ARTIFICIAL INTELLIGENCE

BACS2003|BACS3074|BMCS2003

CHAPTER 3 UNINFORMED SEARCH

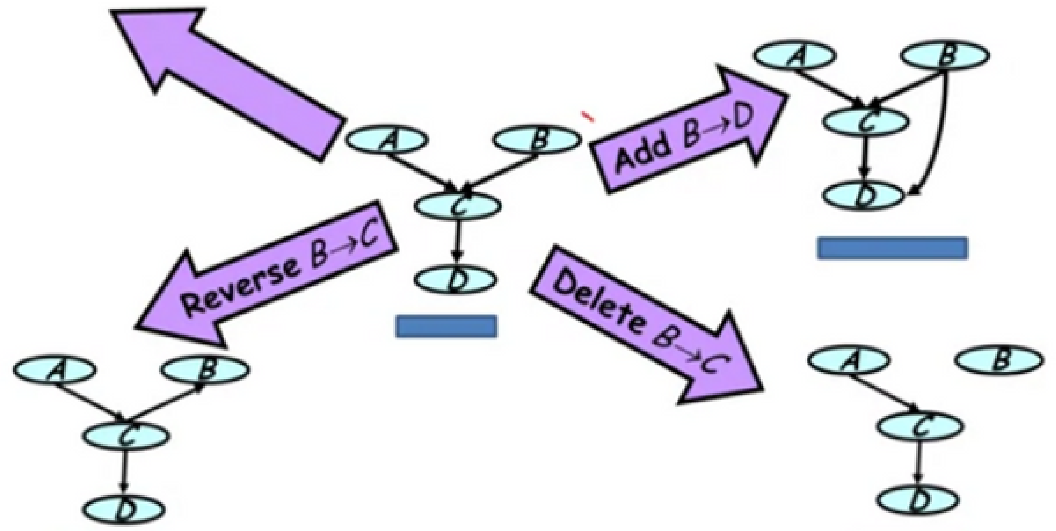
OUTCOMES

1. **Uninformed search**
2. **Performance evaluation**

SEARCH STRATEGIES

- UNINFORMED SEARCH / BLIND SEARCH

- INFORMED SEARCH / HEURISTIC SEARCH



UNINFORMED SEARCH

1. Breadth-first search
2. Depth-first search
3. Depth-limited search
4. Iterative deepening depth-first search
5. Bidirectional search

BREADTH-FIRST SEARCH

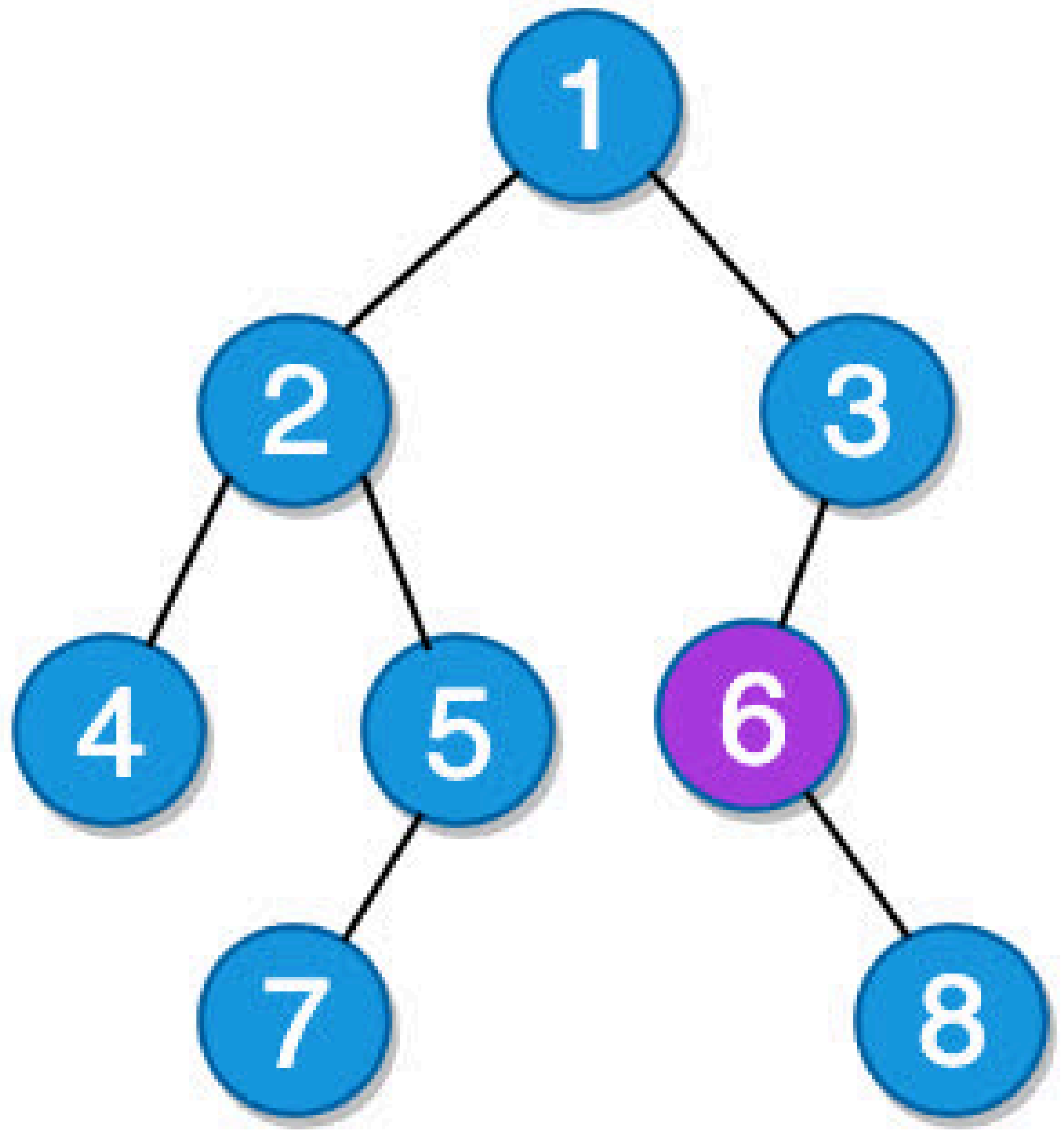
All the nodes are expanded at a given depth in the search tree before any nodes at the next level are expanded.

ALGORITHM

1. Begin with the root node.
2. Examine all of the nodes in each level before moving on to the next level.
3. Repeat the process and work downward from left to right until a solution is found.
4. Return Fail if there is no solution.

BREADTH-FIRST SEARCH

BFS



DEPTH=0

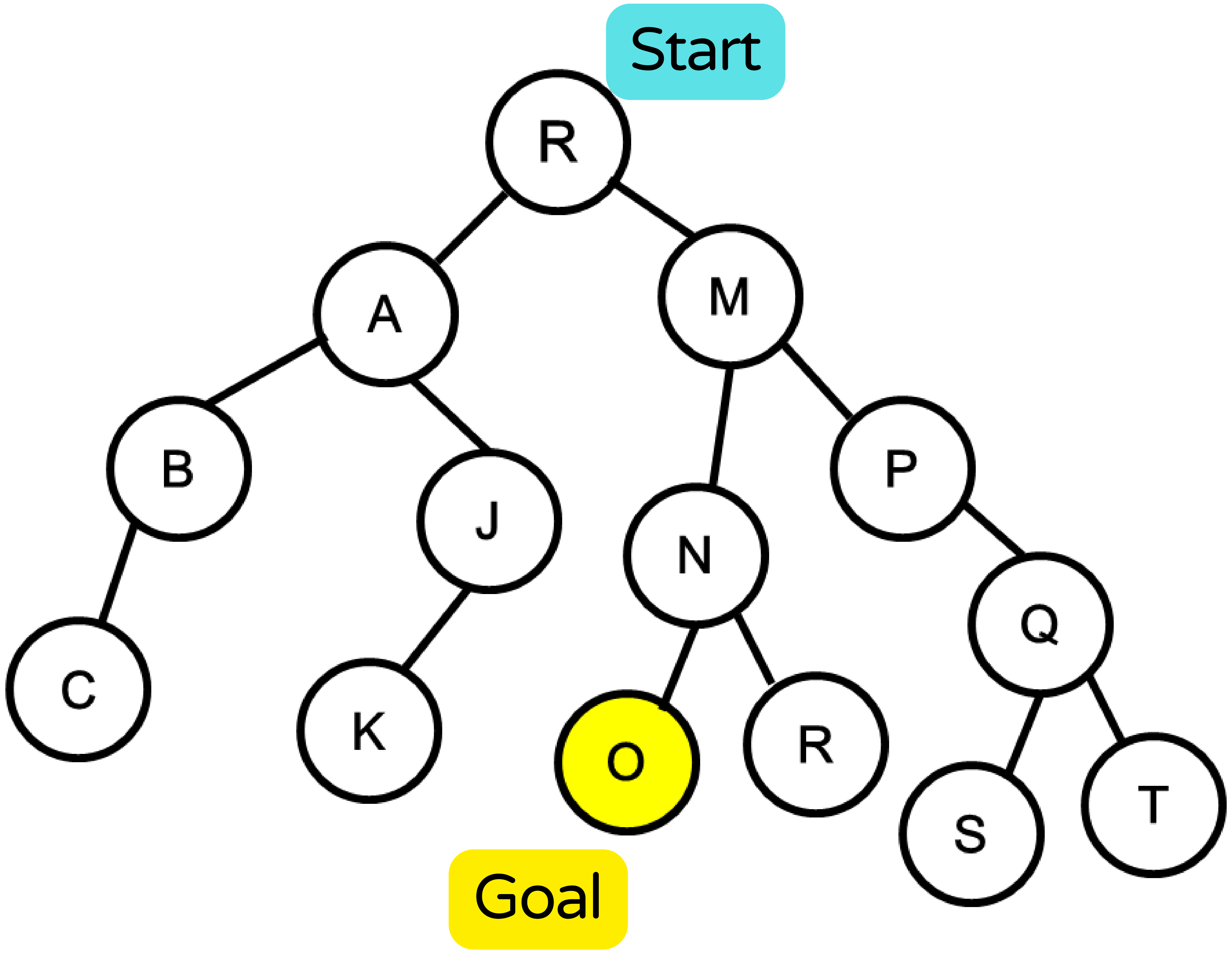
DEPTH=1

DEPTH=2

DEPTH=3

BREADTH-FIRST SEARCH

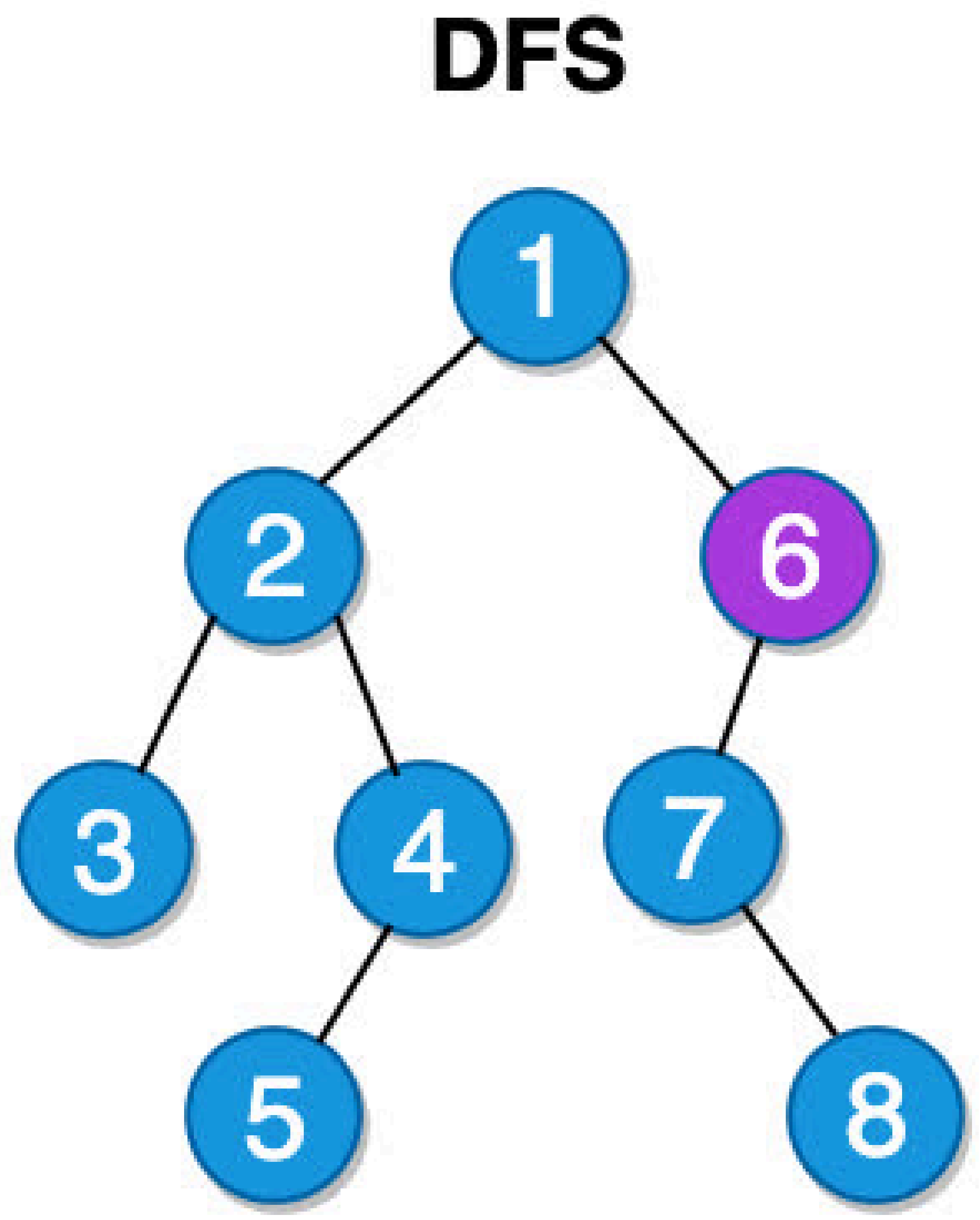
List the sequence of the search path and return path if using breadth-first search?



DEPTH-FIRST SEARCH

- Always expands the deepest node in the current fringe of the search tree.
- Nodes leading from dead end will be discarded from memory.
- Then the search will back up to the next shallowest node that still has unexplored successors.
- Implemented by calling tree search using LIFO (last-in-first-out) strategy.

DEPTH-FIRST SEARCH



DEPTH=0

DEPTH=1

DEPTH=2

DEPTH=3

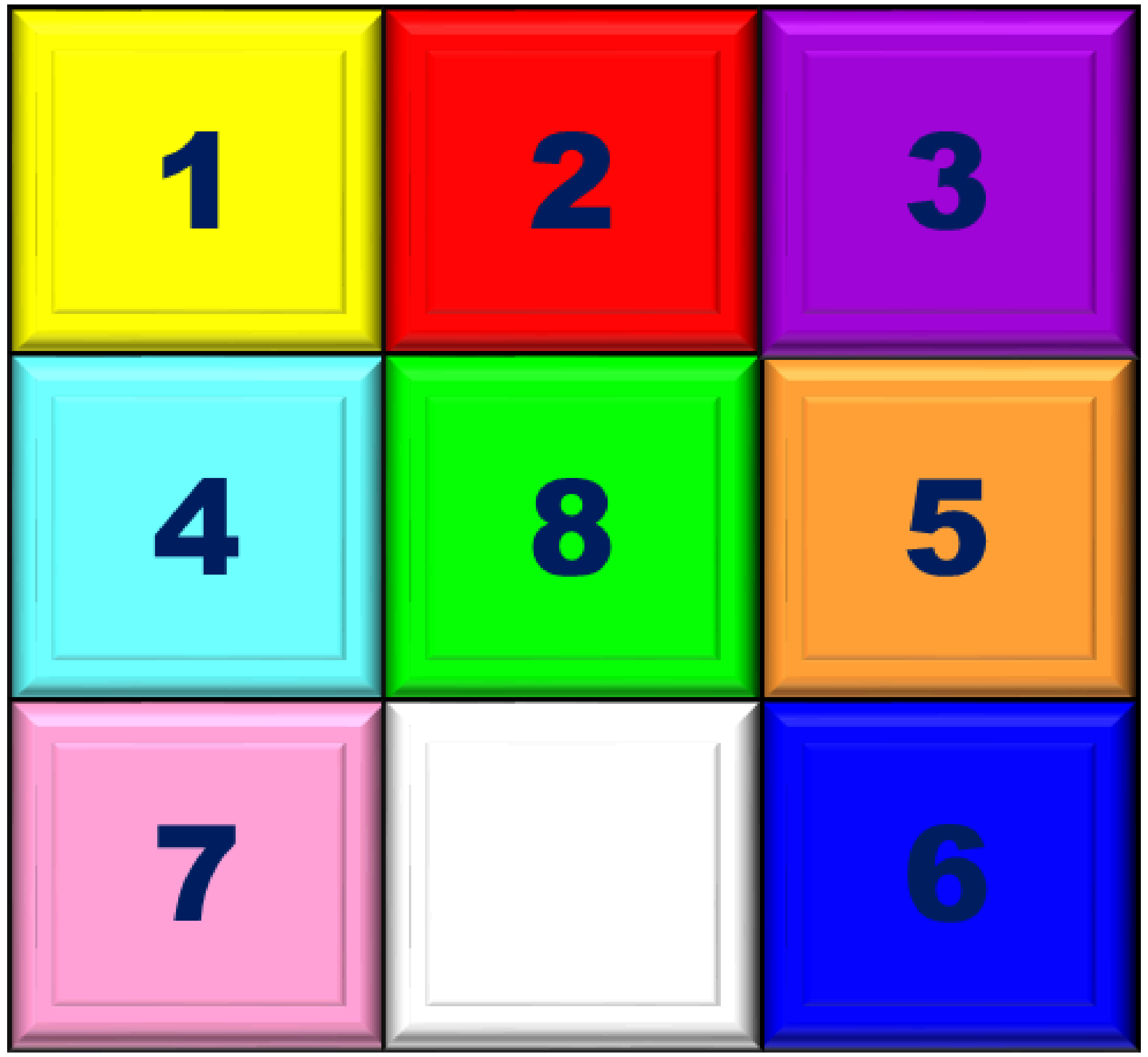
MEASURING PROBLEM-SOLVING PERFORMANCE

Criteria	Breadth-First Search (BFS)	Depth-First Search (DFS)
Completeness	Complete - BFS will find a solution if one exists, and it explores all possible paths level by level.	Not always complete - DFS may get stuck in infinite loops or miss certain paths, depending on the specific implementation and problem.
Optimality	Optimal - BFS guarantees the shortest path in terms of the number of edges or steps taken.	Not guaranteed to be optimal - DFS may find a solution faster, but it doesn't necessarily find the shortest path.
Time Complexity	The time complexity of BFS is *O(V + E)* when Adjacency List is used and *O(V^2)* when Adjacency Matrix is used, where *V* stands for vertices and *E* stands for edges.	The Time complexity of DFS is also *O(V + E)* when Adjacency List is used and *O(V^2)* when Adjacency Matrix is used, where *V* stands for vertices and *E* stands for edges.
Space Complexity	Requires more memory as it needs to store all nodes at the current level.	Requires less memory as it only needs to store the path from the root to the current node. Depth of recursion is limited.

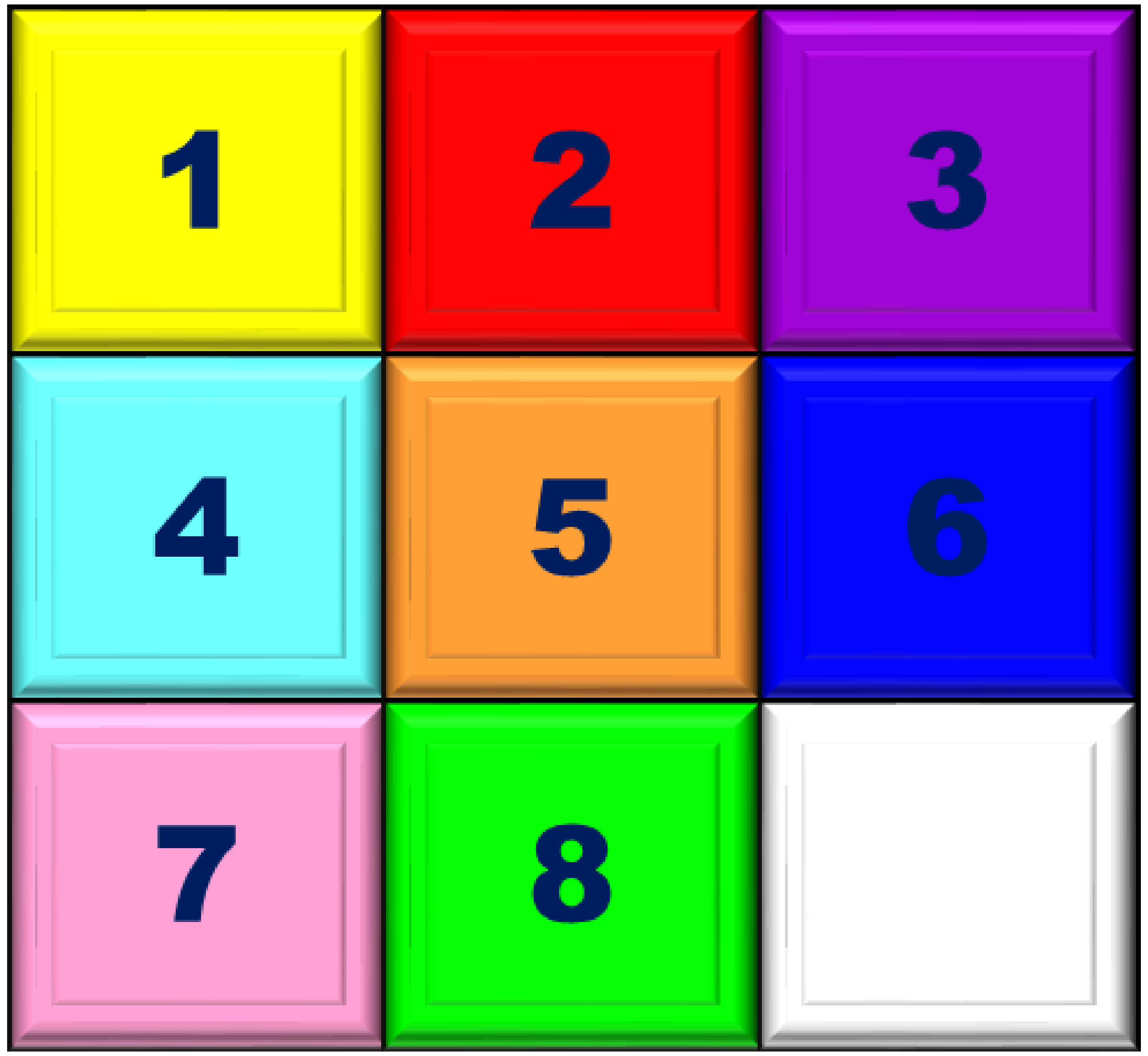
EXAMPLE 1

THE 8-SLIDING PUZZLE

SUCCESSFOR FUNCTION (OPERATOR): MOVE EMPTY TILES LEFT, RIGHT, UP, DOWN



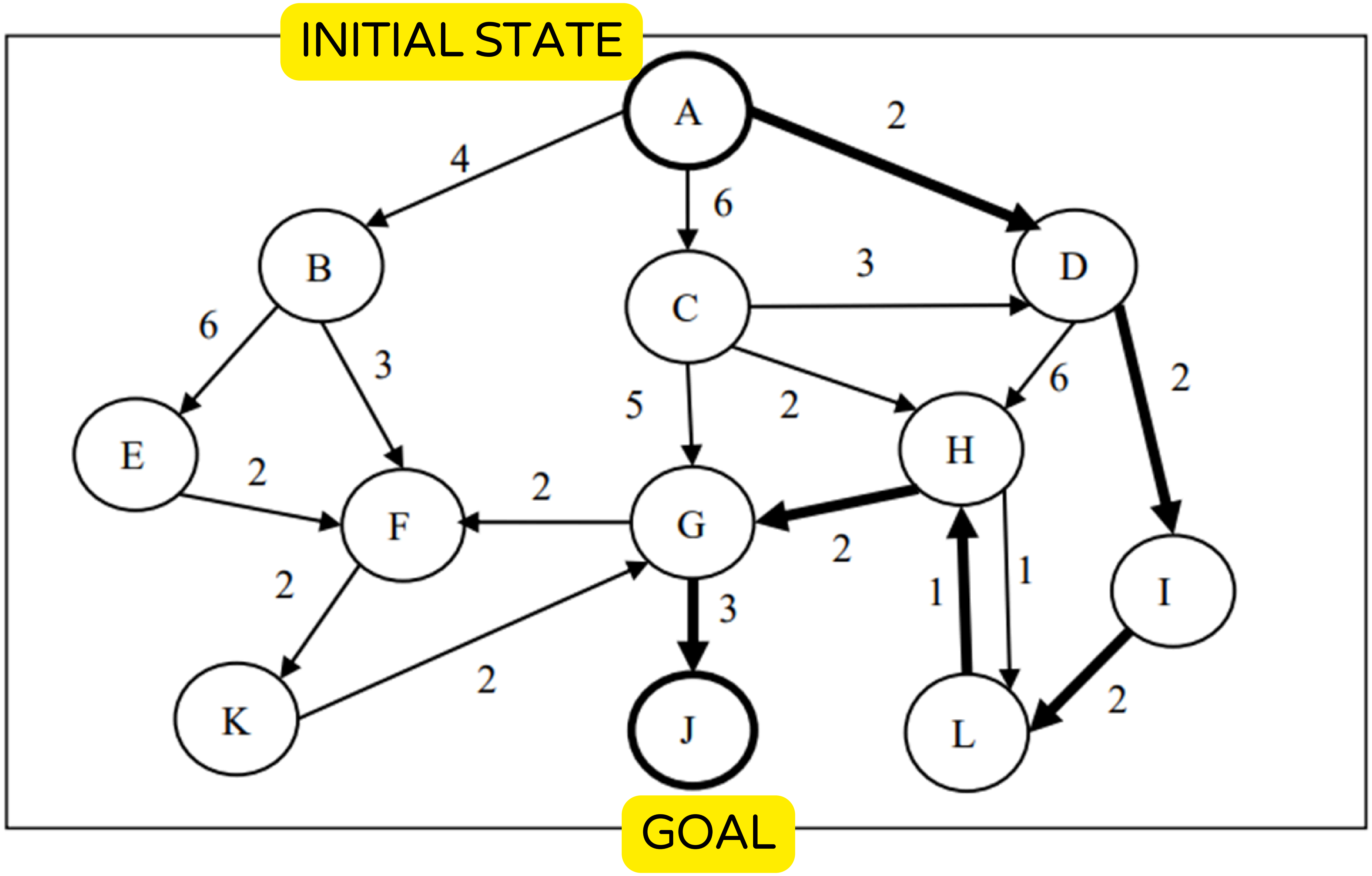
INITIAL STATE



GOAL

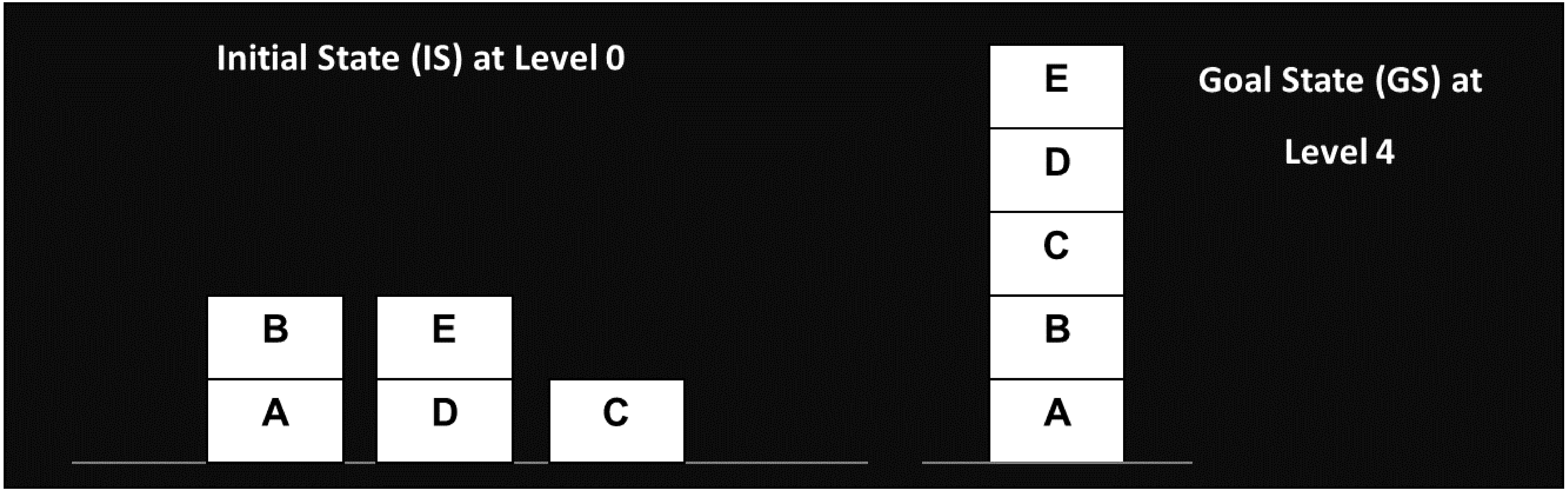
EXAMPLE 2

DIRECTED GRAPH



EXAMPLE 3

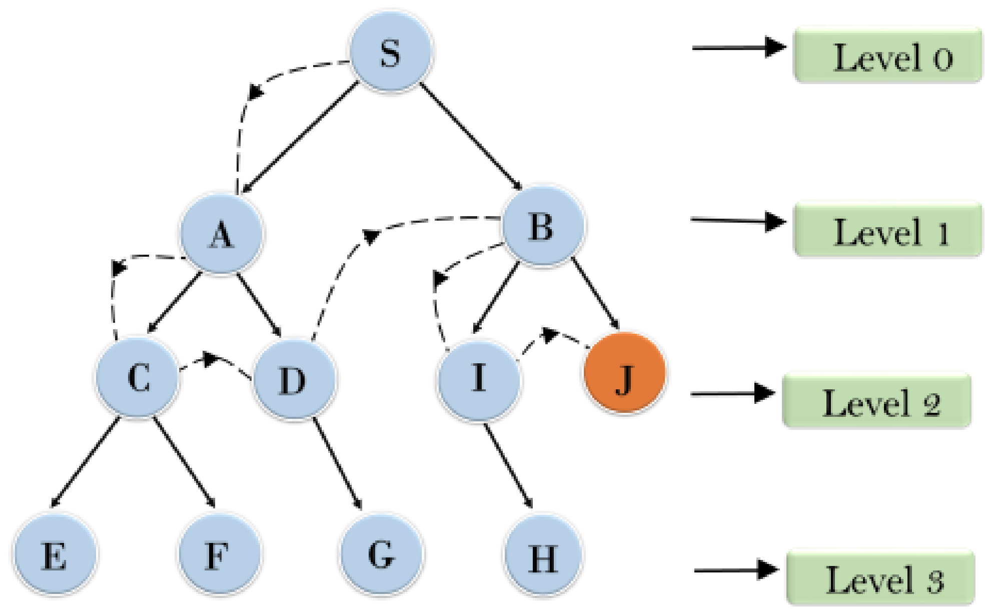
BLOCKS WORLD PROBLEM



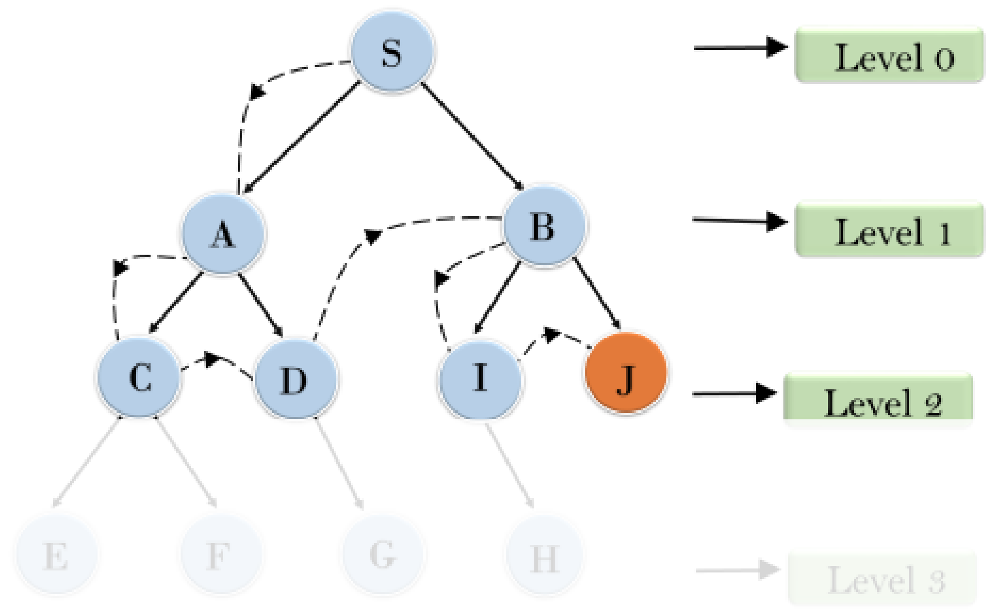
Only one block can be moved at one time

DEPTH-LIMITED SEARCH

- Depth-first search can lead to infinite path.
- Solution – Limit the depth of the expansion to avoid uncontrolled infinite path.



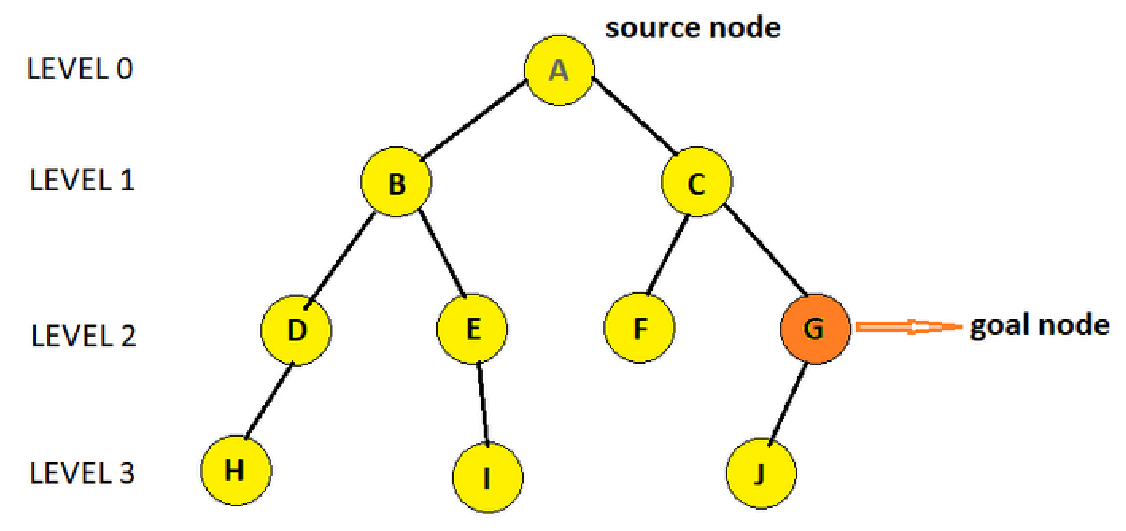
limit = level 2



Challenge – how to determine the depth limit ???

ITERATIVE DEEPENING DEPTH-FIRST SEARCH (IDS)

- In depth-limited search, there is no reliable method to decide a depth limit to include at least one solution.
- Iterative deepening can solve this by checking the entire tree by gradually increasing the depth limit – first depth 0, then depth 1, then depth 2 and so on.



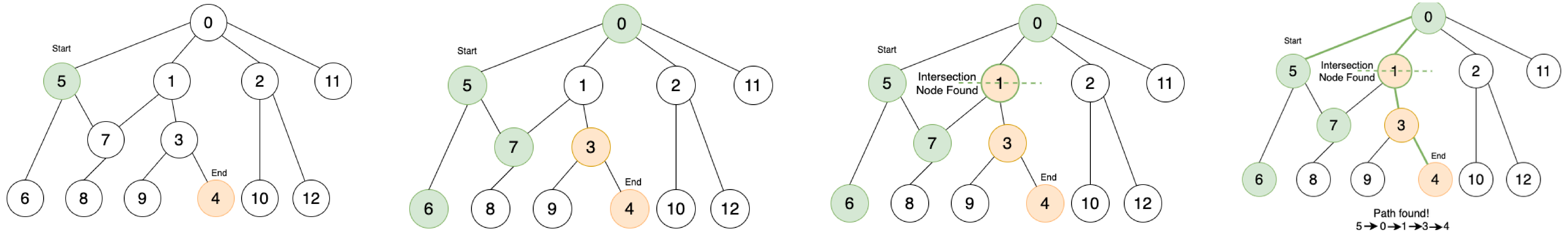
IDDFS with max depth-limit = 3
Note that iteration terminates at depth-limit=2
Iteration 0: A
Iteration 1: A->B->C
Iteration 2: A->B->D->E->C->F->G

BIDIRECTIONAL SEARCH

- Run two simultaneous searches:

1. One forward from the initial state
2. One backward from the goal (final state)

- Stop searching when the two searches meet in the middle.



THE END

NEXT LECTURE

Heuristic search