Time and space complexities:

BFS Time and space complexity:

Time complexity:

Formula: $O(V+E)$

V = Number of Vertices or Nodes

E = Number of edges

Every Node is visited once, and every edge is checked once. The total operations are proportional to the some Sum of Nodes and edges. $O(V+E)$.

$T(V) = C \times V = O(V)$

Edges denoted as $d(V)$

$T(E) = \Sigma [d(V)]$

for all V in V $= 2 * |E| = O|E|$

Time complexity:

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

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

$\boxed{T(n) = O[|V|+|E|]}$

Space complexity:

Formula: $O(V)$

BFS uses a queue to keep track of nodes at the current level. In the worst case the queue may store all nodes at one level.

$\boxed{S(n) = O|V|}$

For each node, Every Adjacent edge is checked. each edge connects two nodes. $O(E)$

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

Breadth-First Search: (BFS)

BFS is a fundamental algorithm for traversing, searching graphs, Exploring nodes layer by layer from a starting node

- Uses a queue to maintain the visiting sequence [FIFO] [First-In, First-out] as data structure.

→ **Representation:** Time complexity.

Adjacency list : $O(V+E)$

Adjacency Matrix : $O(V^2)$

∴ V = no. of vertices
E = no. of edges

→ **Space complexity**

Uses Queue, it hold all nodes at a particular level of graph – $O(V)$

To track visited nodes and Extra space to keep track of which nodes have been visited $O(V)$

→ **Sparse Graphs:**

Adjacency list is much more space efficient ($O(V+E)$)

It makes BFS faster and less memory-intensive.

→ **Dense Graph:**

Has a no. of edges close to the maximum (V^2)

Adjacency Matrix may used but with increased space and cost ($O(V^2)$)

Depth-First Search:

- DFS Explores each branch of a graph as deeply as possible before backtracking.
- It uses Stack data structure explicitly or recursion to manage nodes during traversal
- It visit all vertices reachable from the source. does not about shortest path.

→ **Representation:** Time Complexity

Adjacency list → $O(V+E)$ ∴ Efficient for Sparce graph

Adjacency Matrix → $O(V^2)$ ∴ Inefficient for Sparce graph due to Space & time for each edge

→ **Space complexity:**

Stack stores up to $O(V)$ nodes in the deepest path

Visited set or list Tracks visited nodes.

→ **Sparse Graph:**

Adjacency list recamended. Space & time efficiency is $O(V+E)$

→ **Dense Graph**

Adjacency Matrix may be used for canatant-time edge ups but it consumes $O(V^2)$ space is practical only in small or very dence graph.