DEPTH-FIRST SEARCH

DEPTH-FIRST SEARCH

- Always expands the deepest node in the current frontier of the search tree.
- Depth-first search uses LIFO queue whereas Breadth-first search uses FIFO queue.
- Recursive function that calls itself on each of its children in turn.

DEPTH-FIRST SEARCH

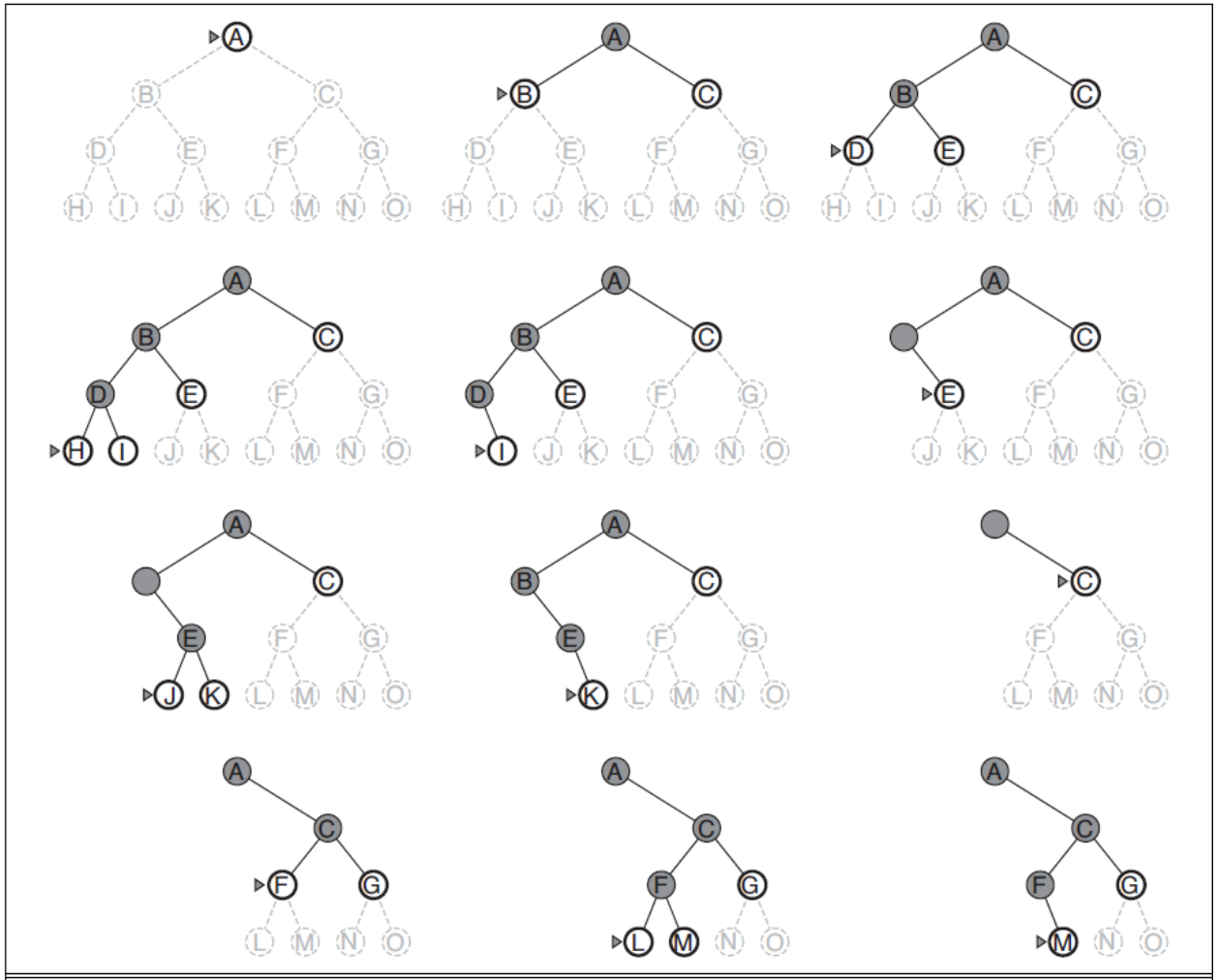

Figure 3.16 Depth-first search on a binary tree. The unexplored region is shown in light gray. Explored nodes with no descendants in the frontier are removed from memory. Nodes at depth 3 have no successors and M is the only goal node.

DEPTH-FIRST SEARCH

Graph-search version

- Avoids repeated states and redundant path.
- Complete in finite space.

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

Tree-search version

- Not complete because of loops.
- Arad – Sibiu – Arad – Sibiu

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

CHALLENGES

- Non-optimal
- Searches complete path even if goal node may be very close to root node in another branch.

TIME COMPLEXITY

- $O(b^m)$
- m is maximum depth of any node

- For tree-search implementation, space complexity is O(bm)

- Backtracking search variant has space complexity of O(m) due to expansion of only one successor.