**Section 1: Arrays and Basic Data Structures**

1. Which of the following best describes java.util.ArrayList?  
   A. A fixed-size array that cannot be resized.  
   B. A dynamic array that can grow or shrink in size.  
   C. A linked list implementation.  
   D. A data structure optimized for frequent insertions and deletions at arbitrary positions.  
   **Correct Answer: B**
2. What is the worst-case time complexity of accessing an element at a specific index in a java.util.ArrayList?  
   A. O(1)  
   B. O(log n)  
   C. O(n)  
   D. O(n log n)  
   **Correct Answer: A**
3. When using a primitive array int[] arr = new int[10]; in Java, what is the initial value of arr[0]?  
   A. null  
   B. 0  
   C. undefined  
   D. garbage value  
   **Correct Answer: B**
4. Which data structure is best suited for implementing a "redo" functionality in a text editor?  
   A. Stack  
   B. Queue  
   C. Linked List  
   D. Binary Search Tree  
   **Correct Answer: B**
5. What is the primary disadvantage of using a static array over a dynamic array (like ArrayList) for storing a collection of items?  
   A. Slower access time.  
   B. Inefficient memory usage.  
   C. Fixed size, requiring manual resizing and copying.  
   D. Cannot store heterogeneous data types.  
   **Correct Answer: C**
6. Which of the following operations has a worst-case time complexity of O(n) for a java.util.LinkedList?  
   A. Getting the first element.  
   B. Adding an element at the beginning.  
   C. Adding an element at the end.  
   D. Getting an element at a specific index i.  
   **Correct Answer: D**
7. A java.util.Stack in Java is a LIFO (Last-In, First-Out) collection. Which methods represent the push and pop operations?  
   A. add() and remove()  
   B. offer() and poll()  
   C. push() and pop()  
   D. enqueue() and dequeue()  
   **Correct Answer: C**
8. In a circular linked list, what points to the first node of the list?  
   A. The last node's next pointer.  
   B. A separate head pointer.  
   C. Both A and B.  
   D. Neither A nor B.  
   **Correct Answer: C**
9. Which of the following is true about a Doubly Linked List compared to a Singly Linked List?  
   A. It uses less memory.  
   B. It allows for efficient traversal in both forward and backward directions.  
   C. Insertion at the end is always O(1) without a tail pointer.  
   D. Deletion of a node requires traversing from the head.  
   **Correct Answer: B**
10. What is the time complexity of deleting a node from the middle of a Doubly Linked List, given a reference to that node?  
    A. O(1)  
    B. O(log n)  
    C. O(n)  
    D. O(n log n)  
    **Correct Answer: A**

**Section 2: Stacks and Queues**

1. Which data structure is essential for implementing a Depth-First Search (DFS) algorithm?  
   A. Queue  
   B. Stack  
   C. Priority Queue  
   D. Hash Table  
   **Correct Answer: B**
2. Evaluate the postfix expression: 8 2 + 5 \* 9 -  
   A. 41  
   B. 59  
   C. 30  
   D. 50  
   **Correct Answer: A** (8+2 = 10, 10\*5 = 50, 50-9 = 41)
3. How would you convert an infix expression (A + B) \* C to postfix notation?  
   A. AB+C\*  
   B. ABC+\*  
   C. A+B\*C  
   D. \*+ABC  
   **Correct Answer: A**
4. What happens if you try to pop() an element from an empty java.util.Stack?  
   A. It returns null.  
   B. It throws a NoSuchElementException.  
   C. It throws an EmptyStackException.  
   D. It returns the default value for the element type.  
   **Correct Answer: C**
5. What is a "deque" (double-ended queue)?  
   A. A queue where elements can only be added to the front.  
   B. A queue where elements can only be removed from the rear.  
   C. A queue that allows insertion and deletion from both ends.  
   D. A queue that prioritizes elements based on their values.  
   **Correct Answer: C**
6. Which of the following is an application of a queue data structure?  
   A. Function call stack for recursion.  
   B. Undo/Redo operations.  
   C. Implementing a buffer for I/O operations.  
   D. Checking for balanced parentheses.  
   **Correct Answer: C**
7. What is the time complexity to enqueue and dequeue an element in a queue implemented using a java.util.LinkedList (where new elements are added to the end and removed from the beginning)?  
   A. Enqueue: O(n), Dequeue: O(n)  
   B. Enqueue: O(1), Dequeue: O(1)  
   C. Enqueue: O(n), Dequeue: O(1)  
   D. Enqueue: O(1), Dequeue: O(n)  
   **Correct Answer: B**
8. What is the main benefit of using a circular queue over a linear queue implemented with an array?  
   A. Allows faster access to middle elements.  
   B. Prevents array overflow without resizing.  
   C. Efficiently utilizes array space by wrapping around.  
   D. Eliminates the need for front and rear pointers.  
   **Correct Answer: C**

**Section 3: Trees**

1. Which of the following tree traversals visits the left subtree, then the root, then the right subtree?  
   A. Pre-order  
   B. In-order  
   C. Post-order  
   D. Level-order  
   **Correct Answer: B**
2. For a Binary Search Tree (BST), what is the best-case time complexity for searching an element?  
   A. O(1)  
   B. O(log n)  
   C. O(n)  
   D. O(n log n)  
   **Correct Answer: B**
3. What characteristic defines a "full binary tree"?  
   A. Every node has either zero or two children.  
   B. All leaf nodes are at the same level.  
   C. All levels are completely filled.  
   D. All leaf nodes are at the same level, and all internal nodes have two children.  
   **Correct Answer: A** (Often confused, 'Full' means every node has 0 or 2 children. 'Complete' fills from left to right. 'Perfect' is both full and complete with all leaves at same level.)
4. In a balanced Binary Search Tree (like an AVL tree or Red-Black tree) with n nodes, what is the maximum possible height?  
   A. O(1)  
   B. O(log n)  
   C. O(n)  
   D. O(n log n)  
   **Correct Answer: B**
5. Which tree data structure is commonly used to implement a java.util.TreeMap or java.util.TreeSet in Java?  
   A. Binary Search Tree  
   B. AVL Tree  
   C. Red-Black Tree  
   D. B-Tree  
   **Correct Answer: C**
6. A min-heap property states that:  
   A. The value of each node is greater than or equal to the value of its parent.  
   B. The value of each node is less than or equal to the value of its parent.  
   C. The value of the root node is the maximum value in the heap.  
   D. The value of a left child is always less than its right child.  
   **Correct Answer: B**
7. What is the time complexity to insert an element into a min-heap with n elements?  
   A. O(1)  
   B. O(log n)  
   C. O(n)  
   D. O(n log n)  
   **Correct Answer: B**
8. Which traversal algorithm would you use to print all elements of a Binary Search Tree in sorted order?  
   A. Pre-order traversal  
   B. In-order traversal  
   C. Post-order traversal  
   D. Level-order traversal  
   **Correct Answer: B**
9. What is the purpose of performing rotations in an AVL tree?  
   A. To increase the height of the tree.  
   B. To keep the tree balanced and maintain O(log n) height.  
   C. To convert it into a complete binary tree.  
   D. To reduce the number of nodes.  
   **Correct Answer: B**

**Section 4: Graphs**

1. Which graph representation is more suitable when the graph is sparse (has few edges)?  
   A. Adjacency Matrix  
   B. Adjacency List  
   C. Incidence Matrix  
   D. Edge List  
   **Correct Answer: B**
2. What is the time complexity of Breadth-First Search (BFS) on a graph with V vertices and E edges, using an adjacency list representation?  
   A. O(V + E)  
   B. O(V \* E)  
   C. O(V^2)  
   D. O(E log V)  
   **Correct Answer: A**
3. Dijkstra's algorithm is used to find:  
   A. Minimum Spanning Tree  
   B. All-Pairs Shortest Path  
   C. Single-Source Shortest Path in graphs with non-negative edge weights.  
   D. Cycle detection in a directed graph.  
   **Correct Answer: C**
4. Which algorithm can detect negative cycles in a graph?  
   A. Dijkstra's Algorithm  
   B. Prim's Algorithm  
   C. Kruskal's Algorithm  
   D. Bellman-Ford Algorithm  
   **Correct Answer: D**
5. A graph where all edges have a direction (e.g., A -> B but not B -> A) is called a:  
   A. Undirected Graph  
   B. Weighted Graph  
   C. Directed Graph  
   D. Connected Graph  
   **Correct Answer: C**
6. Which of the following algorithms is used to find the Minimum Spanning Tree (MST)?  
   A. Floyd-Warshall  
   B. Prim's and Kruskal's  
   C. Breadth-First Search  
   D. Bellman-Ford  
   **Correct Answer: B**
7. In a connected, undirected graph with V vertices and E edges, what is the number of edges in any Minimum Spanning Tree (MST)?  
   A. V  
   B. V - 1  
   C. E  
   D. V + E  
   **Correct Answer: B**
8. Which data structure is typically used to keep track of visited nodes in graph traversal algorithms (BFS or DFS)?  
   A. Stack  
   B. Queue  
   C. Set/Boolean Array  
   D. Priority Queue  
   **Correct Answer: C**

**Section 5: Hashing and Advanced Data Structures**

1. What is a "collision" in a hash table?  
   A. When two distinct keys map to the same index.  
   B. When a key is not found in the hash table.  
   C. When the hash table runs out of memory.  
   D. When the hash function is too slow.  
   **Correct Answer: A**
2. Which collision resolution technique involves finding the next empty slot in the hash table sequentially?  
   A. Chaining  
   B. Quadratic Probing  
   C. Double Hashing  
   D. Linear Probing  
   **Correct Answer: D**
3. What is the worst-case time complexity for searching an element in a hash table that uses chaining for collision resolution?  
   A. O(1)  
   B. O(log n)  
   C. O(n)  
   D. O(n log n)  
   **Correct Answer: C** (When all elements hash to the same bucket, it degrades to a linked list search.)
4. What is the typical average-case time complexity for put() and get() operations in a java.util.HashMap?  
   A. O(1)  
   B. O(log n)  
   C. O(n)  
   D. O(n log n)  
   **Correct Answer: A**
5. What is a "Trie" (Prefix Tree) primarily used for?  
   A. Storing key-value pairs with O(1) average access.  
   B. Efficient searching and auto-completion for strings based on prefixes.  
   C. Maintaining a sorted list of elements.  
   D. Representing hierarchical data like file systems.  
   **Correct Answer: B**

**Section 6: Algorithm Design and Analysis**

1. Which of the following concepts is central to Dynamic Programming?  
   A. Recursion without memoization.  
   B. Making locally optimal choices.  
   C. Overlapping subproblems and optimal substructure.  
   D. Exhaustive search of all possibilities.  
   **Correct Answer: C**
2. What does "Big O" notation (O) represent in algorithm analysis?  
   A. The exact running time of an algorithm.  
   B. The lower bound of an algorithm's running time.  
   C. The worst-case (upper bound) running time of an algorithm.  
   D. The average-case running time of an algorithm.  
   **Correct Answer: C**
3. An algorithm has a time complexity of O(log n). If n increases from 100 to 1,000,000, how does the number of operations roughly change?  
   A. Increases by a factor of 10.  
   B. Increases by a factor of 20.  
   C. Increases exponentially.  
   D. Increases linearly.  
   **Correct Answer: B** (log₂100 ≈ 6.64, log₂1,000,000 ≈ 19.93. Roughly a factor of 3, but 20 is the closest reasonable order of magnitude for the options.) \*Correction for precision: log base 2 of 100 is roughly 6.64, log base 2 of 1,000,000 is roughly 19.93. The difference is 13.29. A factor of 20 implies a very large increase, whereas the operations only triple in magnitude. Let's re-evaluate options or make it more distinct. Assuming common log here for illustrative purposes, or if its natural log, the ratio is small. If it refers to *multiplication* by factor of 20, that's not log n behaviour. If the question implies *approximately* what's the growth, and 100 to 1M is 10^4 increase in n, log(10^4 \* n) vs log(n) is log(10^4) = 4 \* log(10) ~ 13. If n is from 100 to 1,000,000, that's 10,000x increase in n. log(10000n) vs log(n) difference is log(10000) which is about 13-14 for base 2. B is the most plausible among given options if looking for a rough multiplicative factor in the *logarithmic value* itself, not the original n.) Let's refine for clarity.
   * Let's take log\_2. log\_2(100) approx 6.64. log\_2(1,000,000) approx 19.93.
   * The *number of operations* goes from ~7 units to ~20 units. This is roughly 3 times.
   * Given the options, "increases by a factor of 10" or "20" could be interpreted differently. If it means "the *factor* by which n increased is 10,000", then log(10,000) is around 13-14. "Increases by a factor of 20" is misleading.
   * Let's re-evaluate what they're testing. log(10^x \* N) = log(10^x) + log(N). So log(10^4 \* 100) = log(10^6). The *addition* is log(10^4).
   * If f(n) = C \* log(n), then f(10^6) / f(100) = log(10^6) / log(100) = 6 / 2 = 3. So, it increases by a factor of 3.
   * Given the options, none are perfect. This suggests a potentially flawed question or options. However, if forced to choose, "increases by a factor of 20" is definitely too high. "10" is too high. "Exponentially" is wrong. "Linearly" is wrong.
   * **Re-thinking the common intuition:** log(n) grows *very slowly*. A million-fold increase in n leads to a very small additive increase in log n. If 100 is 7, 1M is 20. The *absolute difference* is 13. The *ratio* is 3.
   * Let's consider if the question meant n increases *by* 100 to 1,000,000 which is n=100 to N=1,000,000. Then the operations go from log(100) to log(1,000,000). The *increase* in operations is log(1,000,000) - log(100) = log(10^6 / 10^2) = log(10^4) = 4 \* log(10). If base 2, this is 4 \* 3.32 = 13.28.
   * Since it's MCQs, there might be an intended rough estimate. Let's pick the closest non-obviously wrong one. "A factor of 20" is likely not the intent. "A factor of 10" is also not the intent.
   * **Let's assume the question is asking for the *new value* of log(N) relative to log(n) in order of magnitude, or perhaps testing how slow log grows.**
   * If log\_10(100) = 2, log\_10(10^6) = 6. Factor of 3.
   * If log\_2(100) = 6.64, log\_2(10^6) = 19.93. Factor of 3.
   * None of the options perfectly align with the factor. But the *increase* is around 13 units. "Increases by a factor of 10" would mean from X to 10X.
   * This is a tricky question with bad options if precision is expected. However, if it's about N increasing to N', and log(N') / log(N). It's log(1,000,000) / log(100).
   * If forced, none of A, B, C, D are good. Let's rephrase this for my output to avoid ambiguity. I'll make it simpler.

**Revised Q43 for my list:**  
43. An algorithm has a time complexity of O(log n). If n increases from 1000 to 1,000,000, how does the number of operations roughly change?  
A. It triples  
B. It increases by a factor of 1000  
C. It increases by a fixed additive amount  
D. It increases exponentially  
**Correct Answer: A** (log\_10(1000) = 3, log\_10(1,000,000) = 6. Factor of 2. For log\_2, log\_2(1000)~10, log\_2(1,000,000)~20. Factor of 2. So "doubles" or "triples" are reasonable estimates for this range.) Let's go with "It roughly doubles" or "It increases by a small constant factor". "Triples" is a reasonable estimate from 6.64 to 19.93. Let's stick to the spirit of *small change*.

**Let's use a simpler one if the previous one is flawed.**  
43. An algorithm has a time complexity of O(log n). If n doubles, how does the number of operations roughly change?  
A. It doubles  
B. It triples  
C. It increases by a small constant amount  
D. It increases exponentially  
**Correct Answer: C** (e.g., log(2n) = log(2) + log(n). So it adds log(2) which is a small constant.) This is a better question.

1. Which algorithm design paradigm involves solving subproblems once and storing their solutions for future use?  
   A. Greedy Method  
   B. Divide and Conquer  
   C. Dynamic Programming  
   D. Backtracking  
   **Correct Answer: C**
2. The Master Theorem is primarily used for:  
   A. Analyzing the time complexity of iterative algorithms.  
   B. Solving recurrence relations for recursive algorithms.  
   C. Determining the space complexity of an algorithm.  
   D. Proving algorithm correctness.  
   **Correct Answer: B**
3. What is the space complexity of an in-place sorting algorithm?  
   A. O(1)  
   B. O(log n)  
   C. O(n)  
   D. O(n log n)  
   **Correct Answer: A** (ignoring recursion stack space if applicable)

**Section 7: Searching and Sorting**

1. Which of the following sorting algorithms is generally considered unstable?  
   A. Merge Sort  
   B. Insertion Sort  
   C. Quick Sort  
   D. Bubble Sort  
   **Correct Answer: C** (Quick sort is typically unstable. Merge sort, Insertion sort, Bubble sort are stable in their common implementations.)
2. What is the prerequisite for performing a Binary Search on an array?  
   A. The array must be unsorted.  
   B. The array must contain unique elements.  
   C. The array must be sorted.  
   D. The array must be a fixed size.  
   **Correct Answer: C**
3. Which sorting algorithm's worst-case time complexity is O(n^2), but its best-case (already sorted array) is O(n)?  
   A. Quick Sort  
   B. Merge Sort  
   C. Insertion Sort  
   D. Heap Sort  
   **Correct Answer: C**
4. What is the primary characteristic of a "stable" sorting algorithm?  
   A. Its time complexity remains constant regardless of input.  
   B. It maintains the relative order of equal elements.  
   C. It requires O(1) extra space.  
   D. It sorts elements in linear time.  
   **Correct Answer: B**

**Section 1: Java Collections & Interfaces**

1. Which java.util interface guarantees that elements are stored in a unique, unordered collection?  
   A. java.util.List  
   B. java.util.Set  
   C. java.util.Map  
   D. java.util.Queue  
   **Correct Answer: B**
2. What is the underlying data structure for java.util.HashSet?  
   A. Array  
   B. Doubly Linked List  
   C. Hash Table  
   D. Red-Black Tree  
   **Correct Answer: C**
3. Which of the following java.util.Map implementations maintains insertion order?  
   A. HashMap  
   B. TreeMap  
   C. LinkedHashMap  
   D. Hashtable  
   **Correct Answer: C**
4. Which java.util.Queue method is used to add an element, returning false if the queue is full (without throwing an exception)?  
   A. add()  
   B. offer()  
   C. put()  
   D. insert()  
   **Correct Answer: B**
5. What is the time complexity of adding an element to a java.util.TreeSet?  
   A. O(1) average  
   B. O(log n)  
   C. O(n)  
   D. O(n log n)  
   **Correct Answer: B**
6. Which java.util class implements the List interface using a dynamic array?  
   A. LinkedList  
   B. Vector  
   C. ArrayList  
   D. Stack  
   **Correct Answer: C**
7. A java.util.PriorityQueue is a priority-based queue. What is its default ordering behavior in Java?  
   A. Elements are retrieved in insertion order (FIFO).  
   B. Elements are retrieved in reverse insertion order (LIFO).  
   C. Elements are retrieved based on their natural ordering or a provided Comparator, with the *smallest* element having the highest priority.  
   D. Elements are retrieved based on their natural ordering or a provided Comparator, with the *largest* element having the highest priority.  
   **Correct Answer: C**
8. What is the typical time complexity for Collections.sort() (which uses Timsort) on an ArrayList in the average case?  
   A. O(n)  
   B. O(n log n)  
   C. O(n^2)  
   D. O(log n)  
   **Correct Answer: B**

**Section 2: Recursion and Backtracking**

1. What is the essential component of a recursive function that prevents infinite recursion?  
   A. A loop statement.  
   B. A base case.  
   C. A return type of void.  
   D. A global variable.  
   **Correct Answer: B**
2. Which common error can occur if a recursive function does not have a proper base case or the recursive call does not converge towards it?  
   A. NullPointerException  
   B. ArrayIndexOutOfBoundsException  
   C. StackOverflowError  
   D. OutOfMemoryError  
   **Correct Answer: C**
3. What is Backtracking primarily used for?  
   A. Finding the shortest path in a graph.  
   B. Optimizing resource allocation.  
   C. Solving problems by exploring all possible solutions step-by-step and abandoning paths that do not lead to a valid solution.  
   D. Sorting elements in an array.  
   **Correct Answer: C**
4. The N-Queens problem is a classic example solved using which algorithmic technique?  
   A. Greedy Approach  
   B. Dynamic Programming  
   C. Backtracking  
   D. Divide and Conquer  
   **Correct Answer: C**

**Section 3: Sorting Algorithms (Advanced)**

1. Which of the following sorting algorithms is a non-comparison sort?  
   A. Quick Sort  
   B. Heap Sort  
   C. Counting Sort  
   D. Merge Sort  
   **Correct Answer: C**
2. For Counting Sort to be efficient, what characteristic must the input data have?  
   A. It must be a large range of integer values.  
   B. It must be nearly sorted.  
   C. It must have a small and known range of integer values.  
   D. It must consist of floating-point numbers.  
   **Correct Answer: C**
3. What is the time complexity of Radix Sort?  
   A. O(n + k) where k is the range of values.  
   B. O(d \* (n + k)) where d is number of digits and k is radix.  
   C. O(n log n)  
   D. O(n^2)  
   **Correct Answer: B**
4. An external sorting algorithm is typically used when:  
   A. The data is already partially sorted.  
   B. The data set is too large to fit into main memory.  
   C. The data needs to be sorted in-place.  
   D. The data consists of small, fixed-size records.  
   **Correct Answer: B**
5. Which of the following statements about Quicksort's worst-case scenario is true?  
   A. It occurs when the pivot always divides the array into two equal halves.  
   B. It occurs when the array is already sorted or reverse-sorted, and a bad pivot choice is made.  
   C. It has a time complexity of O(n log n).  
   D. It has a space complexity of O(1).  
   **Correct Answer: B**
6. What is the main advantage of Merge Sort over Quick Sort in general?  
   A. Faster worst-case time complexity.  
   B. It is an in-place sorting algorithm.  
   C. It is always stable.  
   D. Requires less auxiliary space.  
   **Correct Answer: C**

**Section 4: Trees (Advanced)**

1. Which operation is performed to restore the heap property after an element is inserted or deleted in a binary heap?  
   A. Rotation  
   B. Heapify (percolate up/down)  
   C. Traversal  
   D. Balancing  
   **Correct Answer: B**
2. A binary tree where every internal node has exactly two children, and all leaf nodes are at the same level, is called a:  
   A. Complete Binary Tree  
   B. Full Binary Tree  
   C. Perfect Binary Tree  
   D. Skewed Binary Tree  
   **Correct Answer: C**
3. What is the maximum number of nodes in a binary tree of height h (where root is at height 0)?  
   A. 2^h  
   B. 2^(h+1) - 1  
   C. h+1  
   D. 2h  
   **Correct Answer: B**
4. Which of the following is an application of a Binary Search Tree (BST)?  
   A. Implementing a cache (Least Recently Used).  
   B. Storing data that needs to be accessed quickly in sorted order.  
   C. Representing relationships in social networks.  
   D. Managing tasks in an operating system's scheduler.  
   **Correct Answer: B**
5. A B-tree is primarily used in:  
   A. In-memory data caching.  
   B. Database indexing and file systems.  
   C. Network routing protocols.  
   D. Compressing data.  
   **Correct Answer: B**
6. In a Trie data structure, what does each node typically represent?  
   A. A complete word.  
   B. A character or a part of a string.  
   C. A hash value.  
   D. A numeric key.  
   **Correct Answer: B**

**Section 5: Graphs (Advanced)**

1. Breadth-First Search (BFS) is primarily used for:  
   A. Finding cycles in a directed graph.  
   B. Finding the shortest path in an unweighted graph.  
   C. Topological sorting of a graph.  
   D. Detecting negative cycles.  
   **Correct Answer: B**
2. Which of the following is a common application of Depth-First Search (DFS)?  
   A. Finding the shortest path in a weighted graph.  
   B. Detecting cycles in a graph.  
   C. Minimum Spanning Tree generation.  
   D. Finding all-pairs shortest paths.  
   **Correct Answer: B**
3. What is a "bridge" in an undirected graph?  
   A. An edge whose removal increases the number of connected components.  
   B. An edge that connects two nodes with different degrees.  
   C. An edge that is part of a cycle.  
   D. An edge with the maximum weight.  
   **Correct Answer: A**
4. What is a "cut vertex" (or articulation point) in an undirected graph?  
   A. A vertex whose removal makes the graph bipartite.  
   B. A vertex whose removal increases the number of connected components.  
   C. A vertex with the highest degree.  
   D. A vertex that is part of every cycle.  
   **Correct Answer: B**
5. Topological Sort can only be applied to which type of graph?  
   A. Undirected Graph  
   B. Directed Acyclic Graph (DAG)  
   C. Weighted Graph  
   D. Complete Graph  
   **Correct Answer: B**
6. What is the primary use case for Topological Sort?  
   A. Finding the shortest path between two nodes.  
   B. Scheduling tasks with dependencies.  
   C. Detecting negative cycles.  
   D. Finding the minimum spanning tree.  
   **Correct Answer: B**
7. If a graph has V vertices and E edges, what is the space complexity of an adjacency matrix representation?  
   A. O(V + E)  
   B. O(V \* E)  
   C. O(V^2)  
   D. O(E^2)  
   **Correct Answer: C**
8. The Floyd-Warshall algorithm is used to find:  
   A. Single-source shortest path in a graph with non-negative weights.  
   B. Minimum Spanning Tree.  
   C. All-Pairs Shortest Paths.  
   D. Strongly Connected Components.  
   **Correct Answer: C**
9. What distinguishes a Strongly Connected Component (SCC) in a directed graph?  
   A. It is a subgraph where every pair of vertices is connected by an edge in at least one direction.  
   B. It is a subgraph where every pair of vertices has a path from the first to the second and from the second to the first.  
   C. It is a subgraph with the maximum number of edges.  
   D. It is a subgraph that forms a cycle.  
   **Correct Answer: B**

**Section 6: Hashing (Advanced)**

1. What is the "load factor" in a hash table?  
   A. The number of elements currently in the table.  
   B. The total capacity of the hash table.  
   C. The ratio of the number of elements to the table size (n/m).  
   D. The average number of probes required to find an element.  
   **Correct Answer: C**
2. When the load factor of a hash table exceeds a certain threshold, what operation is typically performed to maintain performance?  
   A. Rehashing (resizing the table).  
   B. Switching to a different hash function.  
   C. Clearing the table.  
   D. Converting to a linked list.  
   **Correct Answer: A**
3. Which collision resolution technique is generally considered better for cache performance due to better locality of reference?  
   A. Linear Probing  
   B. Quadratic Probing  
   C. Double Hashing  
   D. Chaining  
   **Correct Answer: D** (Chaining, because linked lists nodes can be allocated anywhere. Among open addressing, linear probing exploits cache lines better than quadratic or double hashing, which jump around more.) *Correction*: For open addressing, linear probing has good cache locality because it checks adjacent cells. Chaining involves following pointers which can be scattered in memory. The question implies open addressing.

**Revisiting Q36:** While chaining's *individual buckets* might be good if they are small arrays, if the question compares collision resolution *techniques*, Linear Probing often shows better cache performance than Quadratic Probing or Double Hashing because it accesses contiguous memory locations. Let's adjust the correct answer based on this common understanding in competitive programming/systems.

**Revised Q36:** Which open addressing collision resolution technique is generally considered better for cache performance due to better locality of reference?  
A. Linear Probing  
B. Quadratic Probing  
C. Double Hashing  
D. None of the above  
**Correct Answer: A**

1. What is the main drawback of linear probing for collision resolution?  
   A. It requires a linked list for each bucket.  
   B. It can lead to primary clustering.  
   C. It has a higher memory overhead.  
   D. It cannot handle deletions efficiently.  
   **Correct Answer: B**
2. A good hash function should ideally:  
   A. Map all keys to the same index.  
   B. Distribute keys uniformly across the hash table.  
   C. Be computationally very complex.  
   D. Always avoid collisions.  
   **Correct Answer: B**

**Section 7: Algorithm Design & Analysis (Advanced)**

1. What does O(n!) (factorial time) typically imply about an algorithm?  
   A. It is highly efficient for large inputs.  
   B. It is generally suitable for problems with very small input sizes.  
   C. It is a linear time algorithm.  
   D. It is a polynomial time algorithm.  
   **Correct Answer: B**
2. An algorithm that makes the locally optimal choice at each step with the hope of finding a global optimum is characteristic of which paradigm?  
   A. Dynamic Programming  
   B. Divide and Conquer  
   C. Greedy Method  
   D. Backtracking  
   **Correct Answer: C**
3. Which of the following problems can be efficiently solved using a Greedy approach?  
   A. 0/1 Knapsack Problem  
   B. Fractional Knapsack Problem  
   C. Longest Common Subsequence  
   D. Travelling Salesperson Problem  
   **Correct Answer: B**
4. What is "memoization" in the context of Dynamic Programming?  
   A. Storing input values in memory.  
   B. Storing the results of expensive function calls and returning the cached result when the same inputs occur again.  
   C. Writing comments in code for clarity.  
   D. Using a stack to manage recursive calls.  
   **Correct Answer: B**
5. The "optimal substructure" property in Dynamic Programming means:  
   A. The problem can be divided into independent subproblems.  
   B. An optimal solution to a problem contains optimal solutions to its subproblems.  
   C. The algorithm runs in optimal time complexity.  
   D. The problem has overlapping subproblems.  
   **Correct Answer: B**
6. If T(n) = 2T(n/2) + O(n), what is the time complexity T(n) according to the Master Theorem?  
   A. O(log n)  
   B. O(n)  
   C. O(n log n)  
   D. O(n^2)  
   **Correct Answer: C** (This is Case 2 of Master Theorem where a = 2, b = 2, f(n) = n^1. n^(log\_b a) = n^(log\_2 2) = n^1. Since f(n) is O(n^1), T(n) = O(n log n))
7. Amortized analysis is used to:  
   A. Find the worst-case time complexity of an algorithm.  
   B. Find the best-case time complexity of an algorithm.  
   C. Determine the average performance of an operation over a sequence of operations.  
   D. Calculate the exact running time of an algorithm.  
   **Correct Answer: C**
8. What is the space complexity of a recursive algorithm that calls itself n times, and each call adds a constant amount to the stack?  
   A. O(1)  
   B. O(log n)  
   C. O(n)  
   D. O(n log n)  
   **Correct Answer: C** (Due to the recursion stack depth)

**Section 8: General DSA Concepts & Java Specifics**

1. What is an Abstract Data Type (ADT)?  
   A. A specific implementation of a data structure.  
   B. A mathematical model for a data structure that defines the operations that can be performed on the data.  
   C. A concrete class in Java.  
   D. A pointer to a memory location.  
   **Correct Answer: B**
2. In Java, how do objects within a data structure (like nodes in a linked list or elements in an ArrayList) primarily reference each other?  
   A. By their physical memory address.  
   B. By their index number in a contiguous block.  
   C. Through references (pointers).  
   D. By copying the entire object data.  
   **Correct Answer: C**
3. Which of the following scenarios would typically benefit most from using a java.util.LinkedHashMap?  
   A. When you need O(1) average time complexity for lookups and don't care about insertion order.  
   B. When you need elements sorted by key and O(log n) performance.  
   C. When you need to maintain the order of insertion or access, in addition to key-value mapping.  
   D. When memory efficiency is the absolute top priority.  
   **Correct Answer: C**
4. What is the main purpose of garbage collection in Java, relevant to data structures?  
   A. To manually free up memory allocated by the programmer.  
   B. To prevent memory leaks by automatically reclaiming memory no longer referenced by the program.  
   C. To speed up array access.  
   D. To manage thread synchronization.  
   **Correct Answer: B**

**Section 1: Java Collections Deep Dive & Best Practices**

1. In Java, which method of the Object class must be overridden if equals() is overridden, to ensure proper functioning of hash-based collections like HashMap or HashSet?  
   A. clone()  
   B. toString()  
   C. hashCode()  
   D. finalize()  
   **Correct Answer: C**
2. What is the primary reason to make keys immutable when using them in java.util.HashMap?  
   A. To prevent NullPointerException.  
   B. To ensure their hashCode() remains constant, guaranteeing correct retrieval.  
   C. To allow serialization of the HashMap.  
   D. To make the HashMap thread-safe.  
   **Correct Answer: B**
3. Which Java Collection class is thread-safe and has its methods synchronized?  
   A. ArrayList  
   B. LinkedList  
   C. Vector  
   D. HashMap  
   **Correct Answer: C`**
4. If you need a List implementation that is optimized for adding/removing elements from the middle of the list frequently, which java.util class would you typically choose?  
   A. ArrayList  
   B. Vector  
   C. LinkedList  
   D. Stack  
   **Correct Answer: C**
5. What is the main advantage of java.util.ArrayDeque over java.util.LinkedList when implementing a queue or a stack?  
   A. It is thread-safe by default.  
   B. It uses less memory.  
   C. It provides better performance (amortized O(1)) due to underlying array and cache locality.  
   D. It allows for random access in O(1) time.  
   **Correct Answer: C**
6. Which java.util.Collection interface guarantees iteration in the natural order of elements (or by a Comparator)?  
   A. List  
   B. Set  
   C. SortedSet  
   D. Queue  
   **Correct Answer: C`**
7. What is the purpose of the iterator() method in Java's Iterable interface?  
   A. To get an element at a specific index.  
   B. To check if the collection is empty.  
   C. To obtain an object that can traverse over the elements of the collection.  
   D. To add elements to the collection.  
   **Correct Answer: C**

**Section 2: Advanced Tree Concepts**

1. In a Red-Black Tree, what is the maximum number of black nodes on any path from the root to a leaf?  
   A. log(n)  
   B. n/2  
   C. n  
   D. 2 \* log(n)  
   **Correct Answer: D** (This refers to the property that the longest path from root to leaf is no more than twice the length of the shortest path, where shortest paths consist only of black nodes.)
2. Which property is *not* a characteristic of a Red-Black Tree?  
   A. Every node is either red or black.  
   B. Every leaf (NIL node) is black.  
   C. If a node is red, then both its children are black.  
   D. All leaf nodes are at the same depth.  
   **Correct Answer: D**
3. What is a "segment tree" primarily used for?  
   A. Storing hierarchical data like file systems.  
   B. Efficiently performing range queries (e.g., sum, min, max) and point updates on an array.  
   C. Representing relationships between entities in a network.  
   D. Optimizing pathfinding in weighted graphs.  
   **Correct Answer: B**
4. What is the time complexity to find the minimum element in a min-heap?  
   A. O(1)  
   B. O(log n)  
   C. O(n)  
   D. O(n log n)  
   **Correct Answer: A**
5. The number of edges in a tree with N vertices is always:  
   A. N  
   B. N - 1  
   C. log N  
   D. N \* (N-1) / 2  
   **Correct Answer: B**
6. How does the iterative in-order traversal of a binary tree typically use a stack?  
   A. To store nodes visited after the right child.  
   B. To store parent nodes whose left subtrees have been processed but roots and right subtrees are pending.  
   C. To keep track of the path from the root to the current node.  
   D. It does not use a stack; it uses a queue.  
   **Correct Answer: B**

**Section 3: Graph Algorithms (Advanced Applications)**

1. What is the main application of the Disjoint Set Union (DSU) data structure?  
   A. Finding shortest paths in a graph.  
   B. Detecting cycles and managing connected components in a graph.  
   C. Sorting elements efficiently.  
   D. Storing key-value pairs.  
   **Correct Answer: B**
2. Kruskal's algorithm for finding the Minimum Spanning Tree relies on which data structure for efficient cycle detection?  
   A. Adjacency List  
   B. Priority Queue  
   C. Disjoint Set Union (DSU)  
   D. Stack  
   **Correct Answer: C**
3. In a weighted graph, if you need to find the shortest path between all pairs of vertices, which algorithm would be most suitable?  
   A. Dijkstra's Algorithm  
   B. Prim's Algorithm  
   C. Floyd-Warshall Algorithm  
   D. Breadth-First Search (BFS)  
   **Correct Answer: C**
4. Which algorithm is used to determine if a graph is bipartite?  
   A. Dijkstra's Algorithm  
   B. Bellman-Ford Algorithm  
   C. Breadth-First Search (BFS) or Depth-First Search (DFS)  
   D. Kruskal's Algorithm  
   **Correct Answer: C** (By trying to 2-color the graph. If a conflict arises, it's not bipartite.)
5. What is the concept of "strongly connected components" primarily applicable to?  
   A. Undirected Graphs  
   B. Directed Graphs  
   C. Weighted Graphs  
   D. Bipartite Graphs  
   **Correct Answer: B**
6. If you need to find *all* paths between two nodes in a graph, which traversal algorithm would be the basis for exploring all possibilities?  
   A. Breadth-First Search (BFS)  
   B. Depth-First Search (DFS)  
   C. Dijkstra's Algorithm  
   D. Prim's Algorithm  
   **Correct Answer: B** (DFS is better suited for exploring all paths/branches)
7. What does the "cut" property of a graph refer to in the context of Minimum Spanning Trees?  
   A. A set of edges whose removal disconnects the graph.  
   B. A partition of the vertices into two non-empty sets.  
   C. An edge that is part of a cycle.  
   D. An edge with negative weight.  
   **Correct Answer: B** (The cut property states that if a cut is made in a graph, the minimum weight edge crossing the cut must be part of any MST.)

**Section 4: Sorting & Searching (Advanced)**

1. What is the lower bound for comparison-based sorting algorithms (in terms of time complexity)?  
   A. O(n)  
   B. O(n log n)  
   C. O(n^2)  
   D. O(log n)  
   **Correct Answer: B**
2. Which of the following is true about java.util.Arrays.sort() for an array of primitive ints?  
   A. It uses QuickSort and is stable.  
   B. It uses MergeSort and is unstable.  
   C. It uses a Dual-Pivot QuickSort and is unstable.  
   D. It uses Insertion Sort for small arrays and Merge Sort for larger ones.  
   **Correct Answer: C** (For primitives, Java uses a variant of QuickSort which is not stable. For objects, it uses Timsort which is stable.)
3. Which sorting algorithm is considered "cache-friendly" due to its sequential memory access patterns?  
   A. Quick Sort  
   B. Insertion Sort  
   C. Merge Sort  
   D. Both B and C  
   **Correct Answer: D** (Insertion sort processes elements sequentially, and Merge Sort accesses its sub-arrays and temp array sequentially during the merge step, leading to good cache performance.)
4. What is the average time complexity of searching for an element in a Hash Table (assuming a good hash function and proper load factor)?  
   A. O(1)  
   B. O(log n)  
   C. O(n)  
   D. O(n log n)  
   **Correct Answer: A**
5. What is the fundamental difference between linear search and binary search?  
   A. Linear search works only on sorted arrays, binary search on unsorted.  
   B. Linear search is iterative, binary search is recursive.  
   C. Linear search compares sequentially, binary search halves the search space.  
   D. Linear search uses less memory.  
   **Correct Answer: C**

**Section 5: Dynamic Programming & Greedy Approaches**

1. Which of the following is a key characteristic of problems solvable by Dynamic Programming?  
   A. Always makes the best immediate choice.  
   B. The solution depends on random choices.  
   C. Optimal substructure and overlapping subproblems.  
   D. Involves exploring all possible permutations.  
   **Correct Answer: C**
2. Which of the following problems is *not* typically solved using a greedy approach?  
   A. Activity Selection Problem  
   B. Coin Change Problem (to find minimum number of coins for arbitrary denominations)  
   C. Prim's Algorithm for MST  
   D. Kruskal's Algorithm for MST  
   **Correct Answer: B** (The Coin Change problem with arbitrary denominations is a classic DP problem, not necessarily solvable by a greedy approach for minimum coins.)
3. In Dynamic Programming, what does "tabulation" refer to?  
   A. Storing results of subproblems in a table by starting from the base cases and building up.  
   B. Storing results of subproblems using recursion with memoization.  
   C. Creating a new thread for each subproblem.  
   D. Using a hash map to store intermediate results.  
   **Correct Answer: A**
4. The Longest Common Subsequence (LCS) problem is a classical example of a problem solved using:  
   A. Greedy Algorithm  
   B. Divide and Conquer  
   C. Dynamic Programming  
   D. Backtracking  
   **Correct Answer: C**

**Section 6: Recursion, Stacks, and Queues Advanced**

1. What is tail recursion?  
   A. A recursive call that is the very last operation in the function.  
   B. A recursive function that does not have a base case.  
   C. A recursive function that calls itself multiple times.  
   D. A recursive function that only processes the tail of a list.  
   **Correct Answer: A**
2. Why is tail recursion optimization (TCO) beneficial?  
   A. It makes the code easier to read.  
   B. It reduces the time complexity of the algorithm.  
   C. It allows recursive calls to be optimized to iterative loops, preventing stack overflow.  
   D. It avoids the need for a base case.  
   **Correct Answer: C** (Though Java's JVM does not explicitly perform TCO.)
3. Which of the following tasks would *not* typically use a stack?  
   A. Reversing a string.  
   B. Checking for palindromes (e.g., using two pointers).  
   C. Converting an infix expression to postfix.  
   D. Implementing a depth-first search (DFS) traversal of a graph.  
   **Correct Answer: B** (While a stack *could* be used, the most common and efficient way to check a palindrome is with two pointers, one from each end.)
4. What is the time complexity to insert an element into a java.util.PriorityQueue?  
   A. O(1)  
   B. O(log n)  
   C. O(n)  
   D. O(n log n)  
   **Correct Answer: B**
5. What is the primary characteristic of an "unbounded" queue?  
   A. It has a fixed maximum size.  
   B. It can grow indefinitely as long as memory permits.  
   C. It can only store primitive data types.  
   D. Elements can be added to both ends.  
   **Correct Answer: B**

**Section 7: General DSA Concepts & Problem Solving**

1. What is the concept of "space-time tradeoff" in algorithms?  
   A. Algorithms can only optimize for either space or time, never both.  
   B. Increasing space usage can sometimes reduce time complexity, and vice versa.  
   C. Time and space complexities are always directly proportional.  
   D. It refers to the time taken to manage memory.  
   **Correct Answer: B**
2. An algorithm that uses a fixed amount of extra memory regardless of the input size n is said to have what space complexity?  
   A. O(n)  
   B. O(log n)  
   C. O(1)  
   D. O(n^2)  
   **Correct Answer: C**
3. What does it mean for a problem to be "NP-Complete"?  
   A. It can be solved in polynomial time.  
   B. It can be solved by a non-deterministic Turing machine in polynomial time, and any other NP problem can be reduced to it in polynomial time.  
   C. It cannot be solved by any known algorithm.  
   D. It always requires exponential time complexity.  
   **Correct Answer: B**
4. Which problem-solving approach typically involves systematically searching for a solution by exploring all possible branches and pruning branches that do not satisfy conditions?  
   A. Greedy Approach  
   B. Dynamic Programming  
   C. Backtracking  
   D. Divide and Conquer  
   **Correct Answer: C**
5. What is a "prefix sum array" used for?  
   A. To store elements in sorted order.  
   B. To quickly calculate the sum of elements within any range (subarray) of an array.  
   C. To count the occurrences of each element in an array.  
   D. To find the maximum element in a given array.  
   **Correct Answer: B**
6. How is a try-catch-finally block related to stack operations in Java?  
   A. It has no relation to the call stack.  
   B. It manipulates the heap memory directly.  
   C. When an exception is thrown, the call stack is unwound until a suitable catch block is found.  
   D. It explicitly adds elements to the call stack.  
   **Correct Answer: C**

**Section 8: Data Structure Properties & Comparisons**

1. Which of the following data structures is typically NOT implemented using pointers/references?  
   A. Linked List  
   B. Binary Search Tree  
   C. Hash Table (using chaining)  
   D. Array  
   **Correct Answer: D** (Arrays store elements in contiguous memory locations and are accessed by index, not direct pointers between elements.)
2. What is the primary difference in memory allocation between static and dynamic data structures?  
   A. Static structures allocate memory on the heap, dynamic on the stack.  
   B. Static structures have fixed memory allocation at compile time, dynamic structures can adjust memory at runtime.  
   C. Dynamic structures are faster due to pre-allocated memory.  
   D. Static structures are less prone to memory leaks.  
   **Correct Answer: B**
3. What is the primary reason to use a StringBuilder over String concatenation in a loop in Java?  
   A. StringBuilder is immutable.  
   B. StringBuilder is thread-safe.  
   C. StringBuilder avoids creating many intermediate String objects, improving performance and memory usage.  
   D. StringBuilder allows for regular expression matching.  
   **Correct Answer: C** (String is immutable, so concatenation creates new String objects repeatedly, which is inefficient. StringBuilder is mutable.)
4. Which of the following is true about a "sparse matrix"?  
   A. It is a matrix where all elements are zero.  
   B. It is a matrix where the number of zero elements is much larger than the number of non-zero elements.  
   C. It is a matrix optimized for parallel processing.  
   D. It can only store boolean values.  
   **Correct Answer: B**
5. Which of these is a typical application of a "Min-Max Heap"?  
   A. Efficiently finding both the minimum and maximum elements.  
   B. Implementing a LIFO queue.  
   C. Sorting elements in reverse order.  
   D. Pathfinding in graphs.  
   **Correct Answer: A**
6. What is the main benefit of using a sentinel node in a linked list?  
   A. It makes the linked list thread-safe.  
   B. It simplifies linked list operations (especially edge cases like empty list, insertion at beginning, deletion of head).  
   C. It reduces the memory footprint of the linked list.  
   D. It allows for O(1) random access.  
   **Correct Answer: B**
7. What is the primary difference between a "tree" and a "graph" data structure?  
   A. Trees have cycles, graphs do not.  
   B. Graphs are always connected, trees are not.  
   C. Trees are a specialized type of graph that is acyclic and connected, with a designated root.  
   D. Trees can only have a maximum of two children per node, graphs can have any number.  
   **Correct Answer: C**
8. When working with graph algorithms in Java, which java.util class is often used to implement a priority queue for algorithms like Dijkstra's or Prim's?  
   A. ArrayDeque  
   B. LinkedList  
   C. PriorityQueue  
   D. Stack  
   **Correct Answer: C`**
9. What is a "hash collision" in a hash table?  
   A. When two different keys produce the same hash code.  
   B. When two identical keys produce different hash codes.  
   C. When the hash table becomes completely full.  
   D. When a key is not found in the table.  
   **Correct Answer: A** (This is a slightly different phrasing of a previous question on hash function properties, but focuses on the *event* itself rather than the function's ability to produce it.)
10. In Java, which keyword is typically used to indicate that a variable will not be reassigned after its initial assignment, making it suitable for use as a key in a HashMap (in conjunction with hashCode() and equals())?  
    A. static  
    B. final  
    C. volatile  
    D. transient  
    **Correct Answer: B**

**Section 1: Java Collections & Interfaces - Advanced**

1. Which java.util.Collection interface ensures that elements are sorted according to their natural ordering or by a Comparator, and allows efficient retrieval of elements within a range?  
   A. java.util.List  
   B. java.util.Queue  
   C. java.util.Set  
   D. java.util.SortedSet  
   **Correct Answer: D**
2. What is the purpose of the Comparable interface in Java when used with collections?  
   A. To allow objects to be sorted based on a specific custom ordering.  
   B. To allow objects to be compared for equality using equals().  
   C. To provide a default, natural ordering for objects of a class.  
   D. To enable iteration over a collection.  
   **Correct Answer: C**
3. When should you use a java.util.ConcurrentHashMap instead of a java.util.HashMap?  
   A. When you need faster single-threaded performance.  
   B. When you need guaranteed iteration order.  
   C. When multiple threads might access and modify the map concurrently.  
   D. When you want to store null keys or values.  
   **Correct Answer: C**
4. Which method is specific to java.util.Queue (and its sub-interfaces like Deque and PriorityQueue) for removing and returning the head of the queue, returning null if the queue is empty?  
   A. remove()  
   B. poll()  
   C. pop()  
   D. get()  
   **Correct Answer: B**
5. What happens if you attempt to add an element that is not Comparable (and no Comparator is provided) to a java.util.TreeSet?  
   A. It will compile successfully but throw a NullPointerException at runtime.  
   B. It will compile successfully but throw a ClassCastException at runtime.  
   C. It will cause a compile-time error.  
   D. The element will be added, but its position will be arbitrary.  
   **Correct Answer: B**
6. Consider List<String> list = new ArrayList<>();. What is the time complexity of list.add(0, "element"); (inserting at the beginning) for an ArrayList containing N elements?  
   A. O(1)  
   B. O(log N)  
   C. O(N)  
   D. O(N log N)  
   **Correct Answer: C**

**Section 2: Advanced Linked List Scenarios**

1. In a singly linked list with a head pointer, what is the best way to efficiently append an element to the end of the list with O(1) complexity?  
   A. Always iterate from the head to the end.  
   B. Maintain an additional tail pointer that points to the last node.  
   C. Use a circular linked list.  
   D. Convert it to an array first.  
   **Correct Answer: B**
2. What is the time complexity for reversing a singly linked list iteratively?  
   A. O(1)  
   B. O(log N)  
   C. O(N)  
   D. O(N log N)  
   **Correct Answer: C**
3. What is a common problem associated with improper memory management of linked lists in languages without automatic garbage collection (like C/C++)?  
   A. Stack overflow  
   B. Array index out of bounds  
   C. Memory leaks (unfreed nodes)  
   D. Type mismatch errors  
   **Correct Answer: C**

**Section 3: Tree Traversals & Properties - Advanced**

1. What is the characteristic feature of a "degenerate tree"?  
   A. All nodes have only one child, forming a linked list.  
   B. All leaf nodes are at the same level.  
   C. It's a balanced binary search tree.  
   D. It has no root node.  
   **Correct Answer: A**
2. Which traversal method for a binary tree would be most suitable to replicate the directory structure of a file system (listing folders then files within them)?  
   A. In-order traversal  
   B. Pre-order traversal  
   C. Post-order traversal  
   D. Level-order traversal  
   **Correct Answer: B** (Pre-order visits the root first, then children, which mirrors listing a directory then its contents recursively.)
3. The height of an empty binary tree is generally defined as:  
   A. 0  
   B. -1  
   C. 1  
   D. Undefined  
   **Correct Answer: B** (Common convention in algorithms to simplify height calculations. A single node tree has height 0.)
4. Which of the following statements about a min-heap is false?  
   A. The root node always contains the smallest element.  
   B. Every node's value is less than or equal to the values of its children.  
   C. It is always a complete binary tree.  
   D. The largest element can be found in O(1) time.  
   **Correct Answer: D** (Finding the largest element requires traversing potentially O(n) nodes.)
5. What is a "Threaded Binary Tree"?  
   A. A binary tree where nodes store pointers to their parent nodes.  
   B. A binary tree used for multi-threaded processing.  
   C. A binary tree where null right/left pointers are replaced by pointers to in-order successor/predecessor nodes.  
   D. A binary tree implemented using a linked list.  
   **Correct Answer: C**
6. What is the advantage of an iterative tree traversal over a recursive one (ignoring TCO)?  
   A. Simpler code structure.  
   B. Guarantees better time complexity.  
   C. Avoids potential StackOverflowError for deep trees.  
   D. Uses less memory overall.  
   **Correct Answer: C**

**Section 4: Specialized Tree Structures**

1. A "Fenwick Tree" (or Binary Indexed Tree) is primarily used for:  
   A. Managing connected components in a graph.  
   B. Efficiently performing prefix sum queries and single point updates on an array.  
   C. Implementing a symbol table.  
   D. Balancing binary search trees.  
   **Correct Answer: B**
2. What is the time complexity for a query operation (e.g., range sum) on a Segment Tree?  
   A. O(1)  
   B. O(log N)  
   C. O(N)  
   D. O(N log N)  
   **Correct Answer: B**
3. In a B-Tree of order m, what is the minimum number of keys a non-root internal node can have?  
   A. m-1  
   B. m/2 - 1 (ceiling of m/2 - 1)  
   C. ceil(m/2) - 1  
   D. 1  
   **Correct Answer: C**
4. Which data structure is best for storing a dictionary of words to quickly check if a word is a valid prefix of another word?  
   A. Hash Map  
   B. Binary Search Tree  
   C. Trie (Prefix Tree)  
   D. Suffix Tree  
   **Correct Answer: C**
5. What is a "Scapegoat Tree"?  
   A. A self-balancing binary search tree that uses a simple rebalancing strategy only when the tree becomes significantly unbalanced.  
   B. A type of heap that prioritizes the most frequently accessed elements.  
   C. A tree used to represent hierarchical data with no specific ordering.  
   D. A tree designed for parallel processing of search queries.  
   **Correct Answer: A**

**Section 5: Graph Algorithms - Advanced**

1. What does it mean for a graph to be "dense"?  
   A. The number of edges is close to the maximum possible (E ≈ V^2).  
   B. The graph is heavily connected, with many cycles.  
   C. It has many isolated vertices.  
   D. It has a small number of vertices.  
   **Correct Answer: A**
2. For a dense graph, which representation is generally more space-efficient?  
   A. Adjacency Matrix  
   B. Adjacency List  
   C. Incidence Matrix  
   D. Edge List  
   **Correct Answer: A** (Adjacency Matrix uses V^2 space, which becomes beneficial over Adjacency List (V+E) when E is close to V^2).
3. Which algorithm would you use to find the maximum flow in a flow network?  
   A. Dijkstra's Algorithm  
   B. Ford-Fulkerson Algorithm  
   C. Kruskal's Algorithm  
   D. Topological Sort  
   **Correct Answer: B**
4. A graph where all vertices have an even degree is guaranteed to have:  
   A. A Hamiltonian Cycle  
   B. An Eulerian Circuit  
   C. No cycles  
   D. Only one connected component  
   **Correct Answer: B** (This is Euler's theorem for Eulerian circuits.)
5. What is the primary purpose of the "Kosaraju's Algorithm" or "Tarjan's Algorithm"?  
   A. To find the shortest path between two vertices.  
   B. To detect cycles in an undirected graph.  
   C. To find all strongly connected components (SCCs) in a directed graph.  
   D. To compute the minimum spanning tree.  
   **Correct Answer: C**
6. Which graph traversal technique is best for finding connected components in an undirected graph?  
   A. Dijkstra's  
   B. Both BFS and DFS  
   C. Topological Sort  
   D. Prim's  
   **Correct Answer: B**
7. What is an "articulation point" (or cut vertex) in a graph?  
   A. A vertex with degree 1.  
   B. A vertex whose removal increases the number of connected components.  
   C. A vertex that is part of every cycle.  
   D. A vertex that has no outgoing edges.  
   **Correct Answer: B**

**Section 6: Hashing & Hash Table Design - Advanced**

1. In Java, what is the default initial capacity of a java.util.HashMap?  
   A. 8  
   B. 16  
   C. 32  
   D. 64  
   **Correct Answer: B**
2. What is the default load factor for a java.util.HashMap in Java?  
   A. 0.5  
   B. 0.75  
   C. 0.9  
   D. 1.0  
   **Correct Answer: B**
3. If a hash table implementation uses linear probing for collision resolution, and many collisions occur, what phenomenon can significantly degrade performance?  
   A. Secondary clustering  
   B. Primary clustering  
   C. Avalanche effect  
   D. Key diffusion  
   **Correct Answer: B**
4. What is the main idea behind "Universal Hashing"?  
   A. To use a single, fixed hash function for all applications.  
   B. To choose a hash function randomly from a family of hash functions at runtime to minimize collisions in expectation.  
   C. To guarantee no collisions will ever occur.  
   D. To create a hash function that is cryptographically secure.  
   **Correct Answer: B**

**Section 7: Algorithm Analysis & Design - Advanced**

1. If an algorithm has a time complexity of O(n log n) and another has O(n√n), which one is asymptotically faster for large n?  
   A. O(n log n)  
   B. O(n√n)  
   C. They are equivalent.  
   D. Cannot be determined.  
   **Correct Answer: A** (log n grows slower than √n, so n log n is faster than n√n.)
2. What does it mean for an algorithm to be "in-place"?  
   A. It sorts the data without using any extra memory.  
   B. It uses a constant amount of extra memory (O(1)) beyond the input storage.  
   C. It modifies the input data directly without creating a copy.  
   D. It runs faster than its out-of-place counterpart.  
   **Correct Answer: B** (It implies O(1) auxiliary space, or at most O(log n) for recursion stack). Option C is also generally true for in-place but B is the more precise definition of space complexity. Given the options, B is the most technically correct definition of *space complexity* for "in-place".
3. Which of the following is true about algorithms classified as "NP-Hard"?  
   A. They can be solved in polynomial time.  
   B. They are problems to which all NP problems can be reduced in polynomial time.  
   C. They are always more difficult than NP-Complete problems.  
   D. They are guaranteed to have a polynomial-time solution.  
   **Correct Answer: B** (NP-Hard means at least as hard as any NP-Complete problem. It may or may not be in NP itself.)
4. The concept of "optimal substructure" in Dynamic Programming implies that:  
   A. Subproblems are independent of each other.  
   B. The optimal solution to a problem can be constructed from the optimal solutions of its subproblems.  
   C. The algorithm uses a greedy approach.  
   D. The problem can be divided into equal halves.  
   **Correct Answer: B**
5. Which of the following is an example of a "divide and conquer" algorithm that is *not* a sorting algorithm?  
   A. Linear Search  
   B. Fibonacci Sequence (recursive standard)  
   C. Binary Search  
   D. Bubble Sort  
   **Correct Answer: C**
6. Which asymptotic notation is used to provide both an upper and lower bound, indicating a tight bound on the running time?  
   A. Big-O (O)  
   B. Big-Omega (Ω)  
   C. Big-Theta (Θ)  
   D. Little-o (o)  
   **Correct Answer: C**
7. What is the typical way to handle Integer.MAX\_VALUE or Integer.MIN\_VALUE during calculations in Java DSA problems to avoid overflow/underflow, especially in shortest path algorithms?  
   A. Use long data type for sums.  
   B. Use BigInteger for all calculations.  
   C. Implement custom arithmetic logic.  
   D. Assume int range is sufficient.  
   **Correct Answer: A**

**Section 8: Miscellaneous DSA Concepts & Java Specifics**

1. What is a "deque" (double-ended queue) often used for in algorithms?  
   A. Implementing a priority queue.  
   B. Storing elements in sorted order.  
   C. Breadth-First Search (BFS) for 0-1 weighted graphs (0-1 BFS).  
   D. Recursion stack management.  
   **Correct Answer: C** (0-1 BFS uses a deque to prioritize 0-weight edges to the front and 1-weight edges to the back.)
2. What is a "memory-mapped file" used for in the context of large data sets, potentially relevant to external sorting?  
   A. Loading the entire file into RAM at once.  
   B. Treating a file on disk as if it were an array in main memory.  
   C. Encrypting file contents for secure storage.  
   D. Compressing files to save disk space.  
   **Correct Answer: B**
3. In Java, how can you define a custom sorting order for objects in a Collections.sort() method if the class does not implement Comparable or you need a different order?  
   A. Implement Serializable.  
   B. Pass a Comparator object.  
   C. Use a HashSet.  
   D. Call Arrays.sort() instead.  
   **Correct Answer: B**
4. What is a "min-cut" in a graph?  
   A. The minimum number of edges to remove to disconnect two specified vertices (s-t cut).  
   B. The minimum number of vertices to remove to make the graph acyclic.  
   C. A path with the minimum sum of edge weights.  
   D. An edge that has the smallest weight in the graph.  
   **Correct Answer: A**
5. Which of the following data structures is typically used to implement a Least Recently Used (LRU) cache?  
   A. Queue and Array  
   B. Stack and Hash Table  
   C. Doubly Linked List and Hash Table  
   D. Binary Search Tree and Queue  
   **Correct Answer: C** (Doubly linked list maintains access order, hash table provides O(1) lookup.)
6. What is "prefix matching" in the context of data structures, and which structure excels at it?  
   A. Finding exact matches in a hash table.  
   B. Finding elements that start with a specific sequence of characters, usually with a Trie.  
   C. Locating the smallest element in a min-heap.  
   D. Identifying cycles in a graph.  
   **Correct Answer: B**
7. What is the significance of the hashCode() method returning a consistent value for an object's lifetime if it's used as a key in a hash-based collection?  
   A. It ensures the object is immutable.  
   B. It prevents StackOverflowError.  
   C. It guarantees that the object can be reliably retrieved from the hash collection.  
   D. It makes the object thread-safe.  
   **Correct Answer: C**
8. When working with graph algorithms in Java, what is a common way to represent infinite distance (for unvisited nodes) in a distance array for algorithms like Dijkstra's?  
   A. Integer.MIN\_VALUE  
   B. 0  
   C. Integer.MAX\_VALUE  
   D. null  
   **Correct Answer: C** (Large positive value ensures any finite path will be smaller.)
9. What is the fundamental difference between pass-by-value and pass-by-reference in the context of passing complex data structures (like arrays or objects) to methods in Java?  
   A. Java uses pass-by-reference for primitive types and pass-by-value for objects.  
   B. Java uses pass-by-value for all arguments, but for objects, the *reference* itself is passed by value.  
   C. Java explicitly supports both pass-by-value and pass-by-reference.  
   D. Java uses pass-by-reference for all arguments.  
   **Correct Answer: B** (This is a common misconception. Java is *always* pass-by-value. For objects, the *value* of the reference (memory address) is passed, allowing the called method to modify the object pointed to by that reference.)
10. What is a "Topological Sort" of a Directed Acyclic Graph (DAG)?  
    A. A path that visits every vertex exactly once.  
    B. A linear ordering of its vertices such that for every directed edge (u, v), vertex u comes before v in the ordering.  
    C. A cycle that includes all vertices.  
    D. A shortest path from a source to a destination.  
    **Correct Answer: B**
11. Which data structure would you use to implement a compiler's symbol table?  
    A. Stack  
    B. Queue  
    C. Hash Table or Binary Search Tree  
    D. Linked List  
    **Correct Answer: C** (Symbol tables require efficient key-value lookups, which hash tables or balanced BSTs provide.)
12. In Java, what is the significance of using transient keyword for a field in a class when dealing with data structures that might be serialized?  
    A. It makes the field immutable.  
    B. It ensures the field is thread-safe.  
    C. It excludes the field from being serialized when the object is written to a persistent storage.  
    D. It indicates the field is a static member.  
    **Correct Answer: C**

**Section 1: Java Collections & Internal Mechanics**

1. In Java, when you pass an object (e.g., an ArrayList) to a method, what is actually passed?  
   A. A copy of the object itself.  
   B. The memory address of the object by reference.  
   C. The value of the object's reference (a copy of the memory address).  
   D. A deep clone of the object.  
   **Correct Answer: C**
2. If an ArrayList's internal array reaches its capacity limit and a new element is added, what typically happens internally in Java's ArrayList?  
   A. An OutOfMemoryError is thrown immediately.  
   B. The array is truncated to fit the new element.  
   C. A new, larger array is allocated, and all elements are copied to it.  
   D. The add() operation becomes O(N) but no resizing occurs.  
   **Correct Answer: C**
3. What is the main distinction between java.util.Vector and java.util.ArrayList regarding thread safety?  
   A. Vector is not thread-safe, ArrayList is.  
   B. Vector is synchronized (thread-safe), ArrayList is not.  
   C. Both are thread-safe and use internal locks.  
   D. Both are non-thread-safe.  
   **Correct Answer: B**
4. Which method in java.util.Collection is used to remove all elements from the collection?  
   A. delete()  
   B. remove()  
   C. clear()  
   D. empty()  
   **Correct Answer: C**
5. What is the significance of the null element in a java.util.HashMap's key set?  
   A. HashMap cannot store null keys.  
   B. HashMap can store at most one null key.  
   C. HashMap can store multiple null keys mapped to different values.  
   D. null keys are only allowed if values are also null.  
   **Correct Answer: B**
6. Which java.util.concurrent collection provides a blocking queue implementation useful for producer-consumer scenarios?  
   A. ConcurrentHashMap  
   B. CopyOnWriteArrayList  
   C. LinkedBlockingQueue  
   D. ConcurrentLinkedDeque  
   **Correct Answer: C**

**Section 2: Advanced Linked List Operations & Types**

1. What is the time complexity of finding the Nth node from the end of a singly linked list in a single pass?  
   A. O(1)  
   B. O(log N)  
   C. O(N)  
   D. O(N^2)  
   **Correct Answer: C** (Requires two pointers: one moves N steps ahead, then both move until the first reaches the end.)
2. What is a "dummy head node" (or sentinel node) in a linked list often used for?  
   A. To store metadata about the list.  
   B. To simplify operations by avoiding special handling for an empty list or operations at the beginning.  
   C. To mark the end of the list.  
   D. To enhance cache locality.  
   **Correct Answer: B**
3. Which problem is efficiently solvable using Floyd's Cycle-Finding Algorithm (Tortoise and Hare)?  
   A. Finding the middle element of a linked list.  
   B. Detecting a cycle in a linked list.  
   C. Reversing a linked list.  
   D. Merging two sorted linked lists.  
   **Correct Answer: B**
4. What is a "XOR Linked List"?  
   A. A linked list where nodes store the XOR sum of the previous and next node's addresses.  
   B. A linked list used for bitwise operations.  
   C. A type of cryptographically secure linked list.  
   D. A linked list that stores only boolean values.  
   **Correct Answer: A**

**Section 3: Tree Algorithms & Structures**

1. What is a "monotonic stack" commonly used for?  
   A. Storing elements in strictly increasing or decreasing order.  
   B. Efficiently finding the next greater/smaller element for each element in an array.  
   C. Balancing binary search trees.  
   D. Implementing a priority queue.  
   **Correct Answer: B**
2. In a min-heap, if you want to decrease the priority of an element, what operation is required to restore the heap property?  
   A. Heapify-down (percolate down)  
   B. Heapify-up (percolate up)  
   C. Rotation  
   D. No operation is required.  
   **Correct Answer: A**
3. What is the time complexity of constructing a binary heap from an unsorted array of N elements?  
   A. O(1)  
   B. O(log N)  
   C. O(N)  
   D. O(N log N)  
   **Correct Answer: C**
4. What is a "self-balancing binary search tree" designed to prevent?  
   A. StackOverflowError in recursive traversals.  
   B. Degradation of search/insertion/deletion to O(N) complexity in worst-case scenarios.  
   C. Duplication of elements.  
   D. High memory consumption.  
   **Correct Answer: B**
5. What is a "Ternary Search Tree" (TST) primarily optimized for?  
   A. Range queries on numerical data.  
   B. Storing and searching strings with shared prefixes, often more space-efficient than Tries for sparse data.  
   C. Balancing complex tree structures.  
   D. Efficiently finding the median element.  
   **Correct Answer: B**
6. What is the role of a "parent pointer" in tree nodes?  
   A. To allow fast traversal to any child.  
   B. To make it a self-balancing tree.  
   C. To simplify operations like deletion or finding ancestors without recursion/stack.  
   D. To make the tree a min-heap.  
   **Correct Answer: C**

**Section 4: Advanced Graph Algorithms & Properties**

1. What is the "Cycle Detection" problem for a directed graph, and which algorithm is commonly used?  
   A. Finding all Hamiltonian cycles using BFS.  
   B. Finding simple cycles using DFS.  
   C. Finding minimum weight cycles using Dijkstra's.  
   D. Detecting negative cycles using Bellman-Ford.  
   **Correct Answer: B** (While Bellman-Ford detects negative cycles, DFS is a general method for detecting any cycle in a directed graph by checking for back edges to an ancestor in the DFS tree.)
2. What is a "bridge" in an undirected graph?  
   A. An edge that connects two disconnected components.  
   B. An edge whose removal increases the number of connected components.  
   C. An edge with maximum weight.  
   D. An edge that forms a cycle.  
   **Correct Answer: B**
3. Which algorithm computes all-pairs shortest paths in a graph using dynamic programming, handling negative edge weights but not negative cycles?  
   A. Dijkstra's Algorithm  
   B. Bellman-Ford Algorithm  
   C. Floyd-Warshall Algorithm  
   D. A\* Search  
   **Correct Answer: C**
4. What is a common application of the "Bipartite Graph" concept?  
   A. Modeling task dependencies for scheduling.  
   B. Resource allocation and matching problems (e.g., stable marriage).  
   C. Finding the shortest path in road networks.  
   D. Detecting network intrusions.  
   **Correct Answer: B**
5. What does the "Min-Cut Max-Flow Theorem" state?  
   A. The maximum flow through a network is equal to the capacity of a minimum cut.  
   B. The minimum cut in a graph is always equal to the total number of edges.  
   C. Maximum flow can only be achieved in acyclic graphs.  
   D. Minimum cut is found using a greedy algorithm.  
   **Correct Answer: A**
6. What is a "Hamiltonian Path" in a graph?  
   A. A path that visits every vertex exactly once.  
   B. A path that visits every edge exactly once.  
   C. A path with the minimum total weight.  
   D. A path that forms a cycle.  
   **Correct Answer: A**
7. Which data structure is efficient for representing a very sparse graph, especially if vertices are not numbered sequentially (e.g., their IDs are strings)?  
   A. Adjacency Matrix  
   B. Adjacency List using HashMap<Vertex, List<Vertex>>  
   C. 2D array  
   D. Stack and Queue  
   **Correct Answer: B**

**Section 5: Sorting & Searching - Advanced**

1. Which sorting algorithm is best known for its use of a "pivot" element and partitioning?  
   A. Merge Sort  
   B. Quick Sort  
   C. Heap Sort  
   D. Bubble Sort  
   **Correct Answer: B**
2. What is "Interpolation Search" best suited for?  
   A. Unsorted arrays.  
   B. Arrays with duplicate elements.  
   C. Large, uniformly distributed sorted arrays.  
   D. Linked lists.  
   **Correct Answer: C** (It's an improvement over binary search for uniformly distributed data, as it estimates the position of the target value.)
3. Which sorting algorithm is considered stable and performs well on small datasets but is inefficient for large ones?  
   A. Merge Sort  
   B. Quick Sort  
   C. Insertion Sort  
   D. Selection Sort  
   **Correct Answer: C**
4. What is "Quickselect" algorithm used for?  
   A. Finding the shortest path in a graph.  
   B. Finding the k-th smallest (or largest) element in an unsorted array in expected O(N) time.  
   C. Sorting an array in O(N log N) time.  
   D. Detecting cycles in a linked list.  
   **Correct Answer: B**
5. When comparing two objects in Java for sorting, which return value of a compareTo() or compare() method indicates that the first object is "greater than" the second?  
   A. 0  
   B. A negative integer  
   C. A positive integer  
   D. true  
   **Correct Answer: C**

**Section 6: Hashing & Bloom Filters**

1. What is a "Bloom Filter"?  
   A. A data structure that stores key-value pairs with O(1) average access.  
   B. A probabilistic data structure that checks for set membership with possible false positives but no false negatives.  
   C. A filter used to remove duplicate elements from a list.  
   D. A cryptographic hash function generator.  
   **Correct Answer: B**
2. Which of the following is true about "false positives" in a Bloom Filter?  
   A. They never occur.  
   B. They occur when an element is reported as not being in the set, but it actually is.  
   C. They occur when an element is reported as being in the set, but it actually isn't.  
   D. They are caused by bad hash functions.  
   **Correct Answer: C**
3. What is the typical solution for "secondary clustering" in hash tables using open addressing?  
   A. Chaining  
   B. Linear Probing  
   C. Quadratic Probing or Double Hashing  
   D. Increasing the load factor  
   **Correct Answer: C** (Linear probing causes primary clustering. Quadratic probing and double hashing attempt to mitigate it, though quadratic probing can suffer from secondary clustering.)
4. If a Java String is used as a key in a HashMap, why is it suitable despite its mutable content (from a character array perspective)?  
   A. String objects are inherently thread-safe.  
   B. String objects are immutable in Java, so their hashCode() is constant.  
   C. String objects are always unique.  
   D. HashMap uses a special hash function for strings.  
   **Correct Answer: B**

**Section 7: Algorithm Design Paradigms & Analysis**

1. What is the primary characteristic of an "approximation algorithm"?  
   A. It always finds the exact optimal solution in polynomial time.  
   B. It finds a near-optimal solution for NP-hard problems within a guaranteed bound in polynomial time.  
   C. It uses random numbers to find a solution.  
   D. It sorts data before processing.  
   **Correct Answer: B**
2. Which of the following problems is an example of an "NP-Complete" problem?  
   A. Sorting  
   B. Shortest Path (Dijkstra's)  
   C. Traveling Salesperson Problem (TSP)  
   D. Searching in a hash table  
   **Correct Answer: C**
3. What is the "Amortized Analysis" technique typically used for?  
   A. To determine the worst-case runtime of a single operation.  
   B. To determine the average runtime of a single operation over a long sequence of operations.  
   C. To prove the correctness of a greedy algorithm.  
   D. To find the lower bound of an algorithm.  
   **Correct Answer: B**
4. An algorithm that solves a problem by breaking it down into smaller, independent subproblems, solving them, and then combining their solutions, is using which paradigm?  
   A. Greedy Approach  
   B. Dynamic Programming  
   C. Divide and Conquer  
   D. Backtracking  
   **Correct Answer: C**
5. If an algorithm takes T(N) = 3T(N/3) + O(1) time, what is its complexity using the Master Theorem?  
   A. O(N)  
   B. O(log N)  
   C. O(N log N)  
   D. O(N^2)  
   **Correct Answer: A** (Case 1: a=3, b=3, f(N)=O(1). N^(log\_b a) = N^(log\_3 3) = N^1. Since f(N) is O(N^0) and log\_b a = 1, it's O(N)).
6. What is the main idea behind "Randomized Algorithms"?  
   A. They always produce the correct output in expected polynomial time.  
   B. They use random numbers to make choices, potentially leading to better average-case performance or simpler algorithms.  
   C. They guarantee a solution for NP-Hard problems.  
   D. They are used exclusively for cryptography.  
   **Correct Answer: B**

**Section 8: Miscellaneous DSA Concepts & Java Specifics**

1. What is a "circular buffer" (or circular array) often used for?  
   A. Implementing a self-balancing tree.  
   B. Efficiently creating a fixed-size queue that reuses space.  
   C. Storing elements in random order.  
   D. Managing linked list nodes.  
   **Correct Answer: B**
2. How would you represent a polynomial (e.g., 3x^2 + 2x + 5) using a data structure?  
   A. A queue to store coefficients.  
   B. A stack to store terms.  
   C. A linked list or array of terms (coefficient, exponent pairs).  
   D. A binary search tree.  
   **Correct Answer: C**
3. What is the purpose of the volatile keyword in Java in the context of multithreaded DSA operations?  
   A. To make a variable immutable.  
   B. To guarantee that writes to the variable are visible to other threads, preventing caching issues.  
   C. To make a method synchronized.  
   D. To declare a constant.  
   **Correct Answer: B**
4. What is a "Skip List"?  
   A. A type of linked list that skips nodes to improve traversal speed.  
   B. A probabilistic data structure that uses multiple levels of linked lists to achieve O(log n) average time complexity for search, insertion, and deletion.  
   C. A specialized tree structure for skipping large subtrees.  
   D. A data structure used for skipping over duplicates in a sorted list.  
   **Correct Answer: B**
5. Which of the following is true about String vs char[] in Java for storing character sequences?  
   A. String is mutable, char[] is immutable.  
   B. String is immutable, char[] is mutable.  
   C. Both are immutable.  
   D. Both are mutable.  
   **Correct Answer: B**
6. What is the primary characteristic of an "abstract class" in Java, relevant to defining ADTs?  
   A. It can be instantiated directly.  
   B. It must contain at least one abstract method.  
   C. It cannot define concrete methods.  
   D. It serves as a blueprint for other classes, possibly with some method implementations, but cannot be instantiated itself.  
   **Correct Answer: D**
7. If you are comparing two objects using their natural ordering, which method should you typically implement?  
   A. equals()  
   B. hashCode()  
   C. compareTo() from Comparable  
   D. compare() from Comparator  
   **Correct Answer: C**
8. What is the main advantage of using a BlockingQueue over a regular Queue in a multi-threaded producer-consumer scenario?  
   A. It's faster for single-threaded use.  
   B. It automatically handles synchronization and thread coordination (blocking/waiting) when the queue is full or empty.  
   C. It uses less memory.  
   D. It allows random access to elements.  
   **Correct Answer: B**
9. What is the fundamental concept behind "Recursion" in algorithms?  
   A. Iterating through a collection.  
   B. A function calling itself to solve smaller instances of the same problem.  
   C. Using a loop to repeatedly execute a block of code.  
   D. Dividing a problem into independent subproblems.  
   **Correct Answer: B**
10. When might a StackOverflowError occur in a Java program implementing a recursive algorithm?  
    A. When the heap memory is exhausted.  
    B. When a loop runs infinitely.  
    C. When the recursion depth exceeds the JVM's maximum stack size.  
    D. When an array index is out of bounds.  
    **Correct Answer: C**
11. What is a "sparse array" and how is it typically optimized for storage?  
    A. An array with all elements being zero; stored as a list of zeros.  
    B. An array with very few non-zero elements; stored using specialized data structures (e.g., hash map or linked list of (index, value) pairs) to save space.  
    C. A multi-dimensional array; stored as a single flat array.  
    D. An array that can grow or shrink dynamically; uses ArrayList.  
    **Correct Answer: B**
12. What is "lazy deletion" in data structures like hash tables or binary search trees?  
    A. Deleting an element immediately and physically removing it from memory.  
    B. Marking an element as deleted without physically removing it, to simplify operations or avoid restructuring.  
    C. Delaying deletion until the entire data structure is rebuilt.  
    D. Only deleting elements when the garbage collector runs.  
    **Correct Answer: B**

**Analysis of an Algorithm & Asymptotic Analysis**

1. Which asymptotic notation describes the **upper bound** of an algorithm's running time, typically representing the worst-case complexity?  
   A) Big Omega (Ω)  
   B) Big Theta (Θ)  
   C) Big O (O)  
   D) Little o (o)  
   **Answer: C) Big O (O)**
2. Which asymptotic notation describes both the **upper and lower bounds** of an algorithm's running time, representing its tight bound or average-case complexity when bounds are identical?  
   A) Big O (O)  
   B) Big Omega (Ω)  
   C) Big Theta (Θ)  
   D) Big Epsilon (Ε)  
   **Answer: C) Big Theta (Θ)**
3. An algorithm with a time complexity of O(log n) means its running time:  
   A) Grows linearly with the input size.  
   B) Grows quadratically with the input size.  
   C) Halves the problem size in each step.  
   D) Remains constant regardless of input size.  
   **Answer: C) Halves the problem size in each step.**
4. The primary purpose of **Algorithm Analysis** is to:  
   A) Write code that is easy to understand.  
   B) Determine the exact execution time on a specific machine.  
   C) Estimate the resources (time and space) an algorithm requires.  
   D) Debug the algorithm for logical errors.  
   **Answer: C) Estimate the resources (time and space) an algorithm requires.**

**Analysis of Different Types of Algorithms**

1. Which algorithmic paradigm breaks a problem into smaller subproblems, solves them recursively, and then combines their solutions?  
   A) Greedy Algorithm  
   B) Dynamic Programming  
   C) Divide and Conquer  
   D) Brute Force  
   **Answer: C) Divide and Conquer**
2. An algorithm that makes the locally optimal choice at each step with the hope of finding a globally optimal solution is known as a:  
   A) Dynamic Programming Algorithm  
   B) Greedy Algorithm  
   C) Backtracking Algorithm  
   D) Branch-and-Bound Algorithm  
   **Answer: B) Greedy Algorithm**
3. Which algorithmic technique solves complex problems by breaking them into overlapping subproblems and storing their results to avoid recomputing?  
   A) Divide and Conquer  
   B) Greedy Algorithm  
   C) Dynamic Programming  
   D) Brute Force  
   **Answer: C) Dynamic Programming**
4. Which algorithm systematically explores all possible paths to build a solution incrementally, "backtracking" when a partial solution leads to a dead end?  
   A) Dynamic Programming  
   B) Greedy Algorithm  
   C) Backtracking Algorithm  
   D) Brute Force  
   **Answer: C) Backtracking Algorithm**
5. Merge Sort is an example of which algorithmic paradigm, typically leading to O(n log n) complexity?  
   A) Greedy Algorithm  
   B) Dynamic Programming  
   C) Divide and Conquer  
   D) Brute Force  
   **Answer: C) Divide and Conquer**
6. Which algorithmic technique is primarily used for optimization problems, exploring a search space but pruning branches that cannot lead to a better solution using bounds?  
   A) Backtracking  
   B) Brute Force  
   C) Branch-and-Bound  
   D) Stochastic Algorithm  
   **Answer: C) Branch-and-Bound**
7. Algorithms that use randomness as part of their logic, such as Monte Carlo methods or Genetic Algorithms, are called:  
   A) Deterministic Algorithms  
   B) Greedy Algorithms  
   C) Stochastic Algorithms  
   D) Exhaustive Search Algorithms  
   **Answer: C) Stochastic Algorithms**

**Complexity & Application of Data Structures**

1. In the context of space complexity, **auxiliary space** refers to:  
   A) The memory used to store the input data.  
   B) The temporary space used during algorithm execution (e.g., stack space for recursion).  
   C) The memory required for the output data.  
   D) The total memory available in the system.  
   **Answer: B) The temporary space used during algorithm execution (e.g., stack space for recursion).**
2. Why are data structures important in algorithm design?  
   A) They make algorithms harder to understand.  
   B) They are only useful for very large datasets.  
   C) Choosing the right data structure can drastically improve an algorithm's time and space performance.  
   D) They allow algorithms to run on any computer.  
   **Answer: C) Choosing the right data structure can drastically improve an algorithm's time and space performance.**

**Introductory Concepts & Basic Data Structures**

1. An Abstract Data Type (ADT) primarily defines:  
   A) How data is stored and operations are implemented.  
   B) The data and the set of operations that can be performed on it, without implementation details.  
   C) The specific memory addresses used by data.  
   D) The efficiency of data manipulation.  
   **Answer: B) The data and the set of operations that can be performed on it, without implementation details.**
2. In Java, which construct is typically used to define an Abstract Data Type (ADT)?  
   A) Class  
   B) Object  
   C) Interface  
   D) Abstract Class  
   **Answer: C) Interface**
3. What is the time complexity for accessing an element in an array by its index?  
   A) O(n)  
   B) O(log n)  
   C) O(1)  
   D) O(n^2)  
   **Answer: C) O(1)**
4. What is the worst-case time complexity for inserting an element into the middle of a traditional array?  
   A) O(1)  
   B) O(log n)  
   C) O(n)  
   D) O(n^2)  
   **Answer: C) O(n)**
5. Which data structure follows the Last-In, First-Out (LIFO) principle?  
   A) Queue  
   B) Linked List  
   C) Stack  
   D) Array  
   **Answer: C) Stack**
6. Which data structure follows the First-In, First-Out (FIFO) principle?  
   A) Stack  
   B) Queue  
   C) Hash Table  
   D) Tree  
   **Answer: B) Queue**
7. The operation to add an element to the top of a stack is called:  
   A) Enqueue  
   B) Pop  
   C) Push  
   D) Peek  
   **Answer: C) Push**
8. Which type of queue allows elements to be dequeued based on their assigned priority, not necessarily their insertion order?  
   A) Circular Queue  
   B) Deque  
   C) Priority Queue  
   D) Standard Queue  
   **Answer: C) Priority Queue**
9. A data structure that allows elements to be added or removed from both the front and the rear is called a:  
   A) Stack  
   B) Priority Queue  
   C) Circular Queue  
   D) Deque (Double-Ended Queue)  
   **Answer: D) Deque (Double-Ended Queue)**
10. What is the typical time complexity for push() and pop() operations on a java.util.ArrayDeque when used as a stack?  
    A) O(n)  
    B) O(log n)  
    C) O(1)  
    D) O(n log n)  
    **Answer: C) O(1)**

**Linked Lists**

1. What is the main disadvantage of a Linked List compared to an Array for accessing elements?  
   A) Linked lists use more memory per element.  
   B) Linked lists have fixed size.  
   C) Linked lists require O(n) time for random access (by index).  
   D) Linked lists are more complex to implement.  
   **Answer: C) Linked lists require O(n) time for random access (by index).**
2. What is an advantage of a Doubly Linked List over a Singly Linked List?  
   A) Uses less memory per node.  
   B) Allows traversal in both forward and backward directions.  
   C) Provides O(1) random access to elements.  
   D) Simpler to implement.  
   **Answer: B) Allows traversal in both forward and backward directions.**
3. In a Circular Linked List, what is the characteristic of the last node's next pointer?  
   A) It points to null.  
   B) It points to the head node.  
   C) It points to the middle node.  
   D) It points to the tail node.  
   **Answer: B) It points to the head node.**
4. What is the time complexity to insert a new node at the head of a Singly Linked List?  
   A) O(n)  
   B) O(log n)  
   C) O(1)  
   D) O(n^2)  
   **Answer: C) O(1)**

**Trees**

1. In a tree data structure, a node that has no children is called a:  
   A) Root node  
   B) Parent node  
   C) Leaf node (External Node)  
   D) Internal node  
   **Answer: C) Leaf node (External Node)**
2. Which tree traversal algorithm visits nodes level by level, typically using a Queue?  
   A) In-order Traversal  
   B) Pre-order Traversal  
   C) Post-order Traversal  
   D) Breadth-First Search (BFS) / Level Order Traversal  
   **Answer: D) Breadth-First Search (BFS) / Level Order Traversal**
3. Which tree traversal, when applied to a Binary Search Tree (BST), visits nodes in ascending order of their values?  
   A) Pre-order Traversal  
   B) In-order Traversal  
   C) Post-order Traversal  
   D) Level Order Traversal  
   **Answer: B) In-order Traversal**
4. What is the fundamental property of a Binary Search Tree (BST)?  
   A) Each node has at most two children.  
   B) All levels are completely filled.  
   C) For every node, values in its left subtree are less than its value, and values in its right subtree are greater.  
   D) All internal nodes have two children.  
   **Answer: C) For every node, values in its left subtree are less than its value, and values in its right subtree are greater.**
5. Why are self-balancing Binary Search Trees (like AVL or Red-Black Trees) preferred over simple BSTs in many applications?  
   A) They are simpler to implement.  
   B) They guarantee O(log n) time complexity for search, insertion, and deletion in the worst case.  
   C) They use less memory per node.  
   D) They can store duplicate values more efficiently.  
   **Answer: B) They guarantee O(log n) time complexity for search, insertion, and deletion in the worst case.**
6. Which Java utility class typically implements a self-balancing binary search tree, specifically a Red-Black Tree, for sorted key-value mapping?  
   A) java.util.HashMap  
   B) java.util.TreeMap  
   C) java.util.HashSet  
   D) java.util.ArrayList  
   **Answer: B) java.util.TreeMap**

**Searching & Sorting**

1. What is a prerequisite for using Binary Search on an array?  
   A) The array must contain unique elements.  
   B) The array must be sorted.  
   C) The array must be a prime number in size.  
   D) The array must contain only integers.  
   **Answer: B) The array must be sorted.**
2. What is the worst-case time complexity of a Linear Search algorithm?  
   A) O(1)  
   B) O(log n)  
   C) O(n)  
   D) O(n^2)  
   **Answer: C) O(n)**
3. What is the time complexity of the Binary Search algorithm?  
   A) O(n)  
   B) O(n log n)  
   C) O(log n)  
   D) O(n^2)  
   **Answer: C) O(log n)**
4. Which sorting algorithm picks an element as a pivot and partitions the array around it, then recursively sorts the sub-arrays?  
   A) Merge Sort  
   B) Heap Sort  
   C) Quick Sort  
   D) Bubble Sort  
   **Answer: C) Quick Sort**
5. Which sorting algorithm is a divide and conquer algorithm, has a worst-case time complexity of O(n log n), but requires O(n) auxiliary space?  
   A) Quick Sort  
   B) Heap Sort  
   C) Merge Sort  
   D) Insertion Sort  
   **Answer: C) Merge Sort**
6. Which of the following sorting algorithms consistently has a time complexity of O(n^2) in its worst, best, and average cases, and is generally in-place?  
   A) Heap Sort  
   B) Merge Sort  
   C) Selection Sort  
   D) Quick Sort  
   **Answer: C) Selection Sort** (Bubble Sort also fits O(n^2) consistently, but Selection Sort is often used as the canonical "always O(N^2)" example due to its fixed number of swaps.)
7. A sorting algorithm is considered **stable** if:  
   A) It always sorts in ascending order.  
   B) It uses a constant amount of extra space.  
   C) It preserves the relative order of equal elements.  
   D) Its time complexity does not change with input distribution.  
   **Answer: C) It preserves the relative order of equal elements.**
8. Which sorting algorithm uses a Binary Heap data structure internally to sort elements?  
   A) Merge Sort  
   B) Quick Sort  
   C) Heap Sort  
   D) Insertion Sort  
   **Answer: C) Heap Sort**
9. What is the best-case time complexity of Insertion Sort?  
   A) O(1)  
   B) O(log n)  
   C) O(n)  
   D) O(n^2)  
   **Answer: C) O(n)** (when the array is already sorted)

**Hashing & Introduction to Hash Tables**

1. The primary goal of a Hash Table is to achieve an average time complexity of O(1) for which operations?  
   A) Sorting elements  
   B) Insertion, deletion, and lookup (search)  
   C) Traversing all elements  
   D) Calculating permutations  
   **Answer: B) Insertion, deletion, and lookup (search)**
2. In the context of hash tables, what is a **collision**?  
   A) When two different hash functions are used.  
   B) When the hash table runs out of memory.  
   C) When two different keys hash to the same index.  
   D) When an element is not found in the table.  
   **Answer: C) When two different keys hash to the same index.**
3. Which collision resolution strategy involves storing a linked list (or other data structure) at each hash table index to hold all keys that map to that index?  
   A) Linear Probing  
   B) Quadratic Probing  
   C) Double Hashing  
   D) Separate Chaining (Chained Hashtables)  
   **Answer: D) Separate Chaining (Chained Hashtables)**
4. Which open addressing collision resolution strategy can suffer from "primary clustering," where long runs of occupied slots form?  
   A) Quadratic Probing  
   B) Double Hashing  
   C) Linear Probing  
   D) Separate Chaining  
   **Answer: C) Linear Probing**
5. When the load factor of a hash table exceeds a certain threshold, what action is typically taken to maintain performance?  
   A) The hash function is changed.  
   B) The table undergoes rehashing (resizing and re-inserting elements).  
   C) All elements are sorted.  
   D) The table converts to a linked list.  
   **Answer: B) The table undergoes rehashing (resizing and re-inserting elements).**
6. What is the average-case time complexity for put(), get(), and remove() operations in a well-designed hash table?  
   A) O(n)  
   B) O(log n)  
   C) O(1)  
   D) O(n log n)  
   **Answer: C) O(1)**

**Graphs**

1. In graph theory, what is an "edge"?  
   A) A fundamental unit containing data.  
   B) A connection between two vertices.  
   C) The topmost node in a tree.  
   D) A sequence of connected edges that starts and ends at the same vertex.  
   **Answer: B) A connection between two vertices.**
2. For a **sparse graph** (a graph with relatively few edges), which representation is generally more memory-efficient?  
   A) Adjacency Matrix  
   B) Adjacency List  
   C) Incidence Matrix  
   D) Edge List (without vertex list)  
   **Answer: B) Adjacency List**
3. Which graph traversal algorithm is ideal for finding the shortest path (in terms of number of edges) from a single source to all other reachable vertices in an **unweighted** graph?  
   A) Depth-First Search (DFS)  
   B) Breadth-First Search (BFS)  
   C) Dijkstra's Algorithm  
   D) Bellman-Ford Algorithm  
   **Answer: B) Breadth-First Search (BFS)**
4. Which algorithm finds the shortest paths from a single source vertex to all other vertices in a graph with **non-negative edge weights**?  
   A) Bellman-Ford Algorithm  
   B) Floyd-Warshall Algorithm  
   C) Dijkstra's Algorithm  
   D) Kruskal's Algorithm  
   **Answer: C) Dijkstra's Algorithm**
5. Which dynamic programming algorithm finds the shortest paths between **all pairs** of vertices in a weighted graph (can handle negative weights but not negative cycles)?  
   A) Dijkstra's Algorithm  
   B) Bellman-Ford Algorithm  
   C) Prim's Algorithm  
   D) Floyd-Warshall Algorithm  
   **Answer: D) Floyd-Warshall Algorithm**
6. Which greedy algorithm for finding a Minimum Spanning Tree (MST) starts from an arbitrary vertex and iteratively adds the cheapest edge connecting a vertex in the growing MST to a vertex outside it?  
   A) Kruskal's Algorithm  
   B) Prim's Algorithm  
   C) Dijkstra's Algorithm  
   D) Bellman-Ford Algorithm  
   **Answer: B) Prim's Algorithm**
7. Which greedy algorithm for finding a Minimum Spanning Tree (MST) works by sorting all edges by weight and adding them if they don't form a cycle, typically using a Disjoint Set Union (DSU) data structure?  
   A) Prim's Algorithm  
   B) Kruskal's Algorithm  
   C) Floyd-Warshall Algorithm  
   D) BFS  
   **Answer: B) Kruskal's Algorithm**
8. In graph theory, a graph where edges have a specific direction (e.g., from A to B, but not necessarily B to A) is called a:  
   A) Undirected Graph  
   B) Complete Graph  
   C) Connected Graph  
   D) Directed Graph (Digraph)  
   **Answer: D) Directed Graph (Digraph)**
9. For a connected graph with V vertices, how many edges will its spanning tree always have?  
   A) V  
   B) V+1  
   C) V-1  
   D) 2V  
   **Answer: C) V-1**
10. Which single-source shortest path algorithm can handle graphs with **negative edge weights** and detect negative cycles?  
    A) Dijkstra's Algorithm  
    B) Prim's Algorithm  
    C) Bellman-Ford Algorithm  
    D) Floyd-Warshall Algorithm  
    **Answer: C) Bellman-Ford Algorithm**

**Analysis of an Algorithm & Asymptotic Analysis**

1. The running time O(n!) typically indicates a problem solved using which approach?  
   A) Polynomial time  
   B) Exponential time  
   C) Logarithmic time  
   D) Factorial time  
   **Answer: D) Factorial time**
2. When analyzing an algorithm's performance, what does "dominant terms" refer to as input size grows?  
   A) The terms that involve constants.  
   B) The terms with the highest growth rate.  
   C) The terms that are always positive.  
   D) The terms that are numerically largest for small inputs.  
   **Answer: B) The terms with the highest growth rate.**
3. What does O(2^n) time complexity usually signify?  
   A) Linear scalability  
   B) Logarithmic growth  
   C) Exponential growth, often from brute-force recursive solutions  
   D) Constant time execution  
   **Answer: C) Exponential growth, often from brute-force recursive solutions**

**Analysis of Different Types of Algorithms**

1. Dijkstra's Algorithm is an example of which algorithmic paradigm?  
   A) Dynamic Programming  
   B) Divide and Conquer  
   C) Greedy Algorithm  
   D) Backtracking  
   **Answer: C) Greedy Algorithm**
2. The 0/1 Knapsack Problem is a classic example often solved using which algorithmic paradigm?  
   A) Greedy Algorithm (for fractional)  
   B) Divide and Conquer  
   C) Dynamic Programming  
   D) Brute Force  
   **Answer: C) Dynamic Programming**
3. Which algorithmic paradigm guarantees a solution if one exists, but is usually the most inefficient due to exhaustive search?  
   A) Dynamic Programming  
   B) Greedy Algorithm  
   C) Divide and Conquer  
   D) Brute Force  
   **Answer: D) Brute Force**
4. The N-Queens problem is a typical application for which algorithmic technique?  
   A) Dynamic Programming  
   B) Greedy Algorithm  
   C) Backtracking  
   D) Divide and Conquer  
   **Answer: C) Backtracking**
5. What is a key characteristic of a Greedy Algorithm?  
   A) It always finds the globally optimal solution.  
   B) It stores results of subproblems to avoid recomputation.  
   C) It makes immediate, locally optimal choices without looking ahead.  
   D) It recursively breaks problems into smaller subproblems.  
   **Answer: C) It makes immediate, locally optimal choices without looking ahead.**
6. Which algorithmic paradigm often uses memoization or tabulation to store results of subproblems?  
   A) Divide and Conquer  
   B) Greedy Algorithm  
   C) Dynamic Programming  
   D) Backtracking  
   **Answer: C) Dynamic Programming**

**Complexity & Application of Data Structures**

1. When discussing time complexity in a Java context, what does it typically refer to?  
   A) The number of lines of code.  
   B) The number of CPU instructions or function calls.  
   C) The amount of data stored on the hard drive.  
   D) The exact execution time in milliseconds.  
   **Answer: B) The number of CPU instructions or function calls.**
2. If an algorithm requires an ArrayList that can dynamically grow, which component contributes to its auxiliary space complexity?  
   A) Only the initial capacity of the array.  
   B) The space for the input data.  
   C) The memory used for elements and internal array resizing.  
   D) The program's executable file size.  
   **Answer: C) The memory used for elements and internal array resizing.**
3. Choosing a Hash Table over a Binary Search Tree might be preferable if the primary operation is:  
   A) Retrieving elements in sorted order.  
   B) Performing range queries.  
   C) Frequent searches, insertions, and deletions with average O(1) time.  
   D) Storing hierarchical data.  
   **Answer: C) Frequent searches, insertions, and deletions with average O(1) time.**

**Introductory Concepts & Basic Data Structures**

1. In Java, if-else and switch statements are examples of which algorithm construct?  
   A) Sequential Statements  
   B) Conditional Statements (Selection)  
   C) Looping Statements (Iteration)  
   D) Function/Method Calls  
   **Answer: B) Conditional Statements (Selection)**
2. java.util.List and java.util.ArrayList represent which relationship in Java ADT context?  
   A) List is a concrete implementation of ArrayList.  
   B) ArrayList is an interface, List is its implementation.  
   C) List defines the ADT, and ArrayList is a concrete implementation.  
   D) They are both ADTs with no concrete implementations.  
   **Answer: C) List defines the ADT, and ArrayList is a concrete implementation.**
3. What is a characteristic of a traditional, fixed-size array?  
   A) Dynamic size, can grow or shrink easily.  
   B) Efficient insertion/deletion in the middle (O(1)).  
   C) Direct access to elements by index (O(1) time complexity).  
   D) More memory overhead due to pointers.  
   **Answer: C) Direct access to elements by index (O(1) time complexity).**
4. Which of the following is a common use case for a Stack data structure?  
   A) Implementing task scheduling.  
   B) Breadth-First Search (BFS) algorithm.  
   C) Function call stack management (e.g., recursion).  
   D) Managing a line of customers.  
   **Answer: C) Function call stack management (e.g., recursion).**
5. What is the peek() operation in a Stack?  
   A) Removes and returns the top element.  
   B) Adds an element to the top of the stack.  
   C) Returns the top element without removing it.  
   D) Checks if the stack is empty.  
   **Answer: C) Returns the top element without removing it.**
6. In a Queue, what is the operation offer() typically used for?  
   A) Removing an element from the front.  
   B) Adding an element to the rear.  
   C) Checking the element at the front without removal.  
   D) Checking if the queue is empty.  
   **Answer: B) Adding an element to the rear.**
7. A Circular Queue implemented using an array prevents which issue of a regular array-based queue?  
   A) Inefficient use of space due to elements being dequeued from the front leaving gaps.  
   B) O(n) operations for enqueue/dequeue.  
   C) Lack of ability to store elements of different types.  
   D) Inability to determine if the queue is full.  
   **Answer: A) Inefficient use of space due to elements being dequeued from the front leaving gaps.**
8. What data structure is typically used to implement a Priority Queue?  
   A) Linked List  
   B) Array  
   C) Hash Table  
   D) Heap  
   **Answer: D) Heap**
9. By default, java.util.PriorityQueue is a:  
   A) Max-heap  
   B) Min-heap  
   C) Hash table  
   D) Linked list  
   **Answer: B) Min-heap**
10. A Deque can efficiently function as both a Stack and a Queue because:  
    A) It is implemented using a tree.  
    B) It supports adding/removing elements from both ends (front and rear) in O(1).  
    C) It automatically sorts elements upon insertion.  
    D) It has fixed size and uses contiguous memory.  
    **Answer: B) It supports adding/removing elements from both ends (front and rear) in O(1).**

**Linked Lists**

1. What is the time complexity to search for a specific value in an unsorted Singly Linked List?  
   A) O(1)  
   B) O(log n)  
   C) O(n)  
   D) O(n log n)  
   **Answer: C) O(n)**
2. What is a key disadvantage of Linked Lists compared to Arrays in terms of memory usage?  
   A) Linked Lists use less memory overall.  
   B) Linked Lists have more memory overhead due to storing pointers/references.  
   C) Linked Lists store elements in contiguous memory, leading to fragmentation.  
   D) Linked Lists require a fixed amount of memory at compile time.  
   **Answer: B) Linked Lists have more memory overhead due to storing pointers/references.**
3. In a Doubly Linked List, why is deletion generally more efficient than in a Singly Linked List, once the node to be deleted is found?  
   A) It doesn't require shifting elements.  
   B) It has direct access to the previous node, making relinking easier.  
   C) It uses less memory for pointers.  
   D) It is always sorted.  
   **Answer: B) It has direct access to the previous node, making relinking easier.**
4. When might "Node-based storage with arrays" (e.g., using array indices as pointers) be particularly useful for implementing a Linked List?  
   A) When high-speed random access is paramount.  
   B) In memory-constrained environments or for custom memory pool management.  
   C) When the list size is dynamic and unpredictable.  
   D) To enable automatic garbage collection.  
   **Answer: B) In memory-constrained environments or for custom memory pool management.**

**Trees**

1. In a tree, the "depth of a node" refers to:  
   A) The number of children it has.  
   B) The number of edges from that node to the deepest leaf.  
   C) The number of edges from the root to that node.  
   D) The total number of nodes in its subtree.  
   **Answer: C) The number of edges from the root to that node.**
2. A "Full Binary Tree" is defined as a binary tree where:  
   A) All leaf nodes are at the same depth.  
   B) All levels are completely filled.  
   C) Every node has either 0 or 2 children.  
   D) Every node has at most two children.  
   **Answer: C) Every node has either 0 or 2 children.**
3. Which type of binary tree is often implemented efficiently using arrays due to its specific structure (all levels filled except possibly the last, and last level nodes as far left as possible)?  
   A) Full Binary Tree  
   B) Perfect Binary Tree  
   C) Complete Binary Tree  
   D) Skewed Binary Tree  
   **Answer: C) Complete Binary Tree**
4. What is the primary use case for a Pre-order Traversal (Root, Left, Right)?  
   A) To print elements in ascending order.  
   B) To delete a tree efficiently.  
   C) To create a copy of the tree or obtain a prefix expression.  
   D) To find the minimum element in a BST.  
   **Answer: C) To create a copy of the tree or obtain a prefix expression.**
5. In a Binary Search Tree, what is the worst-case time complexity for search, insertion, or deletion operations?  
   A) O(log n)  
   B) O(n log n)  
   C) O(n)  
   D) O(1)  
   **Answer: C) O(n)** (when the tree becomes skewed)
6. Which self-balancing BST strictly maintains a height balance factor of -1, 0, or 1?  
   A) Red-Black Tree  
   B) AVL Tree  
   C) Splay Tree  
   D) B-Tree  
   **Answer: B) AVL Tree**
7. What is the "height of a tree"?  
   A) The number of nodes in the longest path from root to leaf.  
   B) The height of its root node.  
   C) The number of levels in the tree.  
   D) The maximum depth of any node.  
   **Answer: B) The height of its root node.** (Which is equivalent to max depth of any node, or number of edges in longest path from root to leaf.)

**Searching & Sorting**

1. What is the space complexity of iterative Binary Search?  
   A) O(n)  
   B) O(log n)  
   C) O(1)  
   D) O(n log n)  
   **Answer: C) O(1)**
2. What is the best-case time complexity of Quick Sort?  
   A) O(n^2)  
   B) O(n log n)  
   C) O(n)  
   D) O(1)  
   **Answer: B) O(n log n)**
3. Which of the following sorting algorithms is **not** typically considered "in-place" due to its O(n) auxiliary space requirement?  
   A) Bubble Sort  
   B) Selection Sort  
   C) Quick Sort  
   D) Merge Sort  
   **Answer: D) Merge Sort**
4. Bubble Sort is generally considered very inefficient, with a time complexity of O(n^2). Why is it still often taught?  
   A) It has the best performance for small datasets.  
   B) It is stable and in-place.  
   C) It is simple to understand and implement, illustrating basic sorting concepts.  
   D) It is used in specialized hardware.  
   **Answer: C) It is simple to understand and implement, illustrating basic sorting concepts.**
5. How is the worst-case scenario of Quick Sort (O(n^2)) typically mitigated in practice?  
   A) By converting it to a Merge Sort.  
   B) By sorting the array before applying Quick Sort.  
   C) By randomizing the pivot choice.  
   D) By always picking the first element as a pivot.  
   **Answer: C) By randomizing the pivot choice.**
6. Which of the following sorting algorithms is generally **unstable**?  
   A) Insertion Sort  
   B) Merge Sort  
   C) Bubble Sort  
   D) Heap Sort  
   **Answer: D) Heap Sort**
7. For a nearly sorted array, which sorting algorithm performs exceptionally well with a best-case time complexity of O(n)?  
   A) Selection Sort  
   B) Quick Sort  
   C) Insertion Sort  
   D) Heap Sort  
   **Answer: C) Insertion Sort**

**Hashing & Introduction to Hash Tables**

1. A good hash function should ideally produce a **uniform distribution** of keys across the hash table's array indices to:  
   A) Reduce memory usage.  
   B) Maximize collisions.  
   C) Minimize collisions and improve performance.  
   D) Make the hash table sorted.  
   **Answer: C) Minimize collisions and improve performance.**
2. If a hash function returns a large integer, what operation is commonly used to map this hash value to a valid index within the hash table's array of size M?  
   A) Multiplication  
   B) Division  
   C) Modulo operator (%)  
   D) Addition  
   **Answer: C) Modulo operator (%)**
3. What is the primary issue that "Linear Probing" suffers from in open addressing?  
   A) Secondary Clustering  
   B) Memory Leakage  
   C) Primary Clustering  
   D) Infinite loops  
   **Answer: C) Primary Clustering**
4. In "Quadratic Probing," if a collision occurs at index i, how are subsequent slots typically probed?  
   A) i+1, i+2, i+3, ...  
   B) i+1^2, i+2^2, i+3^2, ...  
   C) Using a second hash function.  
   D) By starting a new linked list.  
   **Answer: B) i+1^2, i+2^2, i+3^2, ...**
5. Why is it crucial to correctly implement hashCode() and equals() methods for custom objects used as keys in java.util.HashMap?  
   A) To ensure objects are always sorted.  
   B) To prevent memory overflow.  
   C) To ensure correct hashing and equality checks for key retrieval.  
   D) To enable O(log n) performance.  
   **Answer: C) To ensure correct hashing and equality checks for key retrieval.**
6. The Poisson Distribution is relevant in the theoretical analysis of hash tables to model:  
   A) The average load factor of an empty table.  
   B) The time taken to rehash the table.  
   C) The number of collisions and length of chains in separate chaining.  
   D) The optimal size of the hash table.  
   **Answer: C) The number of collisions and length of chains in separate chaining.**
7. What is the worst-case time complexity for operations in a Hash Table (e.g., insertion, deletion, search)?  
   A) O(1)  
   B) O(log n)  
   C) O(n)  
   D) O(n log n)  
   **Answer: C) O(n)** (if all keys hash to the same bucket)

**Graphs**

1. In a Directed Graph, what does the "in-degree" of a vertex represent?  
   A) The number of edges originating from that vertex.  
   B) The number of edges terminating at that vertex.  
   C) The total number of edges connected to that vertex.  
   D) The number of paths from the root to that vertex.  
   **Answer: B) The number of edges terminating at that vertex.**
2. A graph that contains no cycles is known as a(n):  
   A) Complete Graph  
   B) Weighted Graph  
   C) Acyclic Graph  
   D) Bipartite Graph  
   **Answer: C) Acyclic Graph**
3. For a graph with V vertices and E edges, which representation provides O(1) time complexity to check if an edge exists between two specific vertices?  
   A) Adjacency List  
   B) Adjacency Matrix  
   C) Edge List  
   D) Implicit Representation  
   **Answer: B) Adjacency Matrix**
4. What data structure does Breadth-First Search (BFS) primarily use for its traversal?  
   A) Stack  
   B) Priority Queue  
   C) Array  
   D) Queue  
   **Answer: D) Queue**
5. Depth-First Search (DFS) is well-suited for which of the following graph problems?  
   A) Finding the shortest path in unweighted graphs.  
   B) Finding all connected components.  
   C) Topological sorting of a Directed Acyclic Graph (DAG).  
   D) Finding the shortest path in weighted graphs.  
   **Answer: C) Topological sorting of a Directed Acyclic Graph (DAG).**
6. Which algorithm can be used to identify if an undirected graph is Bipartite?  
   A) Dijkstra's Algorithm  
   B) Floyd-Warshall Algorithm  
   C) BFS or DFS with a coloring scheme.  
   D) Kruskal's Algorithm  
   **Answer: C) BFS or DFS with a coloring scheme.**
7. When is the Bellman-Ford algorithm preferred over Dijkstra's algorithm for finding single-source shortest paths?  
   A) When the graph is unweighted.  
   B) When the graph contains negative edge weights.  
   C) When the graph is dense.  
   D) When only a single path needs to be found.  
   **Answer: B) When the graph contains negative edge weights.**
8. What is a "Spanning Tree" of a connected, undirected graph?  
   A) A subgraph that contains some vertices and some edges.  
   B) A subgraph that includes all vertices of the original graph and is a tree (connected with no cycles).  
   C) A path that visits every vertex exactly once.  
   D) A graph that has no edges.  
   **Answer: B) A subgraph that includes all vertices of the original graph and is a tree (connected with no cycles).**
9. How does Kruskal's algorithm build a Minimum Spanning Tree (MST)?  
   A) By starting from a vertex and growing the tree by adding cheapest incident edges.  
   B) By iteratively adding edges in increasing order of weight, avoiding cycles.  
   C) By finding the shortest path between all pairs of vertices.  
   D) By removing edges until a tree is formed.  
   **Answer: B) By iteratively adding edges in increasing order of weight, avoiding cycles.**
10. A "Dense Graph" is characterized by:  
    A) Having very few vertices.  
    B) Having many edges, close to the maximum possible.  
    C) Being disconnected.  
    D) Having a low average degree.  
    **Answer: B) Having many edges, close to the maximum possible.**
11. Which term describes a path that starts and ends at the same vertex, visiting other vertices in between?  
    A) Edge  
    B) Path  
    C) Cycle  
    D) Vertex  
    **Answer: C) Cycle**
12. What is the time complexity of Dijkstra's algorithm when implemented using a binary heap (PriorityQueue)?  
    A) O(V^2)  
    B) O(E log V)  
    C) O(V \* E)  
    D) O(V^3)  
    **Answer: B) O(E log V)**
13. Which algorithm is used to find all-pairs shortest paths in a graph?  
    A) Dijkstra's Algorithm  
    B) Bellman-Ford Algorithm  
    C) Prim's Algorithm  
    D) Floyd-Warshall Algorithm  
    **Answer: D) Floyd-Warshall Algorithm**

**Analysis of an Algorithm & Asymptotic Analysis**

1. When expressing algorithm complexity using Big O notation, why are constant factors and lower-order terms typically ignored?  
   A) They are too difficult to calculate precisely.  
   B) They become insignificant as the input size (n) grows infinitely large.  
   C) They only apply to specific hardware configurations.  
   D) They indicate the best-case performance, which is not usually considered.  
   **Answer: B) They become insignificant as the input size (n) grows infinitely large.**
2. Which time complexity typically describes an algorithm that halves the input data in each step, such as Binary Search?  
   A) O(1)  
   B) O(log n)  
   C) O(n)  
   D) O(n log n)  
   **Answer: B) O(log n)**
3. An algorithm with O(n^2) time complexity is generally considered inefficient for large input sizes because:  
   A) Its running time is constant.  
   B) Its running time grows linearly with the input.  
   C) Its running time grows quadratically with the input, making it very slow for large 'n'.  
   D) It always requires a sorted input.  
   **Answer: C) Its running time grows quadratically with the input, making it very slow for large 'n'.**
4. The "exact execution time" of an algorithm is most influenced by:  
   A) The asymptotic complexity (e.g., O(n)).  
   B) The compiler, hardware, and specific input data.  
   C) Only the size of the input.  
   D) The choice of programming language alone.  
   **Answer: B) The compiler, hardware, and specific input data.**

**Analysis of Different Types of Algorithms**

1. The "Optimal Substructure" property, where an optimal solution to a problem can be constructed from optimal solutions to its subproblems, is a key characteristic of:  
   A) Brute Force Algorithms  
   B) Greedy Algorithms (when they work)  
   C) Dynamic Programming  
   D) Stochastic Algorithms  
   **Answer: C) Dynamic Programming**
2. What distinguishes Branch-and-Bound from pure Backtracking?  
   A) Branch-and-Bound only works for decision problems, while Backtracking works for optimization.  
   B) Branch-and-Bound prunes branches that cannot lead to a better solution using bounds.  
   C) Backtracking uses recursion, while Branch-and-Bound uses iteration.  
   D) Branch-and-Bound explores all possible solutions, while Backtracking prunes early.  
   **Answer: B) Branch-and-Bound prunes branches that cannot lead to a better solution using bounds.**
3. Which of the following problems is *not* typically solved using a Divide and Conquer approach?  
   A) Binary Search  
   B) Merge Sort  
   C) Quick Sort  
   D) Fibonacci Sequence (recursive approach without memoization, but DP for efficiency)  
   **Answer: D) Fibonacci Sequence (While a naive recursive solution is D&C, its inefficiency due to overlapping subproblems makes DP the standard efficient approach, unlike the others which are canonical D&C.)**
4. Prim's Algorithm for MST and Dijkstra's Algorithm for shortest path are both examples of:  
   A) Dynamic Programming  
   B) Backtracking  
   C) Greedy Algorithms  
   D) Brute Force  
   **Answer: C) Greedy Algorithms**
5. When is a Brute Force approach often chosen despite its inefficiency?  
   A) For very large inputs where speed is critical.  
   B) When the problem space is small or no more efficient algorithm is known/needed.  
   C) When the solution needs to be approximate.  
   D) When dealing with negative cycles in graphs.  
   **Answer: B) When the problem space is small or no more efficient algorithm is known/needed.**

**Complexity & Application of Data Structures**

1. If an algorithm's space complexity is O(1), it means it uses:  
   A) No memory at all.  
   B) A constant amount of extra memory regardless of input size.  
   C) Memory proportional to the input size.  
   D) Memory proportional to the square of the input size.  
   **Answer: B) A constant amount of extra memory regardless of input size.**
2. Why might a java.util.Stack be less performant than java.util.ArrayDeque for stack operations if synchronization is not needed?  
   A) Stack uses a linked list internally.  
   B) Stack extends Vector and is synchronized, incurring overhead.  
   C) ArrayDeque has a fixed size.  
   D) ArrayDeque is recursive.  
   **Answer: B) Stack extends Vector and is synchronized, incurring overhead.**

**Introductory Concepts & Basic Data Structures**

1. What is the fundamental difference between an ADT and a concrete data structure implementation?  
   A) An ADT focuses on *how* data is stored; an implementation focuses on *what* operations are performed.  
   B) An ADT defines the interface/behavior; an implementation provides the specific underlying code and storage.  
   C) An ADT can only be used with primitive data types; an implementation can use objects.  
   D) An ADT is always faster than its concrete implementation.  
   **Answer: B) An ADT defines the interface/behavior; an implementation provides the specific underlying code and storage.**
2. In Java, what happens internally when an ArrayList reaches its capacity and a new element is added?  
   A) The ArrayList converts to a LinkedList.  
   B) A new, larger array is allocated, and elements are copied to it.  
   C) The ArrayList throws an OutOfMemoryError.  
   D) Elements are automatically compressed to save space.  
   **Answer: B) A new, larger array is allocated, and elements are copied to it.**
3. What is the effect of the poll() method on a java.util.Queue?  
   A) It adds an element to the rear of the queue.  
   B) It returns the element at the front of the queue without removing it.  
   C) It removes and returns the element at the front of the queue, or returns null if empty.  
   D) It throws an exception if the queue is empty.  
   **Answer: C) It removes and returns the element at the front of the queue, or returns null if empty.**
4. Which data structure is explicitly mentioned as being used for BFS (Breadth-First Search) algorithm?  
   A) Stack  
   B) Priority Queue  
   C) Queue  
   D) Deque  
   **Answer: C) Queue**
5. For a java.util.PriorityQueue, if you want to retrieve elements in descending order (i.e., act as a max-heap), what must you typically provide?  
   A) A custom equals() method.  
   B) A Comparator in its constructor.  
   C) A larger initial capacity.  
   D) It automatically acts as a max-heap.  
   **Answer: B) A Comparator in its constructor.**
6. What is the primary benefit of a Circular Queue over a standard array-based queue when dealing with limited memory?  
   A) Faster element access.  
   B) Prevents wasted memory space after dequeuing elements.  
   C) Supports random access.  
   D) Can store elements of different data types.  
   **Answer: B) Prevents wasted memory space after dequeuing elements.**

**Linked Lists**

1. When performing an insertion or deletion operation in a Linked List, what is the time complexity *after* the point of insertion/deletion has been found?  
   A) O(n)  
   B) O(log n)  
   C) O(1)  
   D) O(n^2)  
   **Answer: C) O(1)**
2. Why is random access (accessing an element by its index, e.g., list.get(5)) inefficient (O(n)) in a Singly Linked List?  
   A) Elements are not stored in contiguous memory.  
   B) Each node only knows about the next node, requiring traversal from the head.  
   C) Linked lists have fixed sizes.  
   D) Pointers consume too much memory.  
   **Answer: B) Each node only knows about the next node, requiring traversal from the head.**
3. Which type of Linked List is most suitable for implementing a Least Recently Used (LRU) cache, where elements need to be efficiently moved to the front upon access?  
   A) Singly Linked List  
   B) Doubly Linked List  
   C) Circular Linked List  
   D) Array-based Linked List  
   **Answer: B) Doubly Linked List**
4. In a Doubly Linked List, what does the prev pointer of the head node typically point to?  
   A) The tail node.  
   B) null.  
   C) The second node.  
   D) It is an invalid concept.  
   **Answer: B) null.**

**Trees**

1. In a tree, nodes that share the same parent are called:  
   A) Root  
   B) Children  
   C) Siblings  
   D) Ancestors  
   **Answer: C) Siblings**
2. What is the "height of a node" in a tree?  
   A) The length of the path from the root to that node.  
   B) The number of children the node has.  
   C) The length of the longest path from that node to a leaf node.  
   D) The number of nodes in its subtree.  
   **Answer: C) The length of the longest path from that node to a leaf node.**
3. A "Perfect Binary Tree" implies:  
   A) All internal nodes have two children, and all leaf nodes are at the same depth.  
   B) All levels are completely filled except possibly the last.  
   C) Every node has either 0 or 2 children.  
   D) The tree is always balanced.  
   **Answer: A) All internal nodes have two children, and all leaf nodes are at the same depth.**
4. Which DFS traversal is typically used to delete a tree because it processes children before the parent?  
   A) In-order Traversal  
   B) Pre-order Traversal  
   C) Post-order Traversal  
   D) Level Order Traversal  
   **Answer: C) Post-order Traversal**
5. In a Binary Search Tree, if elements are inserted in strictly ascending order (e.g., 1, 2, 3, 4, 5), the tree will degenerate into a structure similar to a:  
   A) Complete Binary Tree  
   B) Perfect Binary Tree  
   C) Skewed Binary Tree (like a linked list)  
   D) Full Binary Tree  
   **Answer: C) Skewed Binary Tree (like a linked list)**
6. What is the primary purpose of tree rotations in self-balancing BSTs like AVL or Red-Black Trees?  
   A) To simplify deletion logic.  
   B) To maintain the height balance and preserve O(log n) performance.  
   C) To reduce the total number of nodes.  
   D) To enable faster insertion than search.  
   **Answer: B) To maintain the height balance and preserve O(log n) performance.**
7. Which of the following is true about a Red-Black Tree compared to an AVL Tree?  
   A) Red-Black Trees are stricter on height balance.  
   B) Red-Black Trees generally perform fewer rotations during insertions/deletions.  
   C) AVL Trees are more commonly found in standard library implementations.  
   D) Red-Black Trees do not guarantee O(log n) worst-case performance.  
   **Answer: B) Red-Black Trees generally perform fewer rotations during insertions/deletions.**

**Searching & Sorting**

1. If a search key is smaller than the item in the middle of a sorted interval during binary search, the algorithm narrows the interval to the:  
   A) Upper half  
   B) Lower half  
   C) Entire array  
   D) The middle element itself  
   **Answer: B) Lower half**
2. What is the auxiliary space complexity of Merge Sort?  
   A) O(1)  
   B) O(log n)  
   C) O(n)  
   D) O(n log n)  
   **Answer: C) O(n)**
3. Which of the following scenarios would lead to the *worst-case* time complexity for Quick Sort?  
   A) The pivot always being the median element.  
   B) The array being randomly shuffled.  
   C) The pivot selection consistently choosing the smallest or largest element in an already sorted array.  
   D) The array containing all identical elements.  
   **Answer: C) The pivot selection consistently choosing the smallest or largest element in an already sorted array.**
4. Which sorting algorithm is considered stable and in-place, and relatively efficient for small or nearly sorted arrays?  
   A) Quick Sort  
   B) Heap Sort  
   C) Insertion Sort  
   D) Selection Sort  
   **Answer: C) Insertion Sort**
5. What does it mean for a sorting algorithm to be "in-place"?  
   A) It completes its sorting in a very short time.  
   B) It uses a constant amount of auxiliary space (O(1)).  
   C) It can only sort arrays, not linked lists.  
   D) It never moves elements during the sorting process.  
   **Answer: B) It uses a constant amount of auxiliary space (O(1)).**
6. For datasets that are very large and cannot fit entirely into memory, which sorting algorithm is often preferred due to its external sorting capabilities?  
   A) Quick Sort  
   B) Heap Sort  
   C) Merge Sort  
   D) Bubble Sort  
   **Answer: C) Merge Sort**
7. Given an array [5, 2, 8, 1, 9], after the first pass of Bubble Sort, what could be a possible state of the array if sorted in ascending order?  
   A) [1, 2, 5, 8, 9]  
   B) [2, 5, 1, 8, 9]  
   C) [2, 5, 8, 1, 9]  
   D) [5, 8, 2, 1, 9]  
   **Answer: B) [2, 5, 1, 8, 9]** (5 and 2 swap, then 5 and 8 stay, then 8 and 1 swap, then 8 and 9 stay - largest element '9' moves to end.)

**Hashing & Introduction to Hash Tables**

1. What is the purpose of the hashCode() method in Java when using objects as keys in a HashMap?  
   A) To define equality between objects.  
   B) To provide a unique memory address for the object.  
   C) To compute an integer hash value that can be mapped to a bucket index.  
   D) To convert the object into a string representation.  
   **Answer: C) To compute an integer hash value that can be mapped to a bucket index.**
2. Why is it recommended that if you override equals() in Java, you must also override hashCode()?  
   A) To improve compile-time performance.  
   B) To ensure that equal objects have the same hash code, as required by HashMap/HashSet.  
   C) To prevent StackOverflowError.  
   D) To allow objects to be sorted.  
   **Answer: B) To ensure that equal objects have the same hash code, as required by HashMap/HashSet.**
3. In separate chaining, if multiple keys hash to the same index, they are typically stored in a:  
   A) Separate hash table.  
   B) Fixed-size array at that index.  
   C) Linked List (or potentially a tree for large chains) at that index.  
   D) The next available empty slot in the main array.  
   **Answer: C) Linked List (or potentially a tree for large chains) at that index.**
4. "Secondary Clustering" is a problem associated with which open addressing collision resolution strategy?  
   A) Linear Probing  
   B) Quadratic Probing  
   C) Double Hashing  
   D) Separate Chaining  
   **Answer: B) Quadratic Probing**
5. Which collision resolution technique is generally considered to produce the most diverse probing sequences?  
   A) Linear Probing  
   B) Quadratic Probing  
   C) Double Hashing  
   D) Separate Chaining  
   **Answer: C) Double Hashing**
6. What is a "load factor" in the context of hash tables?  
   A) The average length of a linked list in separate chaining.  
   B) The ratio of the number of elements in the table to the table's size (n/M).  
   C) The number of collisions that have occurred.  
   D) The maximum capacity of the hash table.  
   **Answer: B) The ratio of the number of elements in the table to the table's size (n/M).**
7. A hash function that frequently produces the same hash value for different keys is considered:  
   A) Efficient  
   B) Uniformly distributed  
   C) Collision-resistant  
   D) Poor (or bad)  
   **Answer: D) Poor (or bad)**

**Graphs**

1. A "Weighted Graph" is a graph where:  
   A) All edges are directed.  
   B) Each vertex has a numerical value.  
   C) Each edge has an associated numerical value (cost/distance).  
   D) The graph is very large.  
   **Answer: C) Each edge has an associated numerical value (cost/distance).**
2. A "Connected Component" in an undirected graph refers to:  
   A) A single vertex.  
   B) A subgraph where there is a path between any two vertices within the subgraph.  
   C) A cycle.  
   D) A part of the graph that cannot be traversed.  
   **Answer: B) A subgraph where there is a path between any two vertices within the subgraph.**
3. What is the space complexity of an Adjacency List representation for a graph with V vertices and E edges?  
   A) O(V^2)  
   B) O(V + E)  
   C) O(E log V)  
   D) O(1)  
   **Answer: B) O(V + E)**
4. If you need to detect cycles in a directed graph, which traversal algorithm is commonly used?  
   A) Breadth-First Search (BFS)  
   B) Depth-First Search (DFS)  
   C) Dijkstra's Algorithm  
   D) Prim's Algorithm  
   **Answer: B) Depth-First Search (DFS)**
5. What property must a graph have for Dijkstra's Algorithm to guarantee the correct shortest path?  
   A) It must be a Directed Acyclic Graph (DAG).  
   B) All edge weights must be positive (non-negative).  
   C) It must be a complete graph.  
   D) It must have no cycles.  
   **Answer: B) All edge weights must be positive (non-negative).**
6. A "Directed Acyclic Graph (DAG)" is often used for problems involving:  
   A) Finding cycles.  
   B) Task scheduling and dependency resolution.  
   C) Road networks with two-way streets.  
   D) Calculating minimum spanning trees.  
   **Answer: B) Task scheduling and dependency resolution.**
7. What is the primary difference in approach between Prim's and Kruskal's algorithms for MST?  
   A) Prim's is for directed graphs, Kruskal's for undirected.  
   B) Prim's builds the tree by adding vertices, Kruskal's by adding edges.  
   C) Prim's uses a queue, Kruskal's uses a stack.  
   D) Prim's handles negative weights, Kruskal's does not.  
   **Answer: B) Prim's builds the tree by adding vertices, Kruskal's by adding edges.**
8. The "Traveling Salesperson Problem (TSP)" is a classic example of an optimization problem often tackled with which advanced algorithm technique (among others)?  
   A) Greedy Algorithm  
   B) Dynamic Programming (for smaller instances) / Branch-and-Bound (for larger instances)  
   C) Divide and Conquer  
   D) Brute Force (naive)  
   **Answer: B) Dynamic Programming (for smaller instances) / Branch-and-Bound (for larger instances)** (The context implies optimization algorithms, where B&B is explicitly mentioned for TSP).
9. What is the 'degree of a vertex' in an undirected graph?  
   A) The number of incoming edges.  
   B) The number of outgoing edges.  
   C) The total number of edges incident to that vertex.  
   D) Its position in the adjacency list.  
   **Answer: C) The total number of edges incident to that vertex.**
10. If you are implementing a graph and need to quickly iterate over all neighbors of a specific vertex, which representation is generally more efficient for this task?  
    A) Adjacency Matrix  
    B) Adjacency List  
    C) Implicit Representation  
    D) Edge List  
    **Answer: B) Adjacency List**
11. Which traversal method uses an explicit (or implicit via recursion) stack data structure?  
    A) Level Order Traversal (BFS)  
    B) Breadth-First Search (BFS)  
    C) Depth-First Search (DFS)  
    D) Dijkstra's Algorithm  
    **Answer: C) Depth-First Search (DFS)**
12. The main limitation of Dijkstra's algorithm is that it cannot:  
    A) Find paths in disconnected graphs.  
    B) Handle graphs with cycles.  
    C) Work with negative edge weights.  
    D) Find all-pairs shortest paths.  
    **Answer: C) Work with negative edge weights.**
13. What is the time complexity of the Floyd-Warshall algorithm?  
    A) O(V^2)  
    B) O(E log V)  
    C) O(V \* E)  
    D) O(V^3)  
    **Answer: D) O(V^3)**
14. In Kruskal's algorithm, what data structure is crucial for efficiently detecting cycles?  
    A) Priority Queue  
    B) Stack  
    C) Disjoint Set Union (DSU)  
    D) Hash Table  
    **Answer: C) Disjoint Set Union (DSU)**
15. What is the fundamental property of a Spanning Tree?  
    A) It contains cycles.  
    B) It connects all vertices of the graph using the minimum number of edges possible without forming a cycle.  
    C) It is always a complete graph.  
    D) Its edges are always directed.  
    **Answer: B) It connects all vertices of the graph using the minimum number of edges possible without forming a cycle.**
16. Which type of graph is characterized by vertices being divided into two disjoint sets, with all edges connecting a vertex from one set to a vertex in the other?  
    A) Complete Graph  
    B) Connected Graph  
    C) Bipartite Graph  
    D) Sparse Graph  
    **Answer: C) Bipartite Graph**
17. A graph where E (number of edges) is much less than V^2 (number of vertices squared) is typically called a:  
    A) Dense Graph  
    B) Complete Graph  
    C) Sparse Graph  
    D) Bipartite Graph  
    **Answer: C) Sparse Graph**
18. What does visited array (or set) serve for in graph traversal algorithms like BFS or DFS?  
    A) To store the shortest path distances.  
    B) To keep track of the order of processing.  
    C) To prevent revisiting nodes and avoid infinite loops in graphs with cycles.  
    D) To count the total number of edges.  
    **Answer: C) To prevent revisiting nodes and avoid infinite loops in graphs with cycles.**

**Analysis of an Algorithm & Asymptotic Analysis**

1. Asymptotic analysis primarily describes an algorithm's performance as:  
   A) Its exact execution time on a specific machine.  
   B) The input size grows infinitely large.  
   C) The number of lines of code increases.  
   D) The memory usage remains constant.  
   **Answer: B) The input size grows infinitely large.**
2. An algorithm with time complexity O(n log n) is often referred to as:  
   A) Quadratic time  
   B) Exponential time  
   C) Linearithmic time  
   D) Logarithmic time  
   **Answer: C) Linearithmic time**
3. What is the Big Omega (Ω) notation used to describe?  
   A) The exact execution time of an algorithm.  
   B) The tight bound of an algorithm's running time.  
   C) The lower bound of an algorithm's running time.  
   D) The average case performance.  
   **Answer: C) The lower bound of an algorithm's running time.**
4. Counting basic operations (comparisons, assignments, arithmetic operations) is a key part of:  
   A) Debugging an algorithm.  
   B) Optimizing CPU clock speed.  
   C) Algorithm analysis.  
   D) Choosing the programming language.  
   **Answer: C) Algorithm analysis.**

**Analysis of Different Types of Algorithms**

1. Which example is explicitly listed as a problem typically solved with a Brute Force approach?  
   A) Binary Search  
   B) Longest Common Subsequence  
   C) Trying all permutations for a password  
   D) Merge Sort  
   **Answer: C) Trying all permutations for a password**
2. The concept of "optimal substructure" and "overlapping subproblems" are fundamental to which algorithmic paradigm?  
   A) Divide and Conquer  
   B) Greedy Algorithm  
   C) Dynamic Programming  
   D) Backtracking  
   **Answer: C) Dynamic Programming**
3. Which algorithm paradigm is suitable for solving the Sudoku puzzle?  
   A) Dynamic Programming  
   B) Greedy Algorithm  
   C) Backtracking  
   D) Stochastic Algorithm  
   **Answer: C) Backtracking**
4. Which algorithm, specifically mentioned as a Greedy Algorithm, is used for finding Minimum Spanning Trees?  
   A) Dijkstra's Algorithm  
   B) Floyd-Warshall Algorithm  
   C) Prim's Algorithm  
   D) Bellman-Ford Algorithm  
   **Answer: C) Prim's Algorithm**
5. What is the primary characteristic of a Stochastic Algorithm?  
   A) It always guarantees the globally optimal solution.  
   B) It incorporates randomness into its logic.  
   C) It breaks problems into overlapping subproblems.  
   D) It performs an exhaustive search.  
   **Answer: B) It incorporates randomness into its logic.**

**Complexity & Application of Data Structures**

1. In the context of space complexity in Java, what contributes to the auxiliary space?  
   A) Only the input data itself.  
   B) Memory used by objects, variables, and the recursion stack.  
   C) The size of the compiled .class file.  
   D) The total hard disk space available.  
   **Answer: B) Memory used by objects, variables, and the recursion stack.**
2. Why is choosing the "right" data structure crucial for algorithm performance?  
   A) All data structures have the same time complexity.  
   B) Different data structures are optimized for different types of operations.  
   C) Data structures only affect memory usage, not time.  
   D) Data structures are only relevant in low-level programming.  
   **Answer: B) Different data structures are optimized for different types of operations.**

**Introductory Concepts & Basic Data Structures**

1. What is the characteristic of java.util.ArrayList regarding its size?  
   A) Fixed size, cannot be changed after creation.  
   B) Dynamic size, can grow or shrink as needed.  
   C) Size must be a prime number.  
   D) Size is determined by the operating system.  
   **Answer: B) Dynamic size, can grow or shrink as needed.**
2. What is a common application of a Stack related to a text editor?  
   A) Spell checking  
   B) Find and replace functionality  
   C) Undo/Redo functionality  
   D) File saving  
   **Answer: C) Undo/Redo functionality**
3. Which java.util.Queue method is analogous to enqueue() and adds an element to the rear of the queue?  
   A) poll()  
   B) peek()  
   C) offer()  
   D) remove()  
   **Answer: C) offer()**
4. A PriorityQueue ensures that elements are dequeued based on:  
   A) Their insertion order (FIFO).  
   B) Their removal order (LIFO).  
   C) Their assigned priority.  
   D) Their alphabetical order.  
   **Answer: C) Their assigned priority.**
5. What is the typical time complexity for offer() and poll() operations on a java.util.LinkedList when used as a Queue?  
   A) O(n)  
   B) O(log n)  
   C) O(1)  
   D) O(n log n)  
   **Answer: C) O(1)**
6. Which of the following is NOT a standard operation for a Stack ADT?  
   A) push()  
   B) pop()  
   C) remove(index)  
   D) peek()  
   **Answer: C) remove(index)**

**Linked Lists**

1. What is the time complexity to delete a node by value in an unsorted Singly Linked List, assuming the list needs to be traversed to find the node?  
   A) O(1)  
   B) O(log n)  
   C) O(n)  
   D) O(n^2)  
   **Answer: C) O(n)**
2. The presence of a tail pointer in a Doubly Linked List allows for O(1) operations for:  
   A) Searching by index.  
   B) Insertion at the head.  
   C) Insertion/Deletion at the tail.  
   D) Random access.  
   **Answer: C) Insertion/Deletion at the tail.**
3. A potential problem with Circular Linked Lists if not handled carefully during traversal is:  
   A) Memory leaks.  
   B) Inefficient memory usage.  
   C) Infinite loops.  
   D) Data corruption.  
   **Answer: C) Infinite loops.**
4. What does the data field of a Node in a Linked List store?  
   A) A pointer to the next node.  
   B) The actual value or object stored in that part of the list.  
   C) The memory address of the node.  
   D) The index of the node in the list.  
   **Answer: B) The actual value or object stored in that part of the list.**

**Trees**

1. A "Subtree" is defined as:  
   A) Any branch of the tree that doesn't include the root.  
   B) A node and all its descendants.  
   C) A tree that is part of a larger forest.  
   D) A tree with only one level.  
   **Answer: B) A node and all its descendants.**
2. The "Root" of a tree is unique because:  
   A) It has the highest value.  
   B) It has no children.  
   C) It has no parent.  
   D) It is always a leaf node.  
   **Answer: C) It has no parent.**
3. A "Skewed Binary Tree" has a structure similar to a:  
   A) Balanced Binary Search Tree  
   B) Linked List  
   C) Complete Binary Tree  
   D) Full Binary Tree  
   **Answer: B) Linked List**
4. In a Binary Search Tree, what is the relationship between a node's value and the values in its right subtree?  
   A) Values in the right subtree are less than the node's value.  
   B) Values in the right subtree are equal to the node's value.  
   C) Values in the right subtree are greater than the node's value.  
   D) There is no specific relationship.  
   **Answer: C) Values in the right subtree are greater than the node's value.**
5. Which Java java.util class implements the TreeSet (a sorted set) internally using a Red-Black Tree?  
   A) HashMap  
   B) ArrayList  
   C) LinkedList  
   D) TreeMap (HashSet uses HashMap; TreeSet uses TreeMap)  
   **Answer: D) TreeMap (internally, TreeSet uses a TreeMap where elements are keys and dummy values are used)**
6. The primary reason to use a "Search Tree" is to facilitate:  
   A) Efficient sequential storage.  
   B) Efficient retrieval of elements.  
   C) Parallel processing.  
   D) Direct memory access.  
   **Answer: B) Efficient retrieval of elements.**
7. What is the correct order of processing nodes in a Post-order Traversal?  
   A) Root, Left, Right  
   B) Left, Root, Right  
   C) Left, Right, Root  
   D) Right, Left, Root  
   **Answer: C) Left, Right, Root**

**Searching & Sorting**

1. What is the time complexity of Sequential Search in its best case?  
   A) O(n)  
   B) O(log n)  
   C) O(1)  
   D) O(n^2)  
   **Answer: C) O(1)**
2. Which sorting algorithm's time complexity is O(n^2) in all cases (best, average, worst)?  
   A) Insertion Sort  
   B) Quick Sort  
   C) Selection Sort  
   D) Merge Sort  
   **Answer: C) Selection Sort**
3. Which of the following statements is true about Bubble Sort?  
   A) It is the most efficient sorting algorithm.  
   B) It makes repeated passes, comparing and swapping adjacent elements.  
   C) It is unstable.  
   D) Its best-case time complexity is O(1).  
   **Answer: B) It makes repeated passes, comparing and swapping adjacent elements.**
4. Heap Sort relies on what property of the Heap data structure?  
   A) All elements are sorted after heap creation.  
   B) The maximum (or minimum) element is always at the root.  
   C) It can be implemented with a linked list.  
   D) It guarantees stability.  
   **Answer: B) The maximum (or minimum) element is always at the root.**
5. What distinguishes Merge Sort from Quick Sort concerning auxiliary space?  
   A) Merge Sort is in-place, Quick Sort is not.  
   B) Merge Sort requires O(n) auxiliary space, while Quick Sort is mostly in-place (O(log n) for recursion stack).  
   C) Quick Sort is always O(1) auxiliary space.  
   D) Both require O(n^2) auxiliary space.  
   **Answer: B) Merge Sort requires O(n) auxiliary space, while Quick Sort is mostly in-place (O(log n) for recursion stack).**
6. What is the primary objective of sorting data?  
   A) To minimize memory usage.  
   B) To enable efficient searching and improve data processing.  
   C) To encrypt the data.  
   D) To remove duplicate elements.  
   **Answer: B) To enable efficient searching and improve data processing.**

**Hashing & Introduction to Hash Tables**

1. What is one of the key qualities of a good hash function?  
   A) It should produce a different hash value for the same input each time.  
   B) It should be computationally expensive.  
   C) It should distribute keys uniformly across the hash table.  
   D) It should produce large hash values.  
   **Answer: C) It should distribute keys uniformly across the hash table.**
2. If a hash table uses M as its size, and the hash function h(key) produces a large integer, what is the common way to map it to an index 0...M-1?  
   A) index = h(key) + M  
   B) index = h(key) \* M  
   C) index = h(key) % M  
   D) index = M - h(key)  
   **Answer: C) index = h(key) % M**
3. What is the main idea behind "Open Addressing" for collision resolution?  
   A) Each slot in the array stores a linked list of elements.  
   B) Elements are stored directly in the hash table array itself, probing for an empty slot on collision.  
   C) Elements are stored in a separate overflow area.  
   D) The hash table automatically resizes to avoid collisions.  
   **Answer: B) Elements are stored directly in the hash table array itself, probing for an empty slot on collision.**
4. In the context of hash tables, what happens during "rehashing"?  
   A) The hash function is re-calculated for all existing keys.  
   B) The table's size is increased, and all existing elements are re-inserted into the new, larger table.  
   C) Collisions are resolved by deleting one of the colliding elements.  
   D) The load factor is forcibly decreased to zero.  
   **Answer: B) The table's size is increased, and all existing elements are re-inserted into the new, larger table.**
5. The problem of "secondary clustering" is associated with:  
   A) Linear Probing  
   B) Quadratic Probing  
   C) Double Hashing  
   D) Separate Chaining  
   **Answer: B) Quadratic Probing**
6. java.util.HashSet is internally implemented using which data structure?  
   A) ArrayList  
   B) LinkedList  
   C) HashMap  
   D) TreeMap  
   **Answer: C) HashMap**

**Graphs**

1. Which term refers to a graph whose vertices can be divided into two disjoint sets such that every edge connects a vertex in one set to one in the other?  
   A) Complete Graph  
   B) Connected Graph  
   C) Bipartite Graph  
   D) Acyclic Graph  
   **Answer: C) Bipartite Graph**
2. What does V represent in the context of graph representation space complexity (e.g., O(V^2))?  
   A) Number of edges  
   B) Number of vertices  
   C) Value of edges  
   D) Volume of memory  
   **Answer: B) Number of vertices**
3. If you need to determine if there's a path between any two vertices in an undirected graph, which algorithms are suitable?  
   A) Dijkstra's Algorithm only.  
   B) BFS or DFS.  
   C) Kruskal's Algorithm only.  
   D) Floyd-Warshall Algorithm only.  
   **Answer: B) BFS or DFS.**
4. What is a key characteristic of Depth-First Search (DFS) traversal?  
   A) It explores all neighbors at the current depth level before moving deeper.  
   B) It uses a queue data structure.  
   C) It explores as far as possible along each branch before backtracking.  
   D) It is primarily used for finding shortest paths in unweighted graphs.  
   **Answer: C) It explores as far as possible along each branch before backtracking.**
5. What is the time complexity of Prim's Algorithm when implemented using an adjacency matrix?  
   A) O(E log V)  
   B) O(V^2)  
   C) O(V + E)  
   D) O(V \* E)  
   **Answer: B) O(V^2)**
6. The Bellman-Ford algorithm is unique among single-source shortest path algorithms because it can:  
   A) Only work on unweighted graphs.  
   B) Detect the presence of negative cycles.  
   C) Find all-pairs shortest paths.  
   D) Only work on directed acyclic graphs.  
   **Answer: B) Detect the presence of negative cycles.**
7. A "Minimum Spanning Tree (MST)" is a spanning tree with:  
   A) The maximum number of edges.  
   B) The smallest sum of edge weights.  
   C) The largest sum of edge weights.  
   D) No cycles and no edges.  
   **Answer: B) The smallest sum of edge weights.**
8. What does "In-degree" refer to in a directed graph?  
   A) The number of edges originating from a vertex.  
   B) The number of edges connected to a vertex (incoming + outgoing).  
   C) The number of edges pointing into a vertex.  
   D) The total number of vertices in the graph.  
   **Answer: C) The number of edges pointing into a vertex.**
9. Which graph representation is more memory efficient for **sparse graphs**?  
   A) Adjacency Matrix  
   B) Adjacency List  
   C) Both are equally efficient.  
   D) Neither is efficient.  
   **Answer: B) Adjacency List**
10. The concept of "print spooling" (managing print jobs) is a classic real-world application of which data structure?  
    A) Stack  
    B) Linked List  
    C) Queue  
    D) Hash Table  
    **Answer: C) Queue**

**General Concepts / Mixed Review**

1. If an algorithm requires a stack for its operations, it implies that the problem might involve:  
   A) First-In, First-Out processing.  
   B) Backtracking or recursion.  
   C) Prioritized task execution.  
   D) Random access by index.  
   **Answer: B) Backtracking or recursion.**
2. Which term refers to the number of CPU instructions or function calls performed by an algorithm?  
   A) Space Complexity  
   B) Time Complexity  
   C) Memory Usage  
   D) Algorithm Efficiency  
   **Answer: B) Time Complexity**
3. What is the main advantage of using an Abstract Data Type (ADT) in design?  
   A) It provides the fastest implementation.  
   B) It hides implementation details, allowing for flexible underlying data structures.  
   C) It automatically handles memory management.  
   D) It defines exact execution times.  
   **Answer: B) It hides implementation details, allowing for flexible underlying data structures.**
4. Which of the following is an example of an **iteration** construct in Java?  
   A) if-else  
   B) switch  
   C) for loop  
   D) Method call  
   **Answer: C) for loop**
5. The O(1) time complexity for get() operation in an ArrayList is due to:  
   A) Its dynamic resizing.  
   B) Its underlying contiguous memory allocation, allowing direct address calculation.  
   C) Its use of a linked list internally.  
   D) Its ability to sort elements automatically.  
   **Answer: B) Its underlying contiguous memory allocation, allowing direct address calculation.**
6. In the context of Big O notation, O(n^2) is considered less efficient than O(n log n) because:  
   A) n log n grows faster than n^2.  
   B) n^2 has a lower constant factor.  
   C) n^2 grows at a much faster rate for large n.  
   D) n log n is always for small inputs.  
   **Answer: C) n^2 grows at a much faster rate for large n.**
7. The fundamental trade-off illustrated by comparing arrays and linked lists is usually between:  
   A) Ease of implementation vs. debugging.  
   B) Memory usage vs. CPU clock speed.  
   C) Fast random access/compact storage vs. efficient insertions/deletions/dynamic size.  
   D) Single-threaded vs. multi-threaded operations.  
   **Answer: C) Fast random access/compact storage vs. efficient insertions/deletions/dynamic size.**
8. What is the most common use case for a Binary Search Tree (BST)?  
   A) Storing data in no particular order.  
   B) Efficient searching, insertion, and deletion of ordered data.  
   C) Implementing a Last-In, First-Out queue.  
   D) Representing social network connections.  
   **Answer: B) Efficient searching, insertion, and deletion of ordered data.**
9. If a sorting algorithm is described as "stable," it means:  
   A) Its running time is consistent regardless of input.  
   B) It requires O(1) auxiliary space.  
   C) It maintains the relative order of elements with equal values.  
   D) It always produces the same output array.  
   **Answer: C) It maintains the relative order of elements with equal values.**
10. In a hash table, the quality of the hash function directly impacts:  
    A) The total memory available.  
    B) The worst-case time complexity.  
    C) The likelihood of collisions and overall performance.  
    D) The number of CPU cores used.  
    **Answer: C) The likelihood of collisions and overall performance.**

**Analysis of an Algorithm & Asymptotic Analysis**

1. An algorithm's Big O notation represents its:  
   A) Fastest possible running time.  
   B) Average running time on typical inputs.  
   C) Worst-case scenario or upper bound on resource usage.  
   D) Exact number of operations performed.  
   **Answer: C) Worst-case scenario or upper bound on resource usage.**
2. When we say an algorithm has O(1) time complexity, it implies that its execution time:  
   A) Is less than 1 millisecond.  
   B) Remains constant regardless of the input size.  
   C) Decreases logarithmically with input size.  
   D) Is dependent on the CPU's clock speed.  
   **Answer: B) Remains constant regardless of the input size.**
3. What is the main reason why analyzing an algorithm's exact execution time is impractical for comparing algorithms?  
   A) It requires special hardware.  
   B) It varies significantly based on hardware, compiler, and specific input.  
   C) It is too difficult to measure accurately.  
   D) It only applies to very small inputs.  
   **Answer: B) It varies significantly based on hardware, compiler, and specific input.**
4. If an algorithm is said to be Θ(n log n), it means:  
   A) Its running time is exactly n log n.  
   B) Its running time is bounded above by n log n.  
   C) Its running time is bounded below by n log n.  
   D) Its running time is both bounded above and below by functions proportional to n log n.  
   **Answer: D) Its running time is both bounded above and below by functions proportional to n log n.**

**Analysis of Different Types of Algorithms**

1. The "Fractional Knapsack Problem" is a classic problem often optimally solved by which algorithmic paradigm?  
   A) Dynamic Programming  
   B) Divide and Conquer  
   C) Greedy Algorithm  
   D) Brute Force  
   **Answer: C) Greedy Algorithm**
2. Which characteristic is common to both Divide and Conquer and Dynamic Programming?  
   A) They both always find the globally optimal solution using local choices.  
   B) They both break problems into smaller subproblems.  
   C) They both explore all possible solutions exhaustively.  
   D) They both require a random number generator.  
   **Answer: B) They both break problems into smaller subproblems.**
3. What is a key distinction between Dynamic Programming and Divide and Conquer?  
   A) DP is iterative, D&C is recursive.  
   B) DP deals with overlapping subproblems, D&C typically deals with independent subproblems.  
   C) DP is only for optimization, D&C is for decision problems.  
   D) DP is for weighted graphs, D&C is for unweighted.  
   **Answer: B) DP deals with overlapping subproblems, D&C typically deals with independent subproblems.**
4. Which algorithmic technique would you use if you needed to find *all* possible solutions to a problem, but could prune invalid paths early?  
   A) Greedy Algorithm  
   B) Dynamic Programming  
   C) Backtracking Algorithm  
   D) Stochastic Algorithm  
   **Answer: C) Backtracking Algorithm**
5. The Floyd-Warshall algorithm is a classic example of:  
   A) A Greedy algorithm.  
   B) A Divide and Conquer algorithm.  
   C) A Dynamic Programming algorithm.  
   D) A Stochastic algorithm.  
   **Answer: C) A Dynamic Programming algorithm.**

**Complexity & Application of Data Structures**

1. The phrase "trade-offs" in algorithm design typically refers to balancing:  
   A) Ease of coding versus debugging difficulty.  
   B) Time complexity versus space complexity.  
   C) Development cost versus project deadline.  
   D) High-level language versus low-level language.  
   **Answer: B) Time complexity versus space complexity.**
2. When comparing algorithms, complexity analysis is crucial because:  
   A) It helps determine the exact execution time on any machine.  
   B) It identifies the dominant resource usage as input scales, allowing for informed choices.  
   C) It ensures the algorithm is bug-free.  
   D) It guarantees the algorithm is easy to understand.  
   **Answer: B) It identifies the dominant resource usage as input scales, allowing for informed choices.**

**Introductory Concepts & Basic Data Structures**

1. An algorithm construct that allows a block of code to be repeated multiple times is called a:  
   A) Conditional statement.  
   B) Sequential statement.  
   C) Looping statement (Iteration).  
   D) Function call.  
   **Answer: C) Looping statement (Iteration).**
2. Which Java interface is an ADT that describes a collection of elements that can be iterated over in a specific sequence?  
   A) java.util.Set  
   B) java.util.Map  
   C) java.util.List  
   D) java.util.Optional  
   **Answer: C) java.util.List**
3. For an ArrayList in Java, which operation could potentially have a worst-case time complexity of O(n) due to resizing?  
   A) get(index)  
   B) remove(lastElement)  
   C) add(element)  
   D) size()  
   **Answer: C) add(element)** (When a resize is triggered, elements need to be copied).
4. Which of the following is a common use case for a Queue, often seen in operating systems?  
   A) Storing undo operations.  
   B) CPU scheduling and task management.  
   C) Balancing a binary tree.  
   D) Implementing a symbol table.  
   **Answer: B) CPU scheduling and task management.**
5. The element() method in the Queue interface is similar to peek(), but with one key difference:  
   A) element() removes the head, peek() does not.  
   B) element() returns null if empty, peek() throws an exception.  
   C) element() throws an exception if the queue is empty, peek() returns null.  
   D) element() adds an element, peek() removes one.  
   **Answer: C) element() throws an exception if the queue is empty, peek() returns null.**
6. What is the typical time complexity for peekMax() (retrieving the highest priority element without removing it) in a Priority Queue?  
   A) O(n)  
   B) O(log n)  
   C) O(1)  
   D) O(n log n)  
   **Answer: C) O(1)**
7. java.util.ArrayDeque is generally preferred over java.util.LinkedList for implementing stacks and queues because:  
   A) It provides better worst-case performance for all operations.  
   B) It is implemented with contiguous arrays, offering better cache performance and lower memory overhead per element.  
   C) It supports random access (O(1)) for elements.  
   D) It automatically handles synchronization.  
   **Answer: B) It is implemented with contiguous arrays, offering better cache performance and lower memory overhead per element.**

**Linked Lists**

1. What is the fundamental way a Linked List differs from an Array in terms of how elements are stored in memory?  
   A) Linked List elements are always sorted.  
   B) Linked List elements are stored in contiguous memory locations.  
   C) Linked List elements are not stored in contiguous memory, but linked by pointers.  
   D) Arrays store objects, Linked Lists store primitive types.  
   **Answer: C) Linked List elements are not stored in contiguous memory, but linked by pointers.**
2. The "Node-based storage with arrays" approach for a linked list might involve storing:  
   A) Direct memory addresses within the node.  
   B) Array indices as "pointers" to refer to the "next" node.  
   C) Only the data without any pointers.  
   D) A copy of the entire list within each node.  
   **Answer: B) Array indices as "pointers" to refer to the "next" node.**
3. A Circular Linked List is particularly useful for implementing:  
   A) Binary Search Trees.  
   B) Sequential search algorithms.  
   C) Circular buffers or round-robin scheduling.  
   D) Hash tables with separate chaining.  
   **Answer: C) Circular buffers or round-robin scheduling.**

**Trees**

1. Which tree traversal order would be suitable for creating a postfix expression from an expression tree?  
   A) Pre-order Traversal  
   B) In-order Traversal  
   C) Post-order Traversal  
   D) Level Order Traversal  
   **Answer: C) Post-order Traversal**
2. A "Complete Binary Tree" differs from a "Full Binary Tree" in that:  
   A) A Complete Binary Tree must have all leaves at the same depth.  
   B) A Full Binary Tree can have single children, while a Complete Binary Tree cannot.  
   C) All levels are completely filled except possibly the last in a Complete Binary Tree, with nodes as far left as possible.  
   D) A Complete Binary Tree always has 2^h - 1 nodes.  
   **Answer: C) All levels are completely filled except possibly the last in a Complete Binary Tree, with nodes as far left as possible.**
3. The purpose of java.util.TreeMap is to:  
   A) Store elements in a hash table.  
   B) Provide a sorted map implementation based on a Red-Black Tree.  
   C) Implement a fixed-size array.  
   D) Offer fast random access to elements by index.  
   **Answer: B) Provide a sorted map implementation based on a Red-Black Tree.**
4. If you perform an in-order traversal on a Binary Search Tree, what kind of output do you get?  
   A) Random order.  
   B) Elements in ascending sorted order.  
   C) Elements in descending sorted order.  
   D) The order in which elements were inserted.  
   **Answer: B) Elements in ascending sorted order.**
5. What is the 'depth' of the root node in a tree?  
   A) 1  
   B) 0  
   C) -1  
   D) V (number of vertices)  
   **Answer: B) 0**

**Searching & Sorting**

1. What is the time complexity of a recursive Binary Search in terms of space?  
   A) O(1)  
   B) O(log n) (due to recursion stack)  
   C) O(n)  
   D) O(n log n)  
   **Answer: B) O(log n)**
2. Which sorting algorithm is considered highly parallelizable, making it suitable for multi-core processors?  
   A) Bubble Sort  
   B) Selection Sort  
   C) Merge Sort  
   D) Insertion Sort  
   **Answer: C) Merge Sort**
3. What is a key characteristic that makes Insertion Sort efficient for "nearly sorted" arrays?  
   A) It performs many swaps.  
   B) It stops comparing elements once a correctly placed element is found.  
   C) It uses recursion.  
   D) It divides the array into sub-arrays.  
   **Answer: B) It stops comparing elements once a correctly placed element is found.**
4. Which sorting algorithm has a worst-case time complexity of O(n^2) but is generally the fastest in practice for many datasets?  
   A) Heap Sort  
   B) Merge Sort  
   C) Quick Sort  
   D) Selection Sort  
   **Answer: C) Quick Sort**
5. The ability of a sorting algorithm to preserve the relative order of equal elements is known as:  
   A) In-place property.  
   B) Time complexity.  
   C) Stability.  
   D) Efficiency.  
   **Answer: C) Stability.**
6. If an array is completely reverse-sorted, which sorting algorithm would perform poorly (worst-case O(n^2)) due to its comparison and shift nature?  
   A) Merge Sort  
   B) Heap Sort  
   C) Insertion Sort  
   D) Quick Sort (if pivot selection is poor)  
   **Answer: C) Insertion Sort**

**Hashing & Introduction to Hash Tables**

1. A good hash function is described as "deterministic," meaning:  
   A) It generates random hash values.  
   B) It always produces the same hash value for the same input key.  
   C) It can be optimized for specific hardware.  
   D) It can predict future hash values.  
   **Answer: B) It always produces the same hash value for the same input key.**
2. In separate chaining, if a hash table has a high load factor, what is the consequence?  
   A) The table will resize automatically.  
   B) The average length of the linked lists (chains) will increase, degrading performance.  
   C) There will be fewer collisions.  
   D) The space complexity will become O(1).  
   **Answer: B) The average length of the linked lists (chains) will increase, degrading performance.**
3. The primary benefit of using a prime number for the size M of a hash table's array (when using modulo for mapping) is:  
   A) It guarantees O(1) performance.  
   B) It minimizes memory usage.  
   C) It helps distribute hash values more evenly, reducing collisions.  
   D) It simplifies the hash function calculation.  
   **Answer: C) It helps distribute hash values more evenly, reducing collisions.**
4. Which collision resolution strategy avoids clustering issues but might require finding a new available slot with more computation?  
   A) Linear Probing  
   B) Separate Chaining  
   C) Double Hashing  
   D) Open Addressing (as a general category)  
   **Answer: C) Double Hashing**
5. If the equals() method of a custom object used as a HashMap key is incorrectly implemented (e.g., two "equal" objects return false), what can happen?  
   A) The HashMap will automatically correct it.  
   B) Elements may not be retrieved correctly, as the map might store duplicates or fail to find existing entries.  
   C) The hashCode() method will be ignored.  
   D) The application will crash immediately.  
   **Answer: B) Elements may not be retrieved correctly, as the map might store duplicates or fail to find existing entries.**

**Graphs**

1. What is the fundamental difference between a "path" and a "cycle" in a graph?  
   A) A path has no repeated vertices, while a cycle starts and ends at the same vertex.  
   B) A path can only exist in directed graphs, a cycle in undirected.  
   C) A path is always weighted, a cycle is unweighted.  
   D) A path is always shorter than a cycle.  
   **Answer: A) A path has no repeated vertices, while a cycle starts and ends at the same vertex.**
2. The application of graph theory to "social networks" primarily represents:  
   A) Vertices as posts and edges as likes.  
   B) Vertices as individuals and edges as relationships/connections.  
   C) Vertices as comments and edges as replies.  
   D) Vertices as images and edges as colors.  
   **Answer: B) Vertices as individuals and edges as relationships/connections.**
3. Which graph representation uses a 2D array and is best for **dense graphs** where checking for edge existence is a frequent operation?  
   A) Adjacency List  
   B) Adjacency Matrix  
   C) Incidence Matrix  
   D) Edge List  
   **Answer: B) Adjacency Matrix**
4. To find all connected components in an undirected graph, which traversal algorithm(s) can be effectively used?  
   A) Only Dijkstra's.  
   B) Only Floyd-Warshall.  
   C) Both BFS and DFS.  
   D) Only Kruskal's.  
   **Answer: C) Both BFS and DFS.**
5. What is a key property of a Directed Acyclic Graph (DAG)?  
   A) All vertices have an in-degree of 0.  
   B) It contains at least one cycle.  
   C) It is a directed graph that contains no cycles.  
   D) All edges are weighted.  
   **Answer: C) It is a directed graph that contains no cycles.**
6. The time complexity of Bellman-Ford algorithm is O(V\*E). This makes it generally less efficient than Dijkstra's for graphs with non-negative weights, but compensates by:  
   A) Being simpler to implement.  
   B) Handling graphs with negative edge weights.  
   C) Requiring less memory.  
   D) Finding all-pairs shortest paths.  
   **Answer: B) Handling graphs with negative edge weights.**
7. The purpose of a "Spanning Tree" is to:  
   A) Find the longest path in a graph.  
   B) Connect all vertices of a graph with the minimum possible number of edges while avoiding cycles.  
   C) Determine if a graph is directed or undirected.  
   D) Calculate the shortest path between two specific vertices.  
   **Answer: B) Connect all vertices of a graph with the minimum possible number of edges while avoiding cycles.**
8. What is the core idea of Prim's Algorithm for MST?  
   A) It sorts all edges and adds them if they don't form a cycle.  
   B) It grows the MST by iteratively adding the cheapest edge that connects a vertex in the growing MST to a vertex outside it.  
   C) It finds the shortest path from a source vertex to all others.  
   D) It uses dynamic programming to find all-pairs shortest paths.  
   **Answer: B) It grows the MST by iteratively adding the cheapest edge that connects a vertex in the growing MST to a vertex outside it.**
9. In a graph, what is the significance of "out-degree" for a vertex?  
   A) The number of edges terminating at that vertex.  
   B) The total number of edges connected to that vertex.  
   C) The number of edges originating from that vertex.  
   D) The number of paths leading to that vertex.  
   **Answer: C) The number of edges originating from that vertex.**
10. A graph is said to be "Disconnected" if:  
    A) It contains cycles.  
    B) All vertices have a degree of 0.  
    C) There is no path between every pair of vertices.  
    D) It has only one vertex.  
    **Answer: C) There is no path between every pair of vertices.**
11. Which graph traversal algorithm is implicitly used when solving mazes or exploring all reachable nodes from a starting point, going as deep as possible before backtracking?  
    A) BFS  
    B) Dijkstra's  
    C) DFS  
    D) Prim's  
    **Answer: C) DFS**
12. Which shortest path algorithm's standard implementation uses a PriorityQueue to manage distances to unvisited nodes?  
    A) Bellman-Ford  
    B) Floyd-Warshall  
    C) Dijkstra's  
    D) BFS  
    **Answer: C) Dijkstra's**
13. What is the primary use case for the Floyd-Warshall algorithm?  
    A) Single-source shortest path in unweighted graphs.  
    B) Single-source shortest path with negative cycles.  
    C) All-pairs shortest path in weighted graphs (can have negative weights, but no negative cycles).  
    D) Minimum spanning tree.  
    **Answer: C) All-pairs shortest path in weighted graphs (can have negative weights, but no negative cycles).**