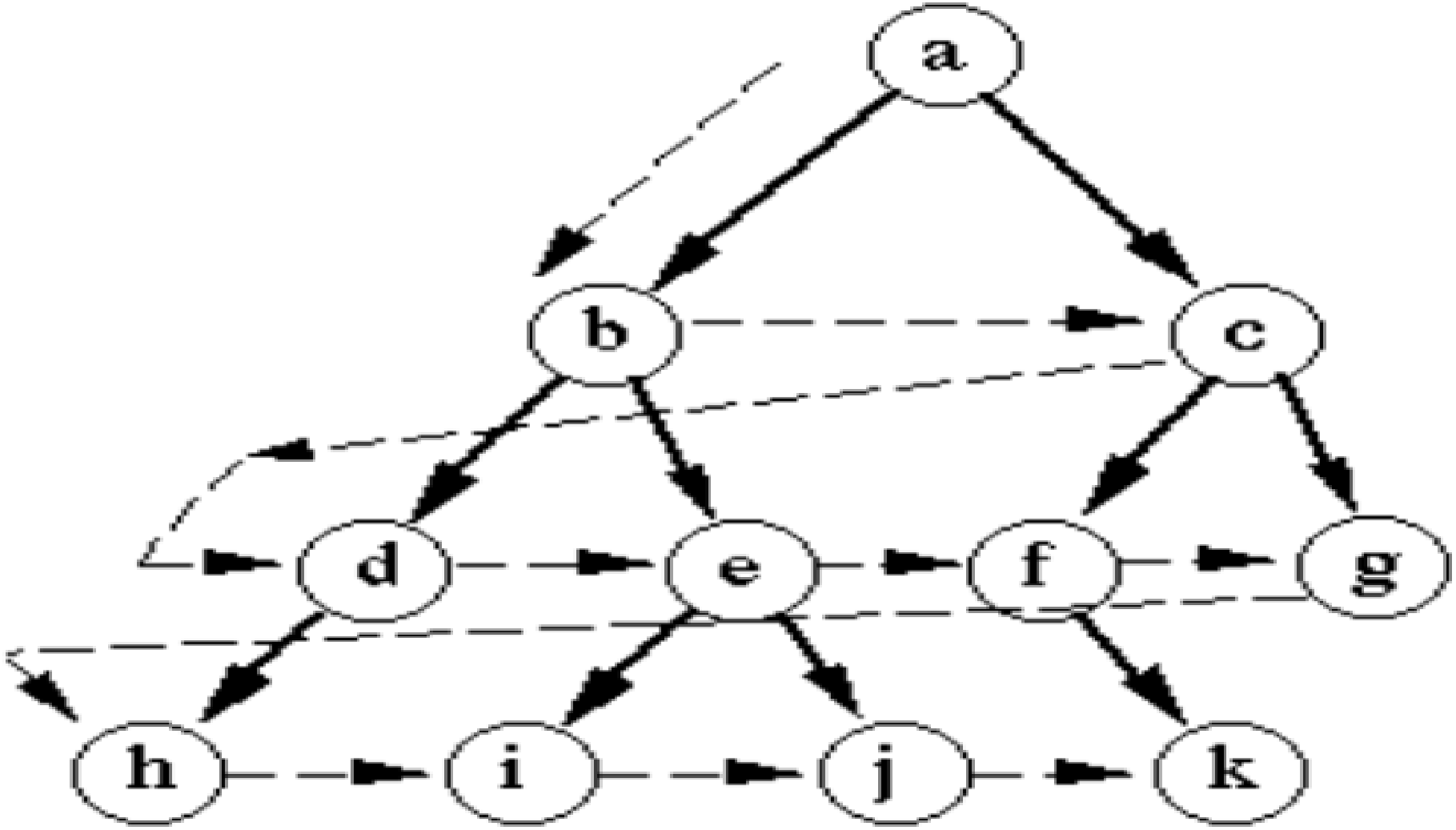
Understanding Breadth First Search (BFS)

Introduction to BFS

- Breadth-First Search (BFS) is a way to explore a graph by starting at one node and visiting all its neighbors before moving to the next level.
- It works like searching layer by layer, using a queue to keep track of nodes.
- **Imagine you are exploring a neighborhood:** you check all the houses on one street before moving to the next.
- BFS is systematic and ensures every point is checked in the right order.

Introduction to BFS



Breadth-first search

What are Levels in BFS?

- **BFS explores a graph level by level, starting from a given node (source), visiting all its neighbors before moving to the next level.**
- In BFS (Breadth-First Search), "levels" represent how far nodes are from the starting node.

- **Level 0**: The starting node.
- **Level 1**: Nodes directly connected to the starting node.
- **Level 2**: Nodes connected to Level 1 nodes, and so on.

Key Characteristics of BFS

- Breadth-First Search (BFS) explores one level of a graph or tree at a time, starting from a source node.
- It uses a **queue** to keep track of nodes to visit next.
- BFS ensures that all nodes at the current level are visited before moving to the next.
- It is best for finding the shortest path in an unweighted graph and works systematically.

How BFS Works

- BFS starts at a chosen node (source) and explores all its neighbors first before moving to the next level. It uses a **queue** to keep track of nodes to visit. The process is simple:

1. Add the starting node to the queue.
2. Visit the node and add its unvisited neighbors to the queue.
3. Repeat until all nodes are visited, level by level.

Queue in BFS

- In BFS (Breadth-First Search), a **queue** is used to keep track of the nodes that need to be visited.
- Nodes are added to the queue when discovered (enqueue) and removed when processed (dequeue).
- This ensures BFS explores all neighbors of a node before moving to the next level, making it a systematic, level-by-level traversal method.
- The queue ensures the **First In, First Out (FIFO)** principle is followed.

Applications of BFS

- **Shortest Path:** Find the shortest path in unweighted graphs, like maps or networks.
- **Maze Solving:** Explore paths to reach the destination.
- **Social Networks:** Find connections between people (like mutual friends).
- **Web Crawlers:** Explore web pages level by level.
- **Connected Components:** Identify groups in graphs where nodes are directly or indirectly connected.

Advantages of BFS

- BFS (Breadth-First Search) is helpful because it always finds the shortest path in unweighted graphs.
- It explores all possible options at one level before moving deeper, ensuring that no possible solution is skipped.
- BFS is great for solving problems like finding the shortest route in maps, discovering connected groups in a network.
- It's a clear and organized way to explore step by step.

Limitations of BFS

- BFS needs more memory because it uses a queue to store nodes.
- If the graph is very large, the memory usage can become a problem.
- It is also slower when the graph has many levels, as it explores all nodes level by level.
- BFS doesn’t work well for weighted graphs when we need the shortest path, as it ignores edge weights.