**Assumptions**

* Graph has **N** nodes and **E** edges.
* Default representation: **adjacency list** (otherwise noted).
* Graph may be directed/undirected, unweighted; we maintain a visited set.
* For disconnected graphs: run from each unvisited node to cover all components (total cost still sums to the same asymptotics).

**How they explore**

* **BFS**: Uses a **queue (FIFO)**; explores **level by level** from a source. In unweighted graphs, it yields **shortest paths (in edges)** from the source.
* **DFS**: Uses a **stack (LIFO)**—typically recursion (call stack) or an explicit stack; goes **as deep as possible**, then backtracks.

**Core data structures**

* **BFS**: adjacency list, visited[N], **queue** (can grow to O(N)), optional parent[]/dist[].
* **DFS**: adjacency list, visited[N], **recursion/stack** (worst-case depth O(N)), optional parent[].

**Time complexity (adjacency list)**

* Each vertex is processed at most once → **O(N)**.
* Each edge is examined at most twice (undirected) or once (directed) → **O(E)**.
* **BFS:** **Θ(N + E)**
* **DFS:** **Θ(N + E)**  
  (For a single-source run, replace N,E with reachable Nr,ErN\_r,E\_r; over all components the total remains Θ(N+E).)

**Space complexity (excluding graph storage)**

* visited: **Θ(N)**
* **BFS queue**: **O(N)** worst case
* **DFS stack/recursion**: **O(N)** worst case
* Optional arrays (parent/dist): **O(N)**
* **BFS auxiliary space:** **Θ(N)**
* **DFS auxiliary space:** **Θ(N)**
* Graph storage (adj list): **Θ(N + E)**

**Sparse vs. dense**

* **Sparse:** E=O(N)E = O(N) → time **Θ(N)**; aux space **Θ(N)**; storage **Θ(N)**.
* **Dense:** E=Θ(N2)E = Θ(N^2) → time **Θ(N^2)**; aux space **Θ(N)**; storage **Θ(N^2)**.

**If using an adjacency matrix (for comparison)**

* Scanning neighbors costs **Θ(N)** per vertex.
* **BFS:** **Θ(N²)**; **DFS:** **Θ(N²)** (regardless of E).
* Storage **Θ(N²)**; aux space still **Θ(N)**.

**Quick compare**

* **Exploration**: BFS (levels) vs DFS (depth-first).
* **Primary structure**: BFS (queue) vs DFS (stack/recursion).
* **Time (adj list)**: both **Θ(N + E)**.
* **Space (aux)**: both **Θ(N)**.
* **Shortest paths in unweighted graphs**: BFS ✅, DFS ❌.

**Notes**

* **Early exit** when a target is found doesn’t change worst-case bounds.
* **Directed vs undirected** only affects constant factors for edge scans.
* **Deep graphs**: recursive DFS can overflow call stack; use an explicit stack if needed.