Depth-First Search is an algorithm for traversing or searching a directed graph. It explores as deeply as possible along each branch before backtracking. It can be implemented either using recursion or iterative (a loop without recursion) using a stack.

Here's the iterative version of the algorithm:

Initialization:

1. Create a stack to store nodes to visit.
2. Create a Boolean array called visited to track visited nodes (initially all false).
3. Choose a starting node.
4. Push the starting node onto the stack.

Iteration:

5. While the stack is not empty:
    6. Pop a node from the stack
    7. If the node has not been visited:
        8. Mark the node as visited.
        9. Process the node (e.g., print its value).
        10. For each neighbor of the current node:
            11. If the neighbor has not been visited, push it onto the stack.

Termination:

The algorithm terminates when the stack is empty, indicating that all reachable nodes have been explored.

Breath-First Search is an algorithm for finding the shortest paths in an unweighted graphs from a starting node to all other reachable nodes. It explores a graph level by level, starting from the root node. It systematically visits all nodes at the current depth before moving to the next depth level. BFS uses a queue to keep track of the nodes to be visited, ensuring a breadth-first exploration.

Here's the iterative version of the algorithm:

Initialization:

1. Create a queue to store nodes to visit.
2. Create a Boolean array called visited to track visited nodes (initially all false).
3. Choose a starting node.
4. Enqueue the starting node to the queue.

Iteration:

5. While the queue is not empty:
    6. Dequeue a node from the queue
    7. For each unvisited neighbor of the dequeue node:
        8. Mark the neighbor as visited.
        9. Process the node (e.g., print its value).
        10. Enqueue the neighbor to the queue.

Termination:

The algorithm terminates when the queue is empty, indicating that all reachable nodes have been explored.