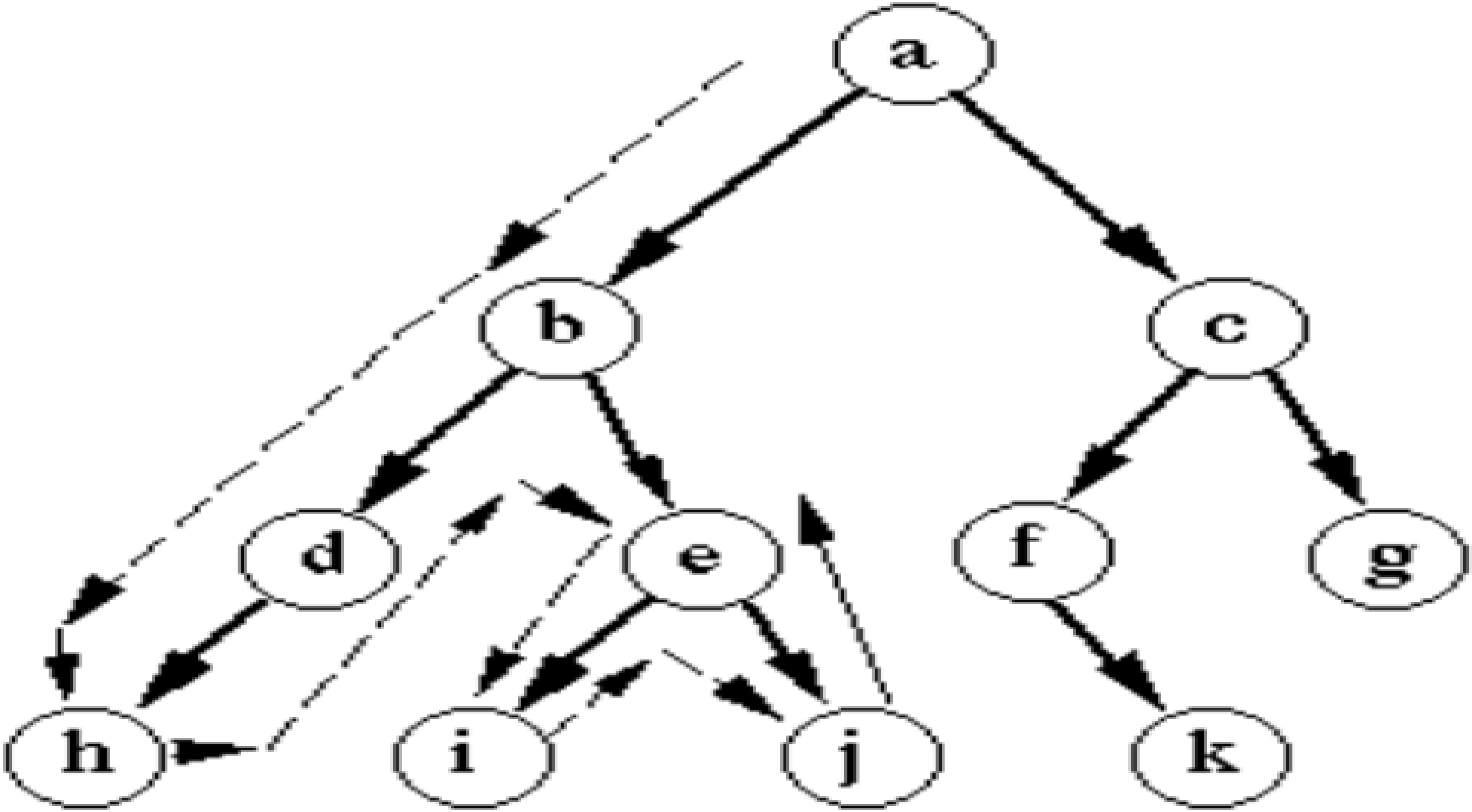
Understanding Depth First Search (DFS)

Introduction to DFS

- Depth-First Search (DFS) is a way to explore graphs or trees by starting at a node and going as deep as possible along one path.
- If you reach the end or no more nodes are left, you go back and try another path.
- It’s like exploring all rooms in a building by going through one hallway completely before backtracking and trying the next hallway.

Introduction to DFS



Depth-first search

What are Depths in DFS?

- DFS explores a graph depth by depth, starting from a given node (source), visiting one branch as deeply as possible before backtracking.
- In DFS (Depth-First Search), "depths" represent how far nodes are from the starting node in terms of exploration.
- Depth 0: The starting node.
- Depth 1: Nodes directly connected to the starting node.
- Depth 2: Nodes connected to Depth 1 nodes, and so on.

Key Characteristics of DFS

- Depth-First Search (DFS) explores a graph or tree by going as deep as possible along one path before moving to another.
- It uses a stack (or recursion) to keep track of nodes. DFS does not guarantee the shortest path but is great for exploring all possibilities.
- It goes deep first and backtracks when no more unvisited nodes are left.

How DFS Works

- DFS starts at a node and explores one path as far as possible before backtracking.
- It uses a stack (or recursion) to keep track of nodes.
- Visit a node, mark it as visited, and move to an unvisited neighbor.
- If no neighbors are left, backtrack to the previous node and continue.
- This process repeats until all nodes are visited. It's like exploring deep paths in a maze one by one.

Stack in DFS

- In DFS (Depth-First Search), a **stack** helps keep track of nodes to visit.
- You start at the root node, push it onto the stack, and explore one path as far as possible.
- If you hit a dead end (no more unvisited neighbors), you backtrack by popping nodes from the stack and continue exploring other paths.
- The stack ensures the traversal dives deep first before checking other branches.

Applications of DFS

- **Maze Solving:** It explores all possible paths in a maze to find a solution.
- **File Searches:** Used to go through all files and folders in a computer.
- **Cycle Detection:** Helps find loops in a graph.
- **Game Moves:** Explores all possible moves in a game to find the best path.

Step-by-Step DFS

1. Start from the root node.
2. Visit the first unvisited neighbor and go deeper.
3. Keep visiting deeper nodes until no more unvisited neighbors are left.
4. Backtrack to the previous node and explore its other neighbors.
5. Repeat this process until all nodes are visited.

- ➢ Think of it like exploring a cave, going as deep as possible before coming back to explore another path!

Advantages of DFS

- Depth-First Search (DFS) uses less memory than Breadth-First Search (BFS) since it doesn’t store all nodes at one level.
- It’s great for exploring deep paths in graphs or trees.
- DFS is useful for problems where we need to visit all paths, like solving puzzles or finding all possible solutions.
- It’s also helpful in tasks like detecting cycles or finding connected components in graphs.

Limitations of DFS

- DFS might not find the shortest path in a graph because it explores deeply first.
- It can also get stuck in infinite loops if the graph has cycles and those cycles are not handled properly.
- Additionally, if the graph is very large, DFS may take a long time to find the desired solution since it explores one path fully before moving to the next.