BFS and DSF : Time & Space Complexity

How each algorithm explores ~~the~~ ~~given~~ graph

* BSF:- Explores level by level from the source. It uses a queue.

* Enque the source node
* Repeatedly deque a node u, check ~~both~~, neighbors v. If unvisited, mark visited and enquere.

* DSF (Depth - First Search).
Explores as deep as possible along one path then back tracks. It uses recursion stack or an explicit stack.

Data Structures used:

BFS: Queue + Visited array + adjacency list

DFS: Stack + Visited array + adjacency.

Graph representation:- By default adjacency list (O(N+E). Adjacency matrix (O(N^2) space) also possible.

Complexity Derivations:-

BFS:

* Time complexity.
* Each vertex in enqued/dequeued once
* Each edge is checked once 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)

DFS

* Time Complexity
* Each vertex is visited once or twice (undirected) → O(E)
* Total: O(N+E)
*
* SPace Complexity:-
* Graph storage: O(N+E)
* Recursion: O(N)
* Visited array: O(N)
* Total O(N+E)

* Sparse VS Dense Graph

* 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 matrix: Always O(N^2) time and O(N^2) Space, regardless 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.