BFS and DFS

A short explanation of the Breadth-First Search and Depth-First Search Algorithm

Breadth-First Search (BFS)

Purpose: BFS is used to traverse or search a graph in a level-wise order.

How it Works:

Start from a selected node (source or root in a tree).

Explore all the neighbour nodes at the present depth before moving on to the nodes at the next depth level.

Queue Usage: BFS uses a queue to keep track of the next location to visit.

Properties:

Finds the shortest path in terms of the number of edges from the source to all other nodes in an unweighted graph.

Time Complexity: O(V+E), where V is vertices and E is edges.

Depth-First Search (DFS)

Purpose: DFS is used to traverse or search a graph by going as far along each branch before backtracking.

How it Works:

Start from a selected node (source or root in a tree).

Explore as far as possible along each branch before backtracking.

Stack Usage: DFS uses a stack (can be implemented with recursion) to keep track of the next location to visit.

Properties:

Useful for checking connectivity and finding components in a graph.

Time Complexity: O(V+E), similar to BFS.

Key Differences

Strategy: BFS uses a queue and explores neighbours first, while DFS uses a stack (or recursion) and explores depth first.

Applications:

BFS is used in shortest path algorithms and for level-order tree traversal.

DFS is used in topological sorting, scheduling problems, cycle detection in graphs, and solving puzzles with only one solution, like mazes.

Space Complexity: DFS can be more space efficient than BFS, especially on sparse graphs.

Example Applications

BFS: Finding the shortest path in a graph, peer-to-peer networking, crawlers in search engines, GPS navigation systems.

DFS: Topological sorting, solving puzzles like mazes, finding connected components in a graph, checking for cycles in a graph.

Would you like more detailed explanations, examples, or help with a specific problem related to BFS and DFS?