# BFS & DFS (Time & space complexity)

(1) Algorithm explanation with graph

* BFS (Breadth first search)
  * Explores level by level, starting from the source.
  * visit all neighbours of the current node before moving deeper.
  * Guarantees shortest Path in an unweighted graph.

* DFS (Depth first search)
  * Explores as deep as possible along one branch before backtracking.
  * uses recursion (or stack) to track unexplored paths.
  * Does not guarantee short path

(2) Data structures used

* BFS :- Queue (FIFO)
* DFS :- Stack (LIFO) or recursion.

(3) Time complexity

* Both BFS and DFS must visit all vertices (N) and traverse all edges (E).
* Each vertex is enqueued/Pushed once and dequeued/Popped once.
* Each edge is checked at most twice (undirected graph).

∴ Time complexity :- O(N+E).

(4) SPACE complexity

* BFS
  * Needs a queue that can hold upto $O(N)$ nodes in the worst case.
  * Also requires adjacency list storage :- $O(N+E)$
  * space :- $O(N+E)$.

* DFS
  * Needs recursion stack (or explicit stack).
  * In worst case, recursion depth = $O(N)$.
  * Plus adjacency list : $O(N+E)$.
  * space :- $O(N+E)$.

(5) Sparse vs Dense Graphs

* Sparse graph :- $E \approx O(N)$
  * complexity :- $O(N+E) \approx O(N)$.

* Dense Graph :- $E \approx O(N^2)$
  * complexity :- $O(N+E) \approx O(N^2)$

(6) Assumptions :-

* Graph stored as adjacency list (not adjacency matrix, since that would be $O(N^2)$ space).
* Graph is connected; otherwise, BFS/DFS must be run on each component.