Graph Traversal

Mina Gabriel

HU

April 9, 2025

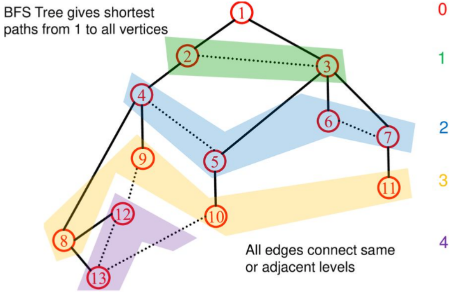
Single-Source Shortest Paths (Unweighted Graphs)

Purpose: Find the shortest distance from a given source node to all other nodes in an *unweighted* graph.

Approach: Use Breadth-First Search (BFS) starting from the source node. BFS explores the graph level-by-level, so the first time a node is reached, the shortest path to it has been found.

Time Complexity: $O(|V| + |E|)$ — Vertex and edge is visited once.

Returns: A dictionary or array mapping each node to its shortest distance from the source.



Algorithm 1 BFS-Shortest-Paths(G, s)

```
for all v ∈ G do
    distance[v] ← ∞
end for
distance[s] ← 0
Q ← empty queue
Enqueue s into Q
while Q is not empty do
    u ← Dequeue(Q)
    for all v ∈ Adj[u] do
        if distance[v] = ∞ then
            distance[v] ← distance[u] + 1
            Enqueue v into Q
        end if
    end for
end while
return distance
```

Single-Source Reachability with BFS or DFS

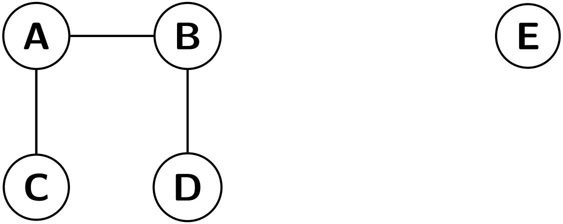
Purpose: Determine which nodes are reachable from a given source node in a graph.

Approach: Use either Breadth-First Search (BFS) or Depth-First Search (DFS). Track nodes as they are visited during traversal — those are the reachable ones.

Time Complexity: $O(|V| + |E|)$ in the worst case — when all nodes and edges are reachable. In practice: $O(\text{reachable nodes} + \text{reachable edges})$ if the graph is sparse or the source is not connected to all nodes.

Returns: A `set` of nodes reachable from the source.

Graph Example: Reachability from A



Reachable from A: {A, B, C, D} **Unreachable:** {E}

DFS Reachability (Iterative Version) – Pseudocode

Algorithm 2 DFS-REACHABILITY(G, s)

```
visited ← ∅
stack ← empty stack
push s onto stack
while stack is not empty do
    u ← pop from stack
    if u ∉ visited then
        add u to visited
        for all v ∈ Adj[u] do
            push v onto stack
        end for
    end if
end while
return visited
```

Connected Components with Full-BFS or Full-DFS

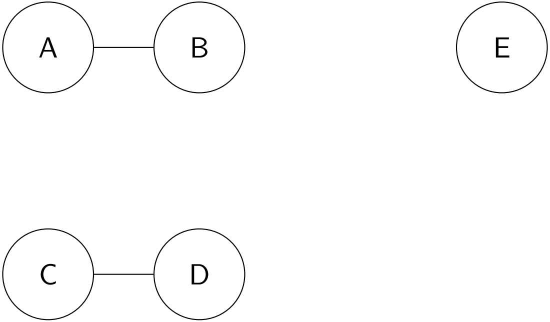
Purpose: Identify all connected components in an *undirected* graph — groups of nodes where each node is reachable from any other node in the same group.

Approach: Repeatedly perform BFS or DFS starting from unvisited nodes. Each search explores one connected component.

Time Complexity: $O(|V| + |E|)$ — Each node and edge is visited exactly once across all searches.

Returns: A list of sets (or lists), where each set contains the nodes in one connected component.

Example: Connected Components



Connected Components: {A, B}, {C, D}, {E}

Algorithm 3 CONNECTED-COMPONENTS(G) **ASSIGNMENT**