list?**

a) Nodes have a `next` pointer b) `PopBack` is O(n)

c) `AddBefore` is O(1) d) `PushFront` is O(1)

27. **What is the time complexity of `TopBack()` in a singly-linked list without a tail pointer?**

a) O(1) b) O(n) c) O(log n) d) O(n^2)

28. **In a doubly-linked list, how is `AddAfter(Node, Key)` implemented?**

a) By traversing the list to find the node

b) By updating the `next` and `prev` pointers of adjacent nodes

c) By deleting the node first d) By reversing the list

29. **What is the time complexity of `Erase(Key)` in a doubly-linked list?**

a) O(1) b) O(n) c) O(log n) d) O(n^2)

30. **Which of the following operations is O(1) in a doubly-linked list but O(n) in a singly-linked list?**

a) `PushFront` b) `AddBefore` c) `PopFront` d) `TopFront`

Answers:

1. b	2. b	3. a	4. b	5. b	6. c	7. b	8. a	9. c	10. b
11. b	12. b	13. c	14. b	15. c	16. b	17. b	18. b	19. b	20. b
21. b	22. b	23. b	24. a	25. b	26. c	27. b	28. b	29. b	30. b

Stacks and Queues Quiz (50 MCQs)

Stacks (Questions 1-25)

1. **What is the fundamental principle of a stack?**

   a) FIFO b) LIFO c) Priority-based d) Random access

2. **Which operation adds an element to the stack?**

   a) Pop() b) Push() c) Top() d) Dequeue()

3. **What does the `Pop()` operation do?**

   a) Returns the top element without removing it

   b) Removes and returns the top element

   c) Checks if the stack is empty

   d) Reverses the stack

4. **What is the time complexity of `Push()` in a stack?**

   a) O(n) b) O(1) c) O(log n) d) $O(n^2)$

5. **What happens if you call `Pop()` on an empty stack?**

   a) Returns null b) Returns 0

   c) Causes an underflow error d) Creates a new element

6. **Which data structure is best suited for implementing a stack?**

   a) Array or linked list b) Hash table c) Binary tree d) Graph

7. **In a linked list implementation of a stack, where does `Push()` insert a new element?**

   a) At the tail b) At the head c) In the middle d) Randomly

8. **What does the `Top()` operation return?**

a) The least recently added element b) The most recently added element

c) The middle element d) The size of the stack

9. **Which of the following is a balanced parentheses string?**

a) `([)]` b) `([])` c) `][` d) `(((`

10. **What is the output after these operations: `Push(A)`, `Push(B)`, `Pop()`?**

a) `[A]` b) `[B]` c) `[A, B]` d) `[]`

11. **In an array-based stack, what does `numElements` track?**

a) The total size of the array b) The number of elements currently in the stack

c) The position of the top element d) The number of empty slots

12. **What happens when you `Push()` to a full array-based stack?**

a) The stack resizes automatically b) An overflow error occurs

c) The oldest element is removed d) The stack becomes empty

13. **What is the final state after: `Push(X)`, `Push(Y)`, `Pop()`, `Push(Z)`?**

a) `[X, Z]` b) `[Y, Z]` c) `[X, Y]` d) `[Z]`

14. **Which algorithm uses a stack to check balanced parentheses?**

a) Binary search b) Depth-first search c) `IsBalanced()` d) Quick sort

15. **What is the time complexity of all stack operations?**

a) O(1) b) O(n) c) O(log n) d) $O(n^2)$

16. **What does LIFO stand for?**

a) Last In, First Out b) First In, First Out c) Linear In, Fixed Out d) Linked In, First Out

17. **In a linked list stack, where is `Pop()` performed?**

a) At the tail b) At the head c) At a random node d) In the middle

18. **What is the output of `Top()` after `Push(1)`, `Push(2)`, `Push(3)`?**

a) `1` b) `2` c) `3` d) `0`

19. **What is the result of `Pop()` after `Push(A)`, `Push(B)`, `Push(C)`?**

a) `A` b) `B` c) `C` d) `null`

20. **Which of the following is NOT a stack operation?**

a) Push b) Pop c) Enqueue d) Top

21. **What happens when `Empty()` is called on a non-empty stack?**

a) Returns `True` b) Returns `False` c) Returns `0` d) Returns the stack size

22. **In an array-based stack, what does `Push()` do?**

a) Decrements `numElements` b) Increments `numElements`

c) Shifts all elements left d) Reverses the array

23. **What is the state of the stack after `Push(1)`, `Push(2)`, `Pop()`, `Push(3)`?**

a) `[1, 3]` b) `[2, 3]` c) `[1, 2]` d) `[3]`

24. **Which of the following is an unbalanced string?**

a) `()[]` b) `([{}])` c) `([)]` d) `[[]]`

25. **What is the purpose of `Empty()`?**

a) To check if the stack is full b) To check if the stack is empty

c) To count elements d) To reverse the stack

Queues (Questions 26-50)

26. **What is the fundamental principle of a queue?**

a) LIFO b) FIFO c) Priority-based d) Random access

27. **Which operation adds an element to the queue?**

a) Dequeue() b) Enqueue() c) Top() d) Pop()

28. **What does the `Dequeue()` operation do?**

a) Returns the front element without removing it b) Removes and returns the front element

c) Checks if the queue is empty d) Reverses the queue

29. **What is the time complexity of `Enqueue()` in a queue?**

a) O(n) b) O(1) c) O(log n) d) $O(n^2)$

30. **What happens if you call `Dequeue()` on an empty queue?**

a) Returns null b) Returns 0 c) Causes an underflow error d) Creates a new element

31. **Which data structure is best suited for implementing a queue?**

a) Array or linked list b) Stack c) Binary tree d) Hash table

32. **In a linked list implementation of a queue, where does `Enqueue()` insert a new element?**

a) At the head b) At the tail c) In the middle d) Randomly

33. **What does FIFO stand for?**

a) First In, First Out b) Last In, First Out c) Fixed In, Fixed Out d) Fast In, Fast Out

34. **What is the output after these operations: `Enqueue(A)`, `Enqueue(B)`, `Dequeue()`?**

a) `[A]` b) `[B]` c) `[A, B]` d) `[]`

35. **In a circular array-based queue, what problem does it solve?**

a) Stack overflow b) Memory fragmentation

c) Wasted space in a linear array d) Slow insertion

36. **What happens when you `Enqueue()` to a full array-based queue?**

a) The queue resizes automatically b) An overflow error occurs

c) The oldest element is removed d) The queue becomes empty

37. **What is the final state after: `Enqueue(X)`, `Enqueue(Y)`, `Dequeue()`, `Enqueue(Z)`?**

a) `[X, Z]` b) `[Y, Z]` c) `[X, Y]` d) `[Z]`

38. **What is the time complexity of all queue operations?**

a) O(1) b) O(n) c) O(log n) d) $O(n^2)$

39. **In a linked list queue, why is a tail pointer necessary?**

a) To allow O(1) `Enqueue` at the end b) To allow O(1) `Dequeue` at the front

c) To reverse the queue d) To count elements

40. **What is the output of `Dequeue()` after `Enqueue(1)`, `Enqueue(2)`, `Enqueue(3)`?**

a) `1` b) `2` c) `3` d) `null`

41. **Which of the following is NOT a queue operation?**

a) Enqueue b) Dequeue c) Push d) Empty

42. **What happens when `Empty()` is called on a non-empty queue?**

a) Returns `True` b) Returns `False` c) Returns `0` d) Returns the queue size

43. **In an array-based queue, what does `Enqueue()` do?**

a) Increments the `write` pointer b) Decrements the `read` pointer

c) Shifts all elements left d) Reverses the array

44. **What is the state of the queue after `Enqueue(A)`, `Enqueue(B)`, `Dequeue()`, `Enqueue(C)`?**

a) `[A, C]` b) `[B, C]` c) `[A, B]` d) `[C]`

45. **What is the purpose of `Empty()` in a queue?**

a) To check if the queue is full b) To check if the queue is empty

c) To count elements d) To reverse the queue

46. **In a circular array queue, how are `read` and `write` pointers handled when they reach the end?**

a) They stop b) They reset to 0 c) They swap places d) They reverse direction

47. **What is the result of `Dequeue()` after `Enqueue(P)`, `Enqueue(Q)`, `Enqueue(R)`?**

a) `P` b) `Q` c) `R` d) `null`

48. **Which of the following best describes a queue?**

a) Last In, First Out b) First In, First Out c) Priority-based d) Random access

49. **What happens if `Dequeue()` is called on an empty queue?**

a) Returns `null` b) Causes an underflow error

c) Returns `0` d) Creates a new element

50. **What is the final state after `Enqueue(10)`, `Enqueue(20)`, `Dequeue()`, `Enqueue(30)`?**

a) `[10, 30]` b) `[20, 30]` c) `[10, 20]` d) `[30]`

Answer Key

Stacks:

1-b, 2-b, 3-b, 4-b, 5-c, 6-a, 7-b, 8-b, 9-b, 10-a, 11-b, 12-b, 13-a, 14-c, 15-a, 16-a, 17-b, 18-c, 19-c, 20-c, 21-b, 22-b, 23-a, 24-c, 25-b

Queues:

26-b, 27-b, 28-b, 29-b, 30-c, 31-a, 32-b, 33-a, 34-b, 35-c, 36-b, 37-b, 38-a, 39-a, 40-a,

41-c, 42-b, 43-a, 44-b, 45-b, 46-b, 47-a, 48-b, 49-b, 50-b

Heap: Multiple-Choice Questions

1. What is the defining property of a binary max-heap?
a) Each node has at most two children
b) The root has the smallest value
c) Every parent node is greater than or equal to its children
d) The tree is always balanced

2. In a binary max-heap, which node contains the maximum value?
a) The root b) A leaf node c) The leftmost child d) The rightmost child

3. What is the time complexity of the GetMax operation in a binary heap?
a) O(n) b) O(log n) c) O(1) d) O(n log n)

4. What operation is used to restore the heap property after insertion?
a) SiftDown b) SiftUp c) ExtractMax d) Reheapify

5. Which operation removes the maximum element from a binary max-heap?
a) Insert b) ChangePriority c) ExtractMax d) Remove

6. What operation is used to restore the heap property after ExtractMax?
a) SiftUp b) Sort c) SiftDown d) Remove

7. In a complete binary tree, how are levels filled?
a) Right to left b) Randomly c) Left to right d) Top to bottom

8. What is the height of a complete binary tree with n nodes?
a) O(n) b) O(log n) c) O(n log n) d) O(1)

9. Where is a new element inserted in a binary heap?
a) At the root b) At the last level, rightmost
c) At the leftmost vacant position in the last level d) Anywhere

10. What happens after inserting an element into the heap?
a) It becomes the new root b) It replaces a random node
c) It sifts up if it violates the heap property d) It sifts down immediately

11. In array representation, what is the index of the left child of node i?
a) 2 * i b) i / 2 c) i + 1 d) 2 * i + 1

12. What is the index of the right child of node i?
a) 2 * i + 1 b) 2 * i c) i / 2 d) i - 1

13. What is the index of the parent of node i?
a) i / 2 b) 2 * i c) i + 1 d) i - 1

14. What is the worst-case time complexity of Insert in a binary heap?
a) O(1) b) O(n) c) O(log n) d) O(n log n)

15. What does the Remove operation internally do?
a) Directly delete the element b) Set priority to -∞ and ExtractMax
c) Replace with zero d) Ignore the element

16. What is the running time of HeapSort?
a) O(n) b) O(log n) c) O(n log n) d) $O(n^2)$

17. HeapSort is based on which data structure?
a) Binary search tree b) Linked list c) Heap d) Queue

18. HeapSort is considered:
a) In-place b) Recursive c) Graph-based d) Not in-place

19. In-place HeapSort builds the heap in which order?
a) Top to bottom b) Random c) Bottom to top d) Level by level

20. Which function is repeatedly called in HeapSort to maintain the heap?
a) SiftUp b) SiftDown c) Insert d) ChangePriority

21. Which two operations must a priority queue support efficiently?
a) Insert and ExtractMax b) Search and Delete
c) Sort and Merge d) Add and Subtract

22. What happens if we exceed the heap's maxSize during Insert?
a) Overflow error b) New size allocated
c) Root replaced d) No change

23. Which is true for SiftDown(i) if both children are smaller?
a) No swap happens b) Swap with left
c) Swap with right d) Delete the node

24. ChangePriority may trigger which operation?
a) Only SiftUp b) Only SiftDown
c) Either SiftUp or SiftDown d) Neither

25. A complete binary tree can be stored efficiently as:
a) Linked List b) Hash Table c) Array d) Set

26. What is the height of a complete binary tree with n nodes?
a) O(n) b) O(log n) c) O(n log n) d) O(1)

27. What happens after inserting an element into the heap?
a) It becomes the new root b) It replaces a random node
c) It sifts up if it violates the heap property d) It sifts down immediately

28. In a complete binary tree, how are levels filled?
a) Right to left b) Randomly c) Left to right d) Top to bottom

29. What is the defining property of a binary max-heap?
a) Each node has at most two children
b) The root has the smallest value
c) Every parent node is greater than or equal to its children
d) The tree is always balanced

30. Where is a new element inserted in a binary heap?
a) At the root b) At the last level, rightmost
c) At the leftmost vacant position in the last level d) Anywhere