DFS and BFS Search (Iterative & Recursive)

Algorithm	Best Case	Average Case	Worst Case
DFS (Iterative & Recursive)	O(1)	O(V + E)	O(V + E)
BFS (Iterative & Recursive)	O(1)	O(V + E)	O(V + E)

Python Code for DFS & BFS Search (Iterative & Recursive)

```python
from collections import deque

# Depth-First Search (DFS) - Iterative
def dfs_iterative(graph, start, target):
    stack = [start]
    visited = set()

    while stack:
        node = stack.pop()
        if node == target:
            return True
        if node not in visited:
            visited.add(node)
            stack.extend(graph.get(node, []))
    return False
    # Time Complexity: O(V + E)

# Depth-First Search (DFS) - Recursive
def dfs_recursive(graph, node, target, visited=None):
    if visited is None:
        visited = set()
    if node == target:
        return True
    if node in visited:
        return False

    visited.add(node)
    for neighbor in graph.get(node, []):
        if dfs_recursive(graph, neighbor, target, visited):
            return True
    return False
    # Time Complexity: O(V + E)

# Breadth-First Search (BFS) - Iterative
def bfs_iterative(graph, start, target):
    queue = deque([start])
    visited = set()

    while queue:
        node = queue.popleft()
        if node == target:
            return True
        if node not in visited:
            visited.add(node)
            queue.extend(graph.get(node, []))
    return False
    # Time Complexity: O(V + E)

# Breadth-First Search (BFS) - Recursive
def bfs_recursive(graph, queue, target, visited=None):
    if not queue:
        return False

    if visited is None:
        visited = set()

    node = queue.popleft()
    if node == target:
        return True
    if node not in visited:
        visited.add(node)
        queue.extend(graph.get(node, []))

    return bfs_recursive(graph, queue, target, visited)
    # Time Complexity: O(V + E)
```