Breadth-first search (BFS)

BFS explores a graph by starting at a root node and visiting all its direct neighbors first. Then, for each of those neighbors, it visits their unvisited neighbors, and so on. this process continues, expanding outward in a wave-like pattern until the entire graph is traversed. it's like ripples spreading out when you drop a stone in a pond.

Data structure used

It uses a queue data structure to manage which nodes to visit next. when a node is visited, all its unvisited neighbors are added to the queue. the algorithm then proceeds by taking the node from the front of the queue.

Time and Space complexity

Time complexity: $O(V+E)$, V is the no.of vertices (Nodes) and E is the no.of edges. this is because every vertex & every edge will be visited at most once.

Space complexity: $O(V)$, it is needed to store the queue. in the worst-case the queue might hold all the vertices in the graph.

Sparse vs Dense Graphs:

* Sparse Graphs: A graph with relatively few edges ($E \approx V$). BFS performs well as its time complexity is efficient
* Dense Graphs: A graph with many edges ($E \approx V^2$). BFS is still a good choice, as it systematically explores all nodes and edges

Depth-First Search (DFS)

DFS explores a graph by starting at a root node & traversing as it can down a single Path or branch before it hits a dead end or a visited node. once it can't go any further, it backtracks to the last node that has unvisited neighbors and continues the process down a different branch.
the analogy fits because DFS explores a graph by going as deeply as possible down a single Path or branch before it backtracks to explore another Path

Data structure used

It uses a stack to keep track of the nodes to visit. when a node is visited, one of its unvisited neighbors is pushed onto the stack. the algorithm then proceeds by taking the node from the top of the stack

Time & Space complexity:-

Time complexity :- $O(V+E)$. V is the no. of vertices & E is the no. of edges. similar to BFS every vertex & edge is visited once.

Space complexity :- $O(V)$. It is needed for the recursion stack or the explicit stack. in the worst case, the stack depth can be equal to the no. of vertices.

Sparse vs Dense Graphs

Sparse graph :- DFS is well-suited for sparse graph, it can quickly explore deep Paths without excessive backtracking

Dense graph :- DFS is efficient in dense graph as it time complexity remains the same. the key difference compared to BFS is in the order of exploration. it can impact Performance in specific application