Time and space complexity of BFS and DFS

Breadth First Search (BFS)

Traversal idea:
BFS explores the graph level. it visits all neighbors of a node before moving to the next depth level.
(Think of it as wave-like expansion)

Data structure used:
- A queue (FIFO) is used to store nodes that are discovered but not yet processed.
- A visited array (or set) keeps track of visited nodes.

Time complexity
- visiting each vertex takes ~~(ov)~~. O(V)
- Exploring edges takes O(E) (each edge is looked at once)

$$T(V+E) = O(V+E)$$

- using adjacency list → O(V+E)
- using adjacency matrix → $O(V^2)$

Space complexity
- Queue may hold up to all vertices in the worst case → O(V)
- visited array also takes O(V)
- Total space: O(V)

Sparse Graphs:
- Adjacency list is efficient → runs in O(V+E)

Dense Graphs:
- Adjacency matrix leads to $O(V^2)$, which may be less efficient.

Depth First Search (DFS)

Traversal Idea:

DFS explores as far as possible along one branch before backtracking. It recursively visits unvisited neighbors until a dead end is reached, then backtracks.

Data Structure used:

- A stack (LIFO) (explicit or via recursion) to manage nodes.
- A visited array to avoid revisiting.

Time Complexity

- Visiting vertices : $O(V)$
- Exploring edges : $O(E)$

$$T(V+E) = O(V+E)$$

- with adjacency list → $O(V+E)$
- with adjacency matrix → $O(V^2)$

Space Complexity

- Stack can store up to $O(V)$ nodes in the recursion path.
- visited array → $O(V)$
- Total space : $O(V)$

Sparse Graphs:

- When $E << V^2$, time complexity $\approx O(V+E)$

Dense Graphs:

- As E approaches V^2, complexity becomes closer to $O(V^2)$