1) Depth-First Search (DFS):

- Advantage: Memory efficient.
- Drawback: Not necessarily find the shortest path.
- Limitation: Can get stuck in infinite loops for cyclic graphs.

EXAMPLE:

A -- B -- C
| |
D -- E

Starting at A, DFS might go A-B-C-E-D or A-D-E-C-B.

2) Breadth-First Search (BFS):

- Advantage: Guarantees shortest path.
- Drawback: More memory usage compared to DFS.
- Limitation: May not be efficient for dense graphs.

Example:

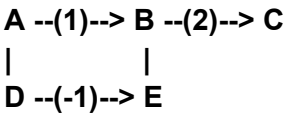
A -- B -- C
| |
D -- E

Starting at A, BFS would go A-B-D-C-E.

3) Bellman-Ford Algorithm:

- Advantage: Handles negative edge weights.
- Drawback: Slower than Dijkstra's for non-negative weights.
- Limitation: Inefficient for large graphs.

Example:

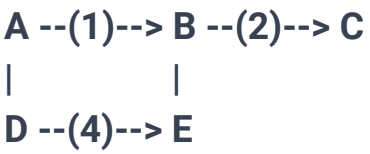


Computes shortest paths from A to all nodes.

4) Dijkstra's Algorithm:

- Advantage: Finds the shortest paths in weighted graphs.
- Drawback: Doesn't work with negative edge weights.
- Limitation: Inefficient for large graphs.

Example:



Computes shortest paths from A to all nodes.

These algorithms serve different purposes and are suited for different types of graphs and problems. The choice depends on the specific requirements and characteristics of the graph at hand.