BFS and DFS: Time and Space Complexities

Breadth-First Search (BFS):

=> Explores the graph level by level starting from a source node.

=> Visits all immediate neighbors first, then their neighbors, and so on.

=> Useful for finding the shortest path (unweighted graphs).

Depth-First search (DFS):

=> Explores the graph as deeply as possible along one branch before backtracking.

=> Visits one neighbor, then recursively its neighbor, until no further nodes can be visited.

=> Useful for path finding, cycle detection, and topological sorting.

Data Structures Used:

- BFS: Uses a Queue [FIFO].
- DFS: Uses a Stack [LIFO]. Can be implemented with:
  - => Explicit stack (iterative DFS), or
  - => Implicit stack (recursive DFS).

Time Complexity

Let: N = Number of nodes (vertices)
E = Number of edges

Both BFS and DFS must visit all nodes and explore all edges at most once.

- Adjacency List Representation:
  - BFS = $O(N+E)$
  - DFS = $O(N+E)$
- Adjacency Matrix Representation:
  - BFS = $O(N^2)$
  - DFS = $O(N^2)$ [∵ because each node checks all possible edges]

Space Complexity

⇒ BFS:
- Needs space for queue storing up to $O(N)$ nodes in the worst case.
- Also stores visited list → $O(N)$.
- Total = $O(N)$

⇒ DFS:
- Needs space for stack/recursion depth, which in the worst case is $O(N)$.
- Also stores visited list → $O(N)$.
- Total = $O(N)$

Sparse vs Dense Graph:

- Sparse Graph ($E \approx N$):
  - Complexity ~ $O(N)$
- Dense Graph ($E \approx N^2$):
  - Complexity ~ $O(N^2)$

⇒ So Performance depends on graph density and representation

Assumptions:

=> Graph is connected (otherwise, BFS/DFS must be repeated for each connected component).

=> Graph is stored as an adjacency list unless stated otherwise.