I. Fundamental Concepts

Mark the following statements as True or False. Provide a concise explanation. 5 points

a) A recursive algorithm always uses more memory than its iterative equivalent.

b) Breadth-first search (BFS) can be used to find the shortest path in weighted graphs where all edges have equal weight.

c) The time complexity of binary search on any array is O(log n).

II. Code Output

Analyze and trace the following code snippets. What is the output? Provide a concise explanation 8 points

```java
List<String> algorithms = new ArrayList<>();
algorithms.add("DFS");
algorithms.add("BFS");
algorithms.add("Dijkstra");

Stack<String> stack = new Stack<>();
for (String alg : algorithms) {
    stack.push(alg);
}
stack.pop();
stack.push("A*");

while (!stack.isEmpty()) {
    System.out.print(stack.pop() + " ");
}
```

```java
Queue<String> queue = new LinkedList<>();
queue.offer("Array");
queue.offer("LinkedList");
queue.offer("Heap");

Stack<String> stack = new Stack<>();
while (!queue.isEmpty()) {
    String ds = queue.poll();
    if (!ds.equals("LinkedList")) {
        stack.push(ds);
    }
}

while (!stack.isEmpty()) {
    System.out.print(stack.pop() + " ");
}
```

III. Fundamentals of Trees and Graphs

3. 1 Trees 4 points

Consider the following tree, and briefly answer the questions

```
              Root
            /      \
        Alpha       Beta
       /    \           \
    Gamma  Delta         Omega
```

- List the nodes visited in a *preorder* traversal.

- Suppose you delete a node Alpha. Describe how the BST property is maintained after deletion. What do we replace the deleted node with?

Logical Reasoning

- Consider the statement: "*in any binary tree, the number of nodes is always greater than the height*." Justify with a counterexample or explanation.

3.2 Graphs

4 points

Consider this graph

```
[A]-----3-----[B]
 |             |
 2             4
 |             |
[D]-----1-----[C]
 |
 5
[E]
```

- Represent the graph using an adjacency list

- What is the degree of node A?

- List the order of nodes visited in Depth-First Search (DFS) starting from node A.

- What is the main difference between Dijkstra’s and A in terms of node selection during traversal? Explain using as illustration this graph.

IV. Key Algorithms 15 points

Consider Dijkstra’s algorithm presented below. Complete the missing parts and concisely motivate your answer

```
function dijkstra(graph, source):
    dist = _________
    dist[source] = _________

    visited = _________
    priorityQueue = _________
    insert (0, source) into priorityQueue

    while _________:
        (currentDist, u) = _________

        if u in visited:
            continue

        add u to visited

        for each neighbor v of u:   // complete part below




    return dist
```

- What is the purpose of the priority queue in Dijkstra’s algorithm?

- When is a node considered “visited” or finalized in Dijkstra’s algorithm?

- What is the time complexity of Dijkstra's algorithm?

V. Data Structure Selection for Use Cases — 6 points

A text editor needs to support undo and redo operations efficiently, where users can revert their last actions or reapply undone actions in order. The operations should be performed quickly without scanning the entire history.

Which data structure(s) would you use to implement this functionality? Explain your choice.

VI. Coding Challenge — 8 points

Consider the following problem and highlight a model solution using either Java code or pseudocode.

Given the root of a binary tree, determine if it is a valid Binary Search Tree (BST). A BST must satisfy the property that left subtree nodes < node < right subtree nodes for every node.

- Implement `boolean isValidBST(TreeNode root)`
- Think carefully about edge cases like duplicate values or null nodes.