BFS and DSF : Time & space Complexity

How each algorithm explores the graph

* BSF :-

Explores level by level from the Source.
It uses a queue (First in first out)

* Enque the Source node
* Repeatedly deque a node u, Check all neigbors v. if unvisited, mark visited and enqueue v.

* DSF (Depth-First Search):

Explores as deep as possible along one path then back tracks. It uses recursion stack or an explicit stack (Last in first out).

→ Data Structures used

BFS: Queue + visited array + adjacency list.
DFS: Stack (recursion or explicit) + visited array + adjaceny
Graph reperesentation: By default adjacency list (O(N+E) space). Adjacency matrix (O(N^2) space) also Possible.

Complexity Derivations

BFS:

* Time Complexity:
* Each vertex is enqueued/dequeued once → $O(N)$
* Each edge is checked once (directed) or twice Undirected → $O(E)$
* Total: $O(N+E)$

* Space Complexity:

* Graph storage: $O(N+E)$
* Queue: up to $O(N)$
* Visited array: $O(N)$
* Total: $O(N+E)$

<u>DFS</u>

* Time Complexity:
* Each Vertex is visited once → $O(N)$
* Each edge is explred once directed or twice (undirected) → $O(E)$
* Total: $O(N+E)$

* Space Complexity:
* Graph Storage: $O(N+E)$
* Recursion/Stack depth: $O(N)$
* Visited array: $O(N)$
* Total $O(N+E)$

* Sparse vs Dense Gaph

* Sparse graphs: $E = O(N)$
→ BFS/DFS take $O(N)$ time and $O(N)$ space.

* Dense graph: $E = O(N^2)$
→ BFS/DFS take $O(N^2)$ time and space

* Adjacency matris: Always $O(N^2)$ time and $O(N^2)$ space, regarless of how many edges.

Final Results:

Adjacency list:
* BFS: $O(N+E)$ time, $O(N+E)$ space
* DFS: $O(N+E)$ time, $O(N+E)$ space
* Adjacency matrix:
* BFS/DFS: $O(N^2)$ time, $O(N^2)$ space.