CHAPTER # 03
SOLVING PROBLEMS BY SEARCHING

PART I – UNINFORMED SEARCH STRATEGIES

COURSE INSTRUCTOR:
ENGR. FARHEEN QAZI
ASSISTANT PROFESSOR, SED

DEPARTMENT OF CE / CS / IT / BI & SE
SIR SYED UNIVERSITY OF ENGINEERING & TECHNOLOGY

AGENDA

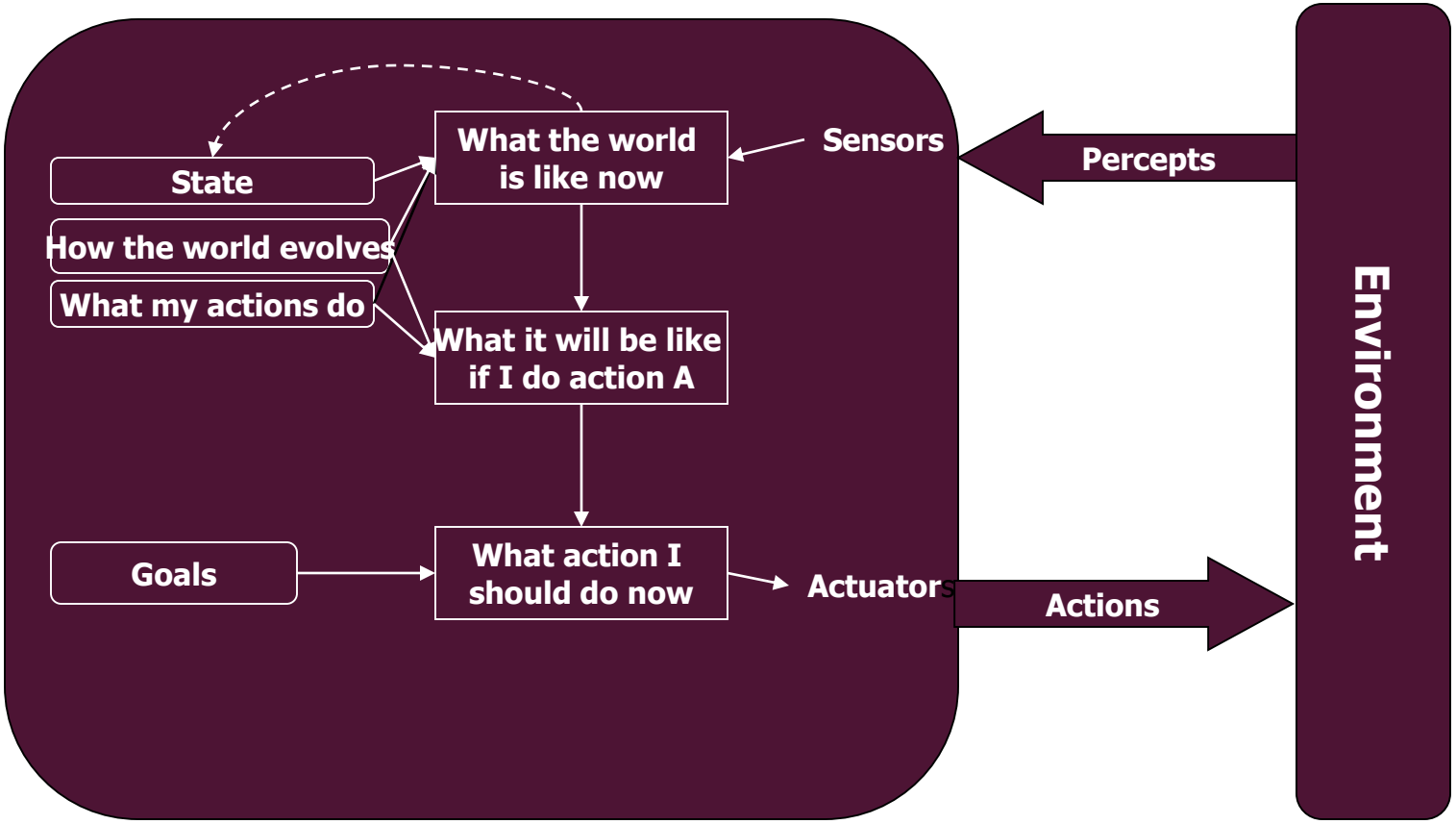
- Search strategies
  - Uninformed Search Strategies
  - Informed Search Strategies

PROBLEM SOLVING AGENTS

- Problem solving agent
  - A kind of "goal based" agent
  - Finds sequences of actions that lead to desirable states.

- The algorithms are uninformed
  - No extra information about the problem other than the definition
    - No extra information
    - No heuristics (rules)

GOAL BASED AGENT

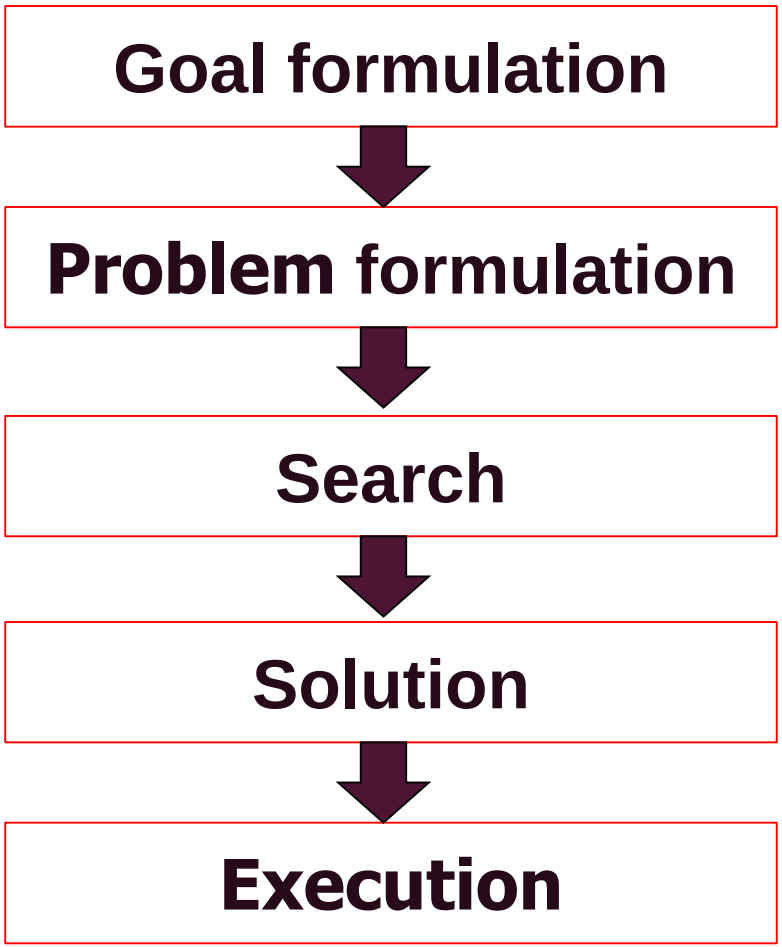


GOAL BASED AGENTS

- Assumes the problem environment is:
  - Static
    - The plan remains the same
  - Observable
    - Agent knows the initial state
  - Discrete
    - Agent can enumerate the choices
  - Deterministic
    - Agent can plan a sequence of actions such that each will lead to an intermediate state

- The agent carries out its plans with its eyes closed
  - Certain of what's going on
  - Open loop system

PROBLEM SOLVING AGENTS

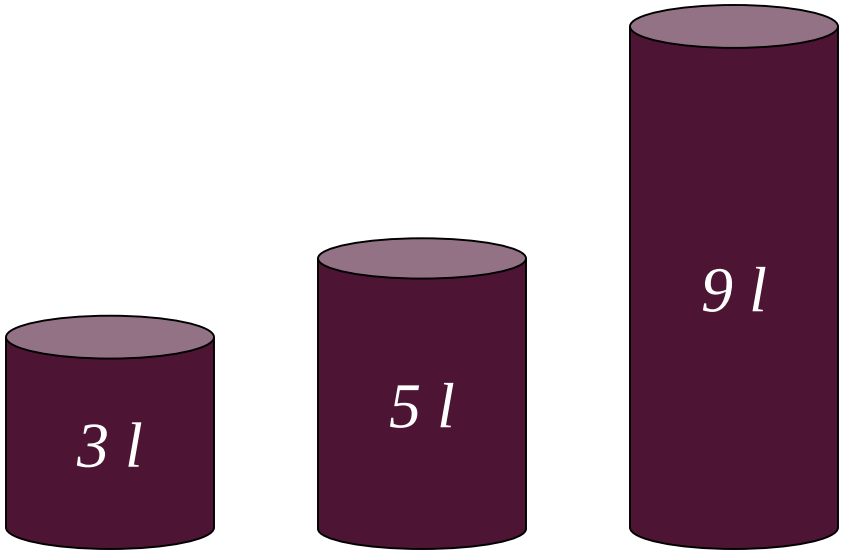


WELL DEFINED PROBLEMS FORMULATION

- Initial State
  - The agent starts in.
- Actions
  - A description of the possible actions available to the agent
- Transition model
  - A description of what each action does
- Goal test
  - It determines whether a given state is a goal state
- Path Cost
  - It assigns a numeric cost to each path

EXAMPLE: WATER POURING

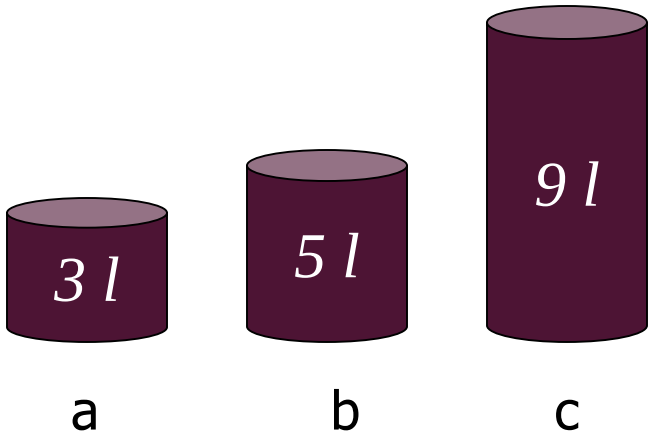
- Given a 9 gallon bucket, 5 gallon bucket and a 3 gallon bucket, how can we measure exactly 7 gallons into one bucket?
  - There are no markings on the bucket
  - You must fill each bucket completely



EXAMPLE: MEASURING PROBLEM!

- **(one possible) Solution:**

a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	
0	3	6	
3	3	6	
1	5	6	
0	5	7	goal



EXAMPLE: MEASURING PROBLEM!

- **Solution 1:**

a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	
0	3	6	
3	3	6	
1	5	6	
0	5	**7**	**goal**

- **Solution 2:**

a	b	c	
0	0	0	start
0	5	0	
3	2	0	
3	0	2	
3	5	2	
3	**0**	**7**	**goal**

EXAMPLE: WATER POURING

Measure 7 liters of water using a 3-liter, a 5-liter, and a 9-liter buckets.

- **Initial state:**
  - The buckets are empty
  - Represented by the tuple (0 0 0)

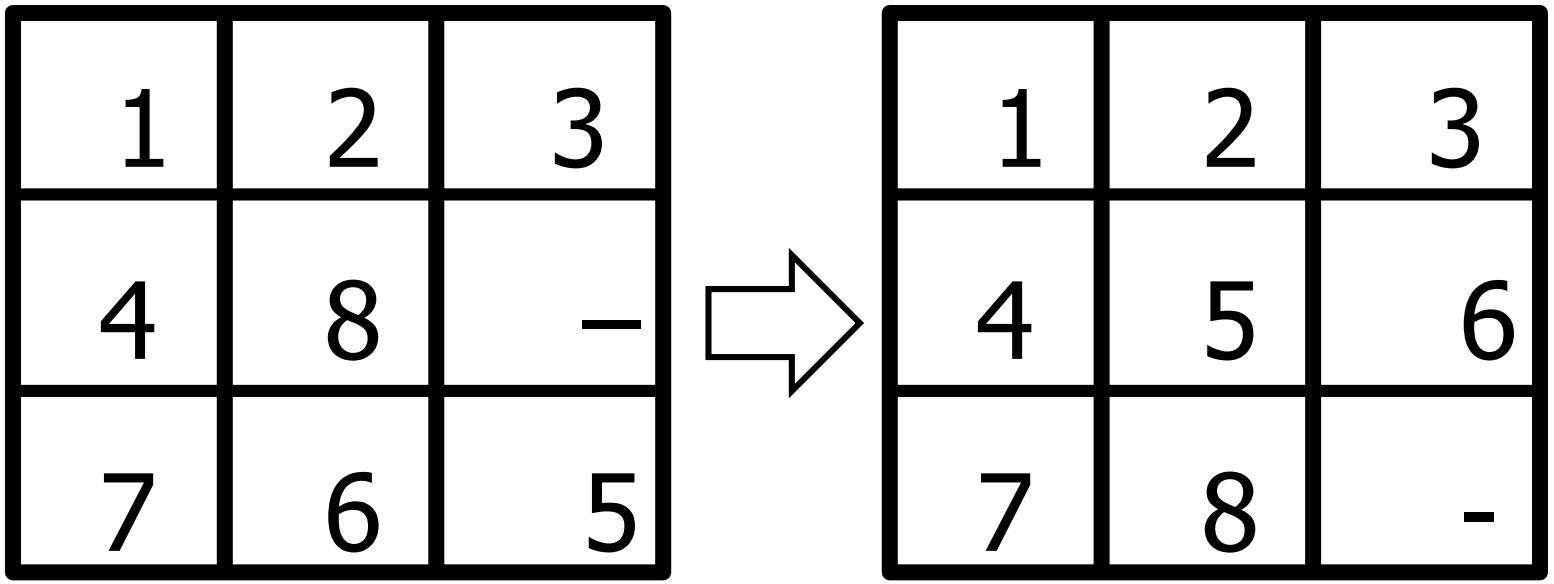
- **Formulate goal:** Have 7 liters of water
  in 9-liter bucket

- **Formulate problem:**
  - States: amount of water in the buckets
  - Operators: Fill bucket from source, empty bucket

- **Find solution:** sequence of operators that bring you
  from current state to the goal state

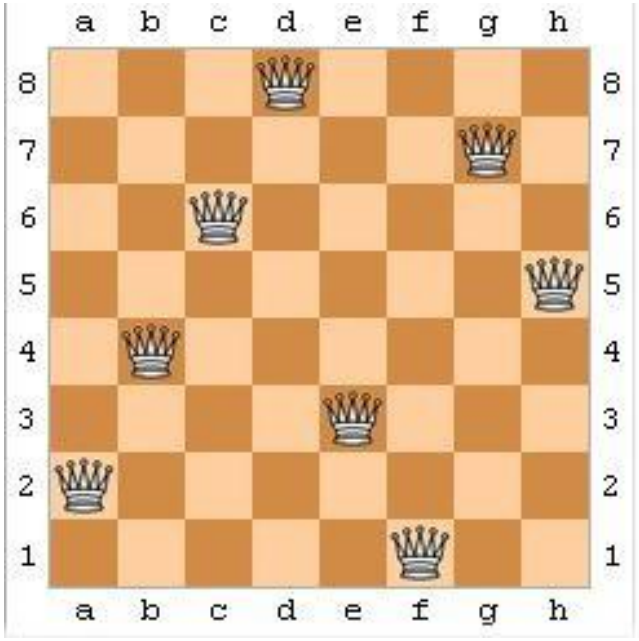
THE 8-PUZZLE

- **Initial State:** any state
- **Actions:** move blank left, right, up, down
- **Goal Test:** goal state (given)
- **Path Cost:** 1 per move

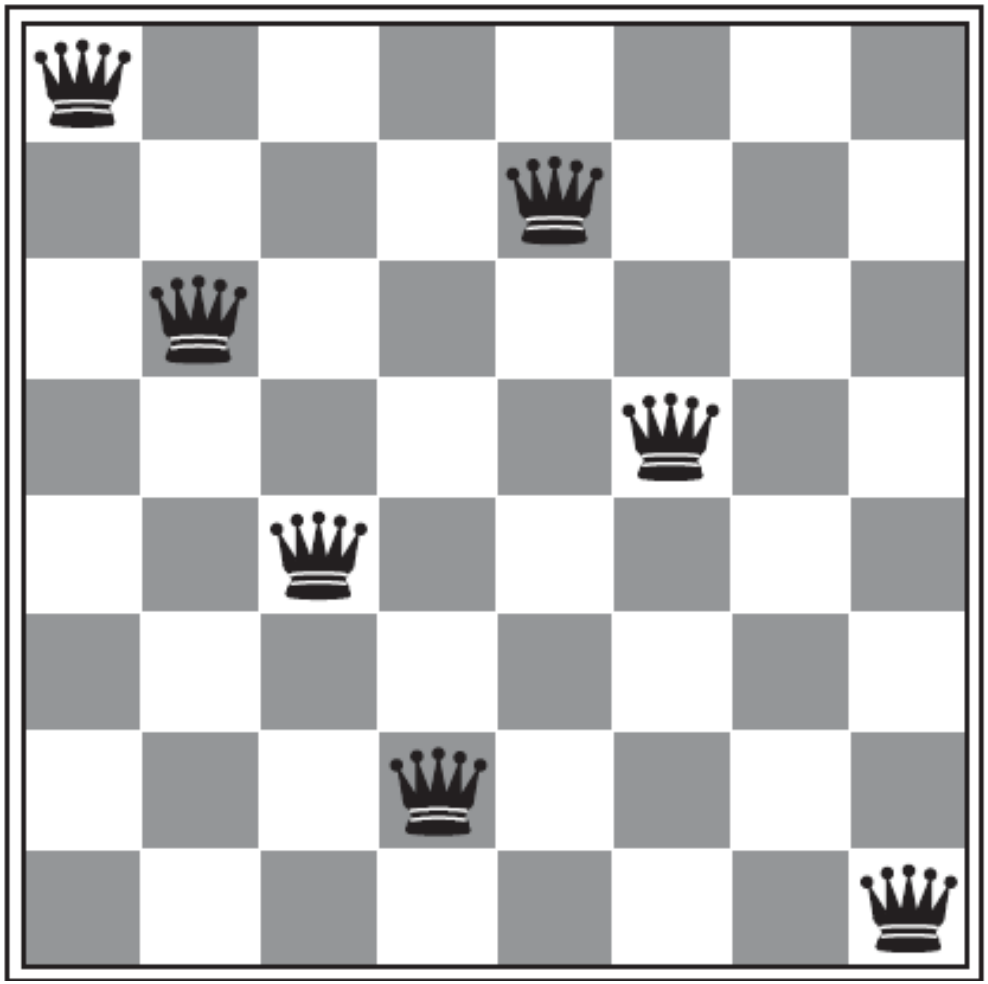


8- QUEENS PROBLEM

- The goal of the **8-queens problem** is to place eight queens on a chessboard
  - such that no queen attacks any other.
  - A queen attacks any piece in the same row, column or diagonal.



8- QUEENS PROBLEM



- It shows an attempted solution that fails.
- The queen in the rightmost column is attacked by the queen at the top left.

8- QUEENS PROBLEM

- **Initial State:** No queens on the board.
- **Actions:** Add a queen to any empty square.
- **Transition Model:** Returns the board with a queen added to the specified square.
- **Goal Test:** 8 queens are on the board, none attacked.

SEARCHING FOR SOLUTIONS

- Having formulated some problems…how do we solve them?
- Search through a state space
- Use a search tree that is generated with an initial state and successor functions that define the state space

SEARCHING FOR SOLUTIONS

- A ***state*** is (a representation of) a physical configuration
- A ***node*** is a data structure constituting part of a search tree
  - Includes parent, children, depth, path cost
- States do not have children, depth, or path cost
- The EXPAND function creates new nodes, filling in the various fields and using the SUCCESSOR function of the problem to create the corresponding states

UNINFORMED SEARCH STRATEGIES

- ***Uninformed*** strategies use only the information available in the problem definition
  - Also known as blind searching

- Breadth-first search
- Uniform-cost search
- Depth-first search
- Depth-limited search
- Iterative deepening search

COMPARING UNINFORMED SEARCH STRATEGIES

- Completeness
  - Will a solution always be found if one exists?
- Time
  - How long does it take to find the solution?
  - Often represented as the number of nodes searched
- Space
  - How much memory is needed to perform the search?
  - Often represented as the maximum number of nodes stored at once
- Optimal
  - Will the optimal (least cost) solution be found?

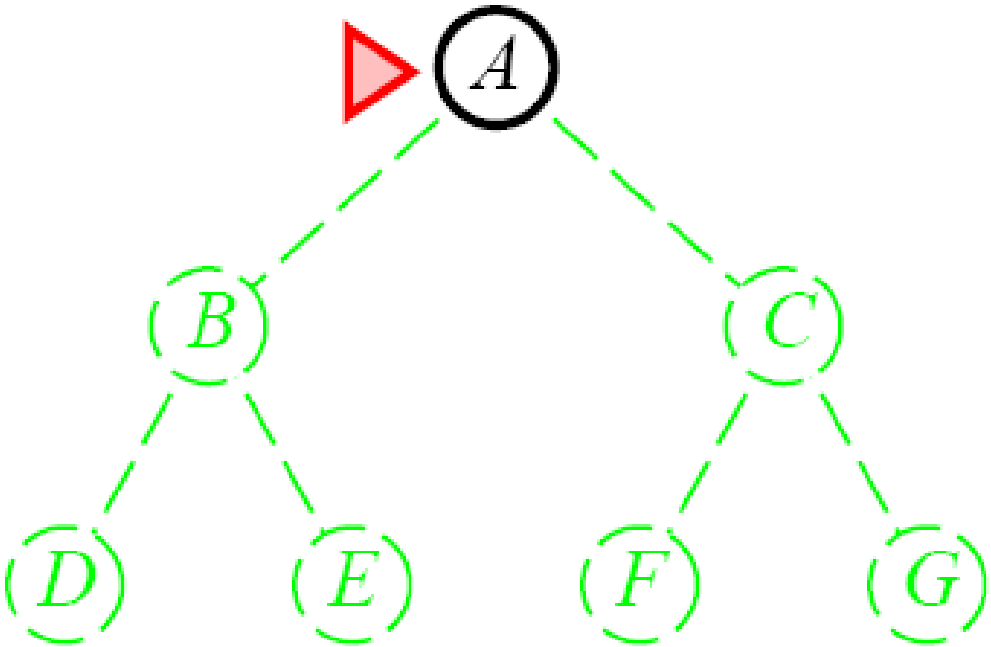
BREADTH-FIRST SEARCH

- Breadth-first search is the most common search strategy for traversing a tree or graph.
- This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.
- BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level.
- The breadth-first search algorithm is an example of a general-graph search algorithm.
- Breadth-first search implemented using FIFO queue data structure.

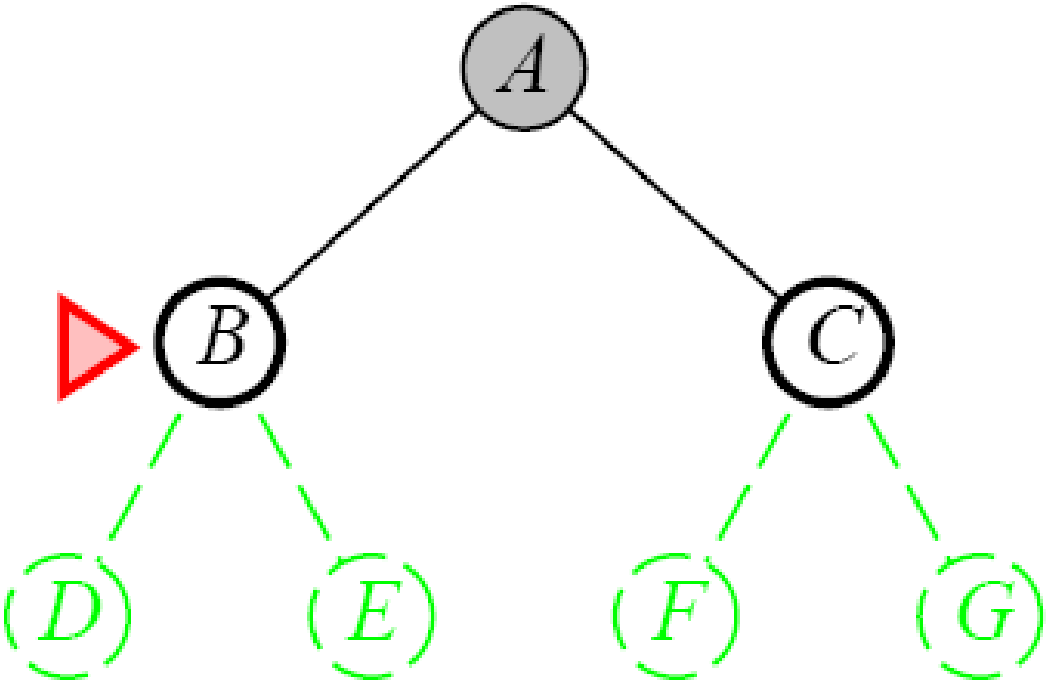
CONTD….

- **Advantages:**
  - BFS will provide a solution if any solution exists.
  - If there are more than one solutions for a given problem, then BFS will provide the minimal solution which requires the least number of steps.
- **Disadvantages:**
  - It requires lots of memory since each level of the tree must be saved into memory to expand the next level.
  - BFS needs lots of time if the solution is far away from the root node.

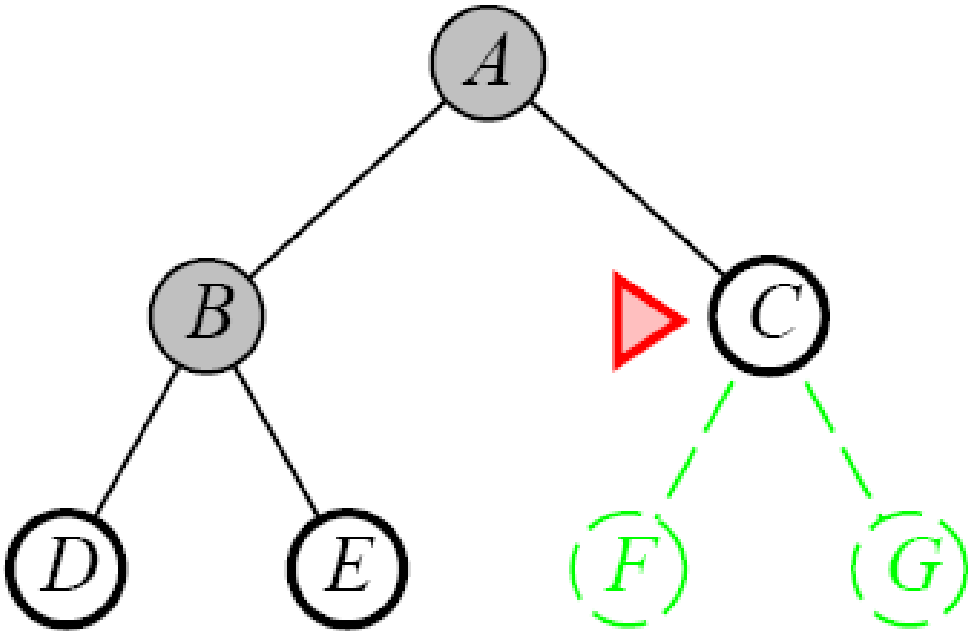
BREADTH-FIRST SEARCH



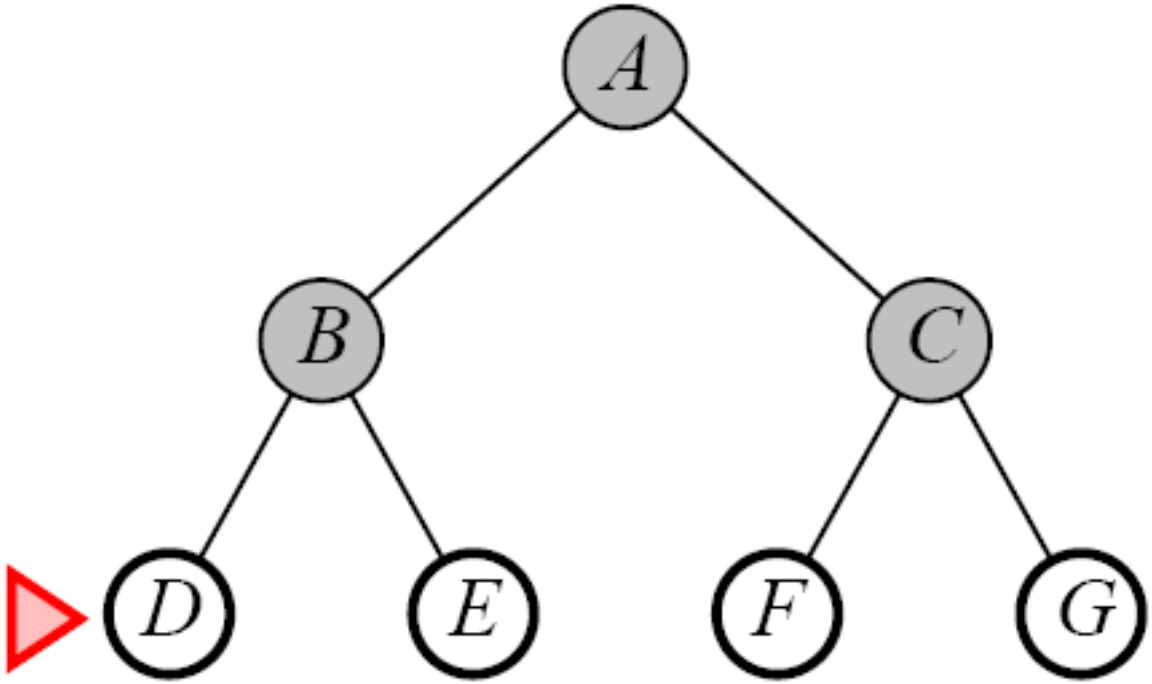
BREADTH-FIRST SEARCH



BREADTH-FIRST SEARCH



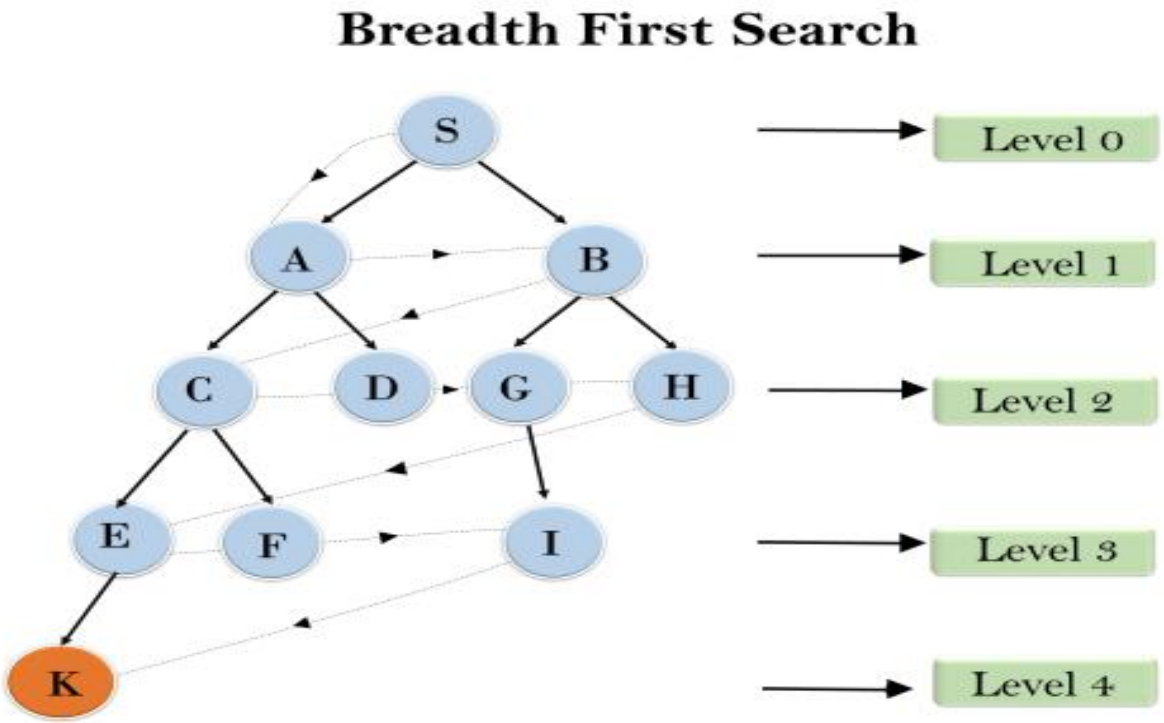
BREADTH-FIRST SEARCH



EXAMPLE#01

In the below tree structure, we have shown the traversing of the tree using BFS algorithm from the root node S to goal node K. BFS search algorithm traverse in layers, so it will follow the path which is shown by the dotted arrow, and the traversed path will be:

```
S---> A--->B---->C--->D---->G--->H--->E---->F---->I---->K
```

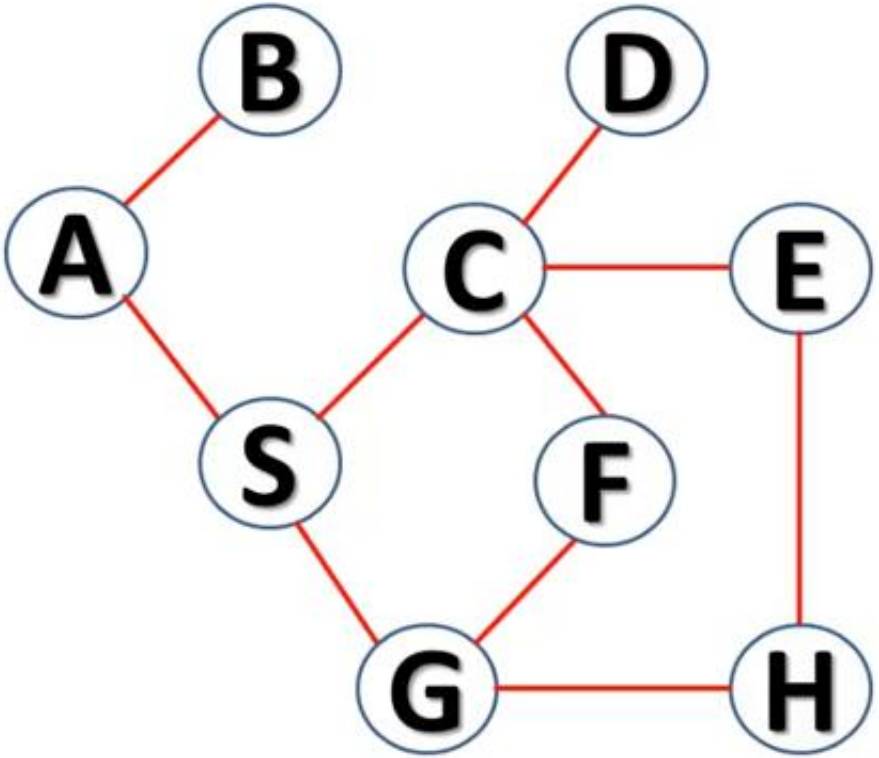


EXAMPLE#02

For the given graph, find the path, number of node tested and expanded node using Breadth First Search (BFS) Algorithm

Starting State: A

Goal State: E



SOLUTION

OPEN LIST	CLOSE LIST
[A]	[ ]
[B,S]	[A]
[S]	[A,B]
[C,G]	[A,B,S]
[G,D,E,F]	[A,B,S,C]
[D,E,F,H]	[A,B,S,C,G]
[E,F,H]	[A,B,S,C,G,D]
[F,H]	[A,B,S,C,G,D,E]
	GOAL ACHIEVED

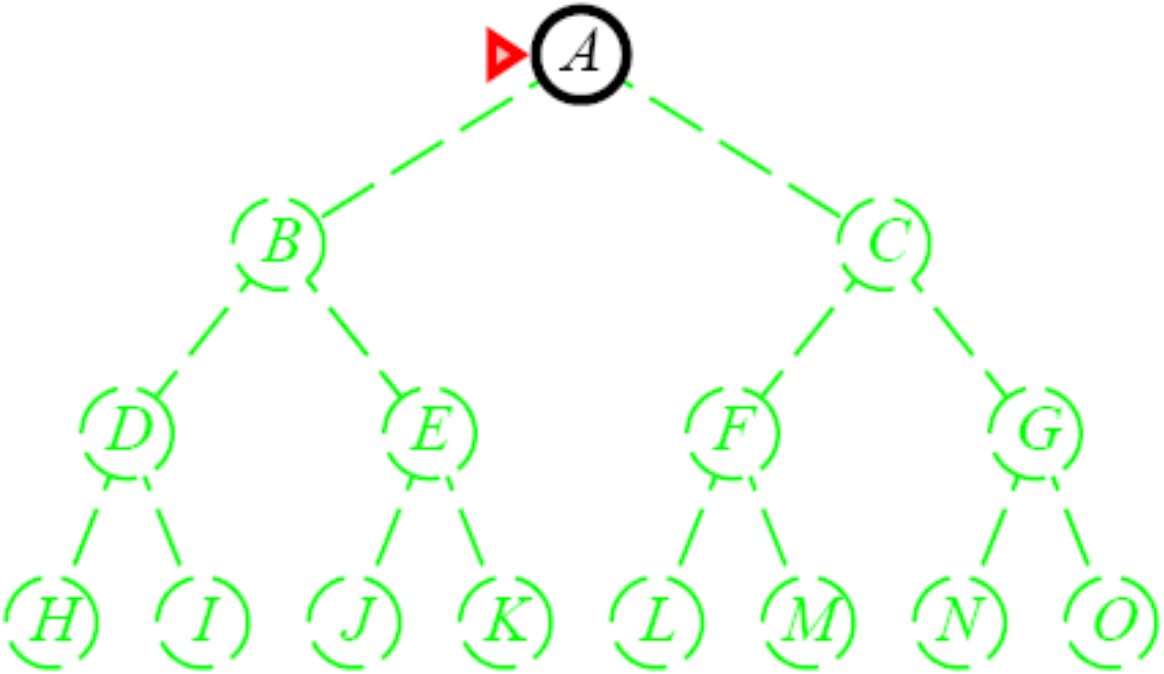
DEPTH-FIRST SEARCH

- Depth-first search is a recursive algorithm for traversing a tree or graph data structure.
- It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.
- DFS uses a stack data structure for its implementation.
- The process of the DFS algorithm is similar to the BFS algorithm.

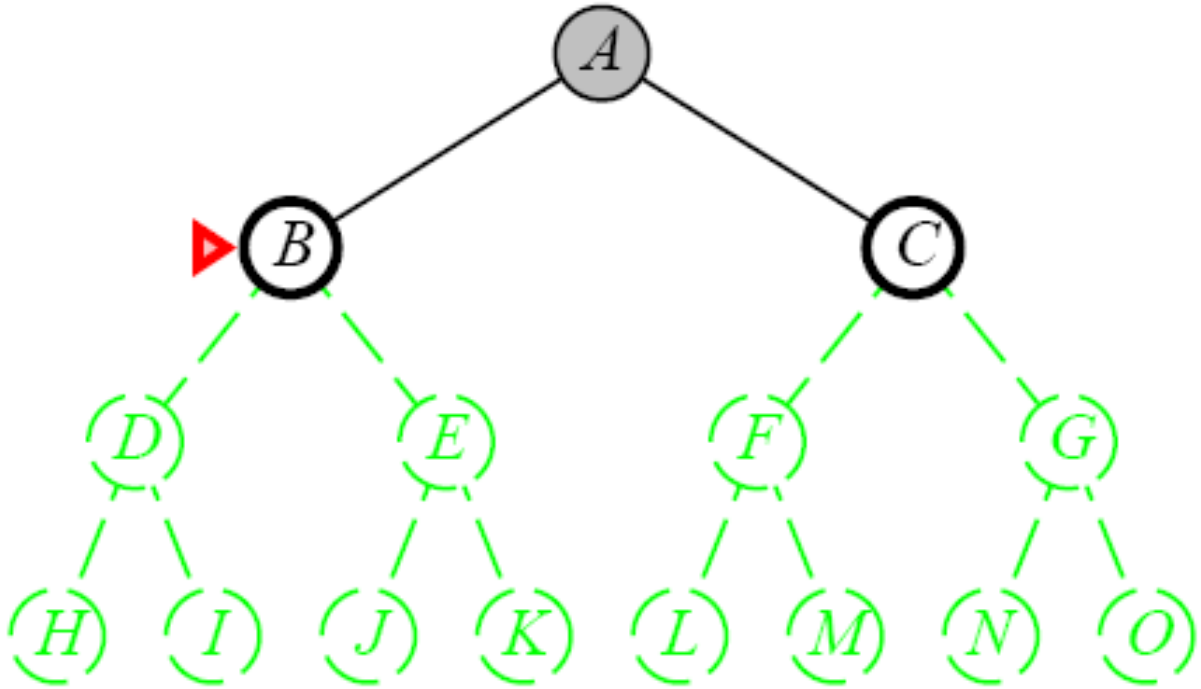
DEPTH-FIRST SEARCH

- **Advantage:**
  - DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.
  - It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).
- **Disadvantage:**
  - There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
  - DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

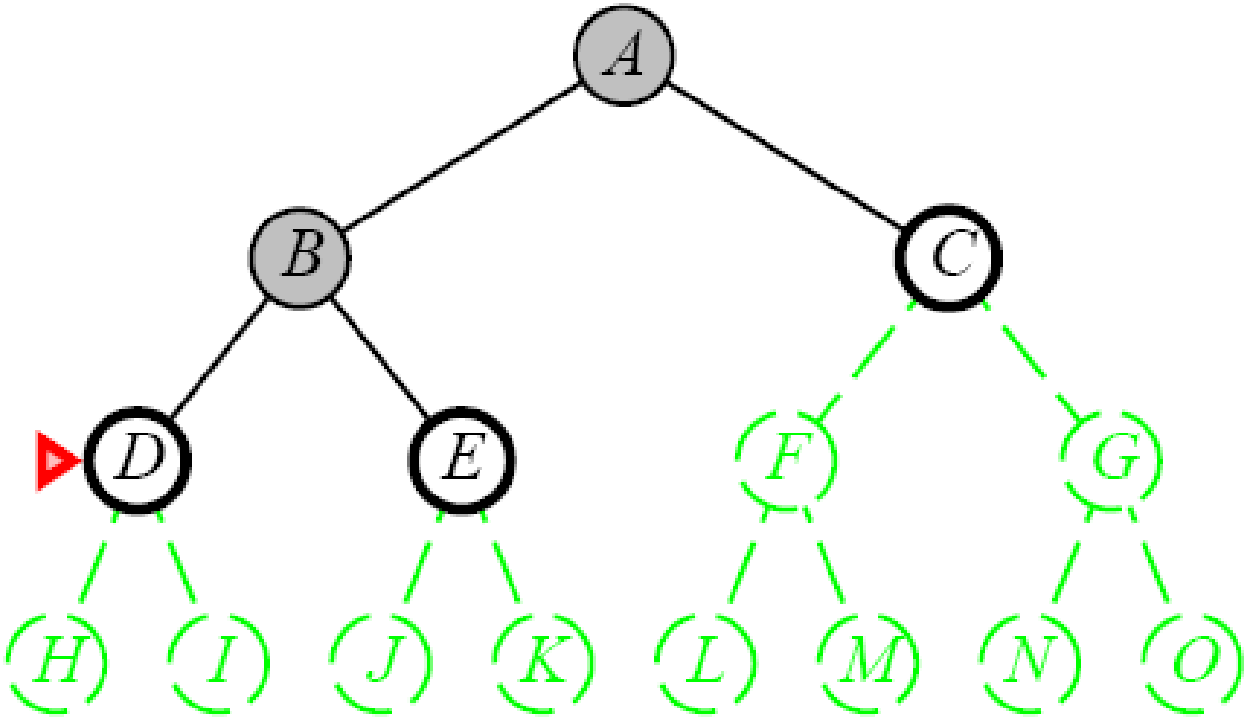
DEPTH-FIRST SEARCH



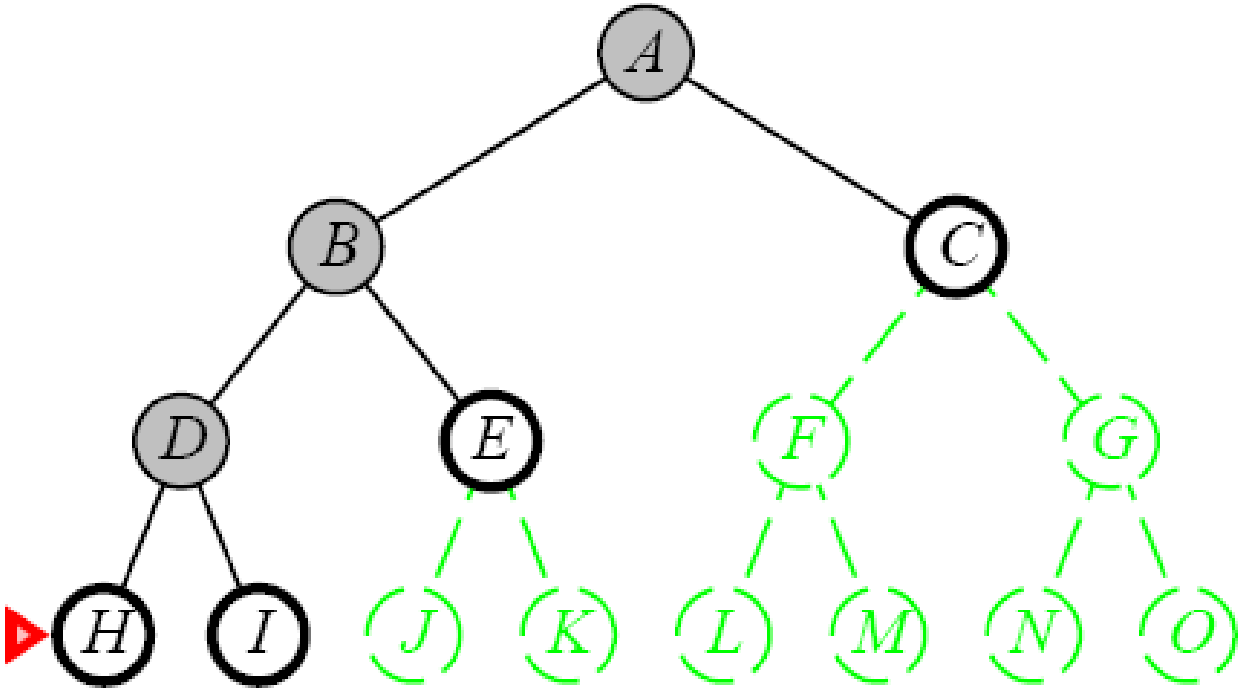
DEPTH-FIRST SEARCH



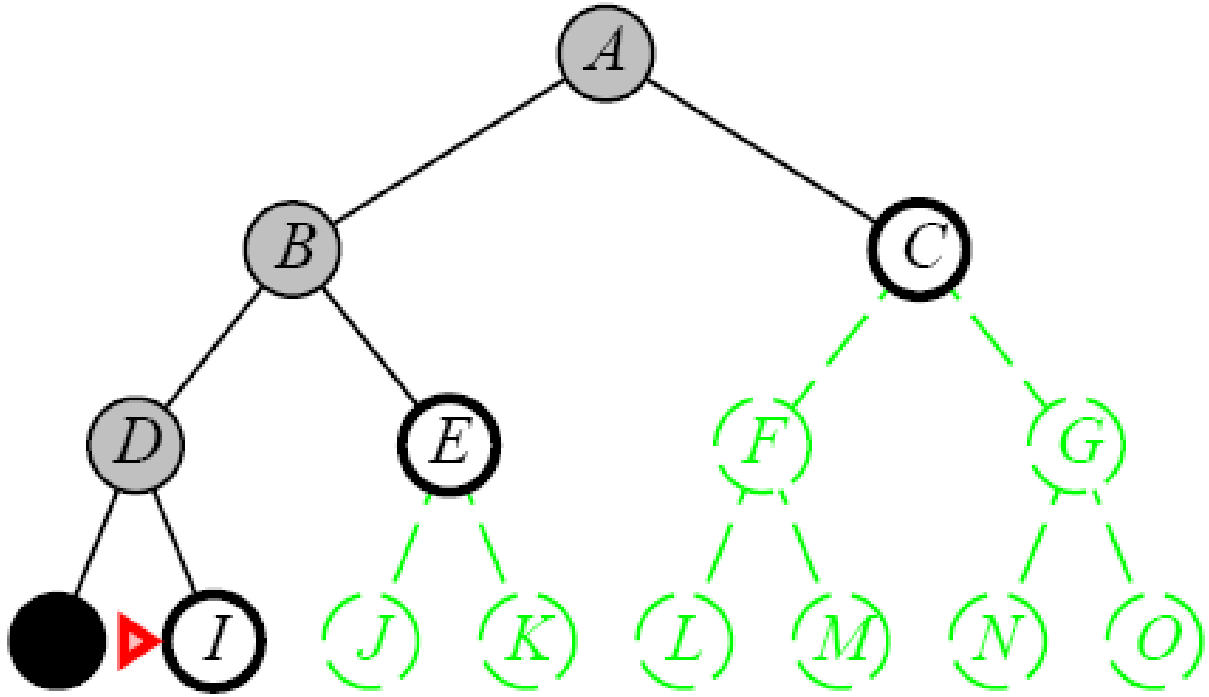
DEPTH-FIRST SEARCH



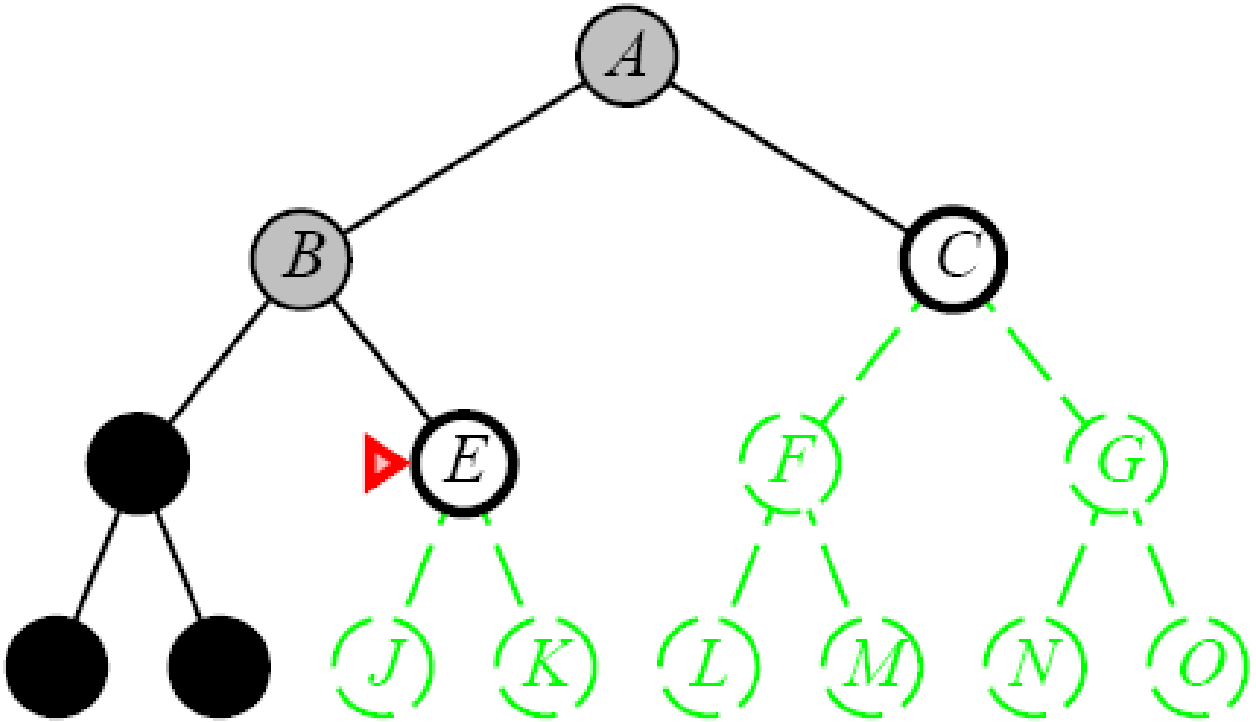
DEPTH-FIRST SEARCH



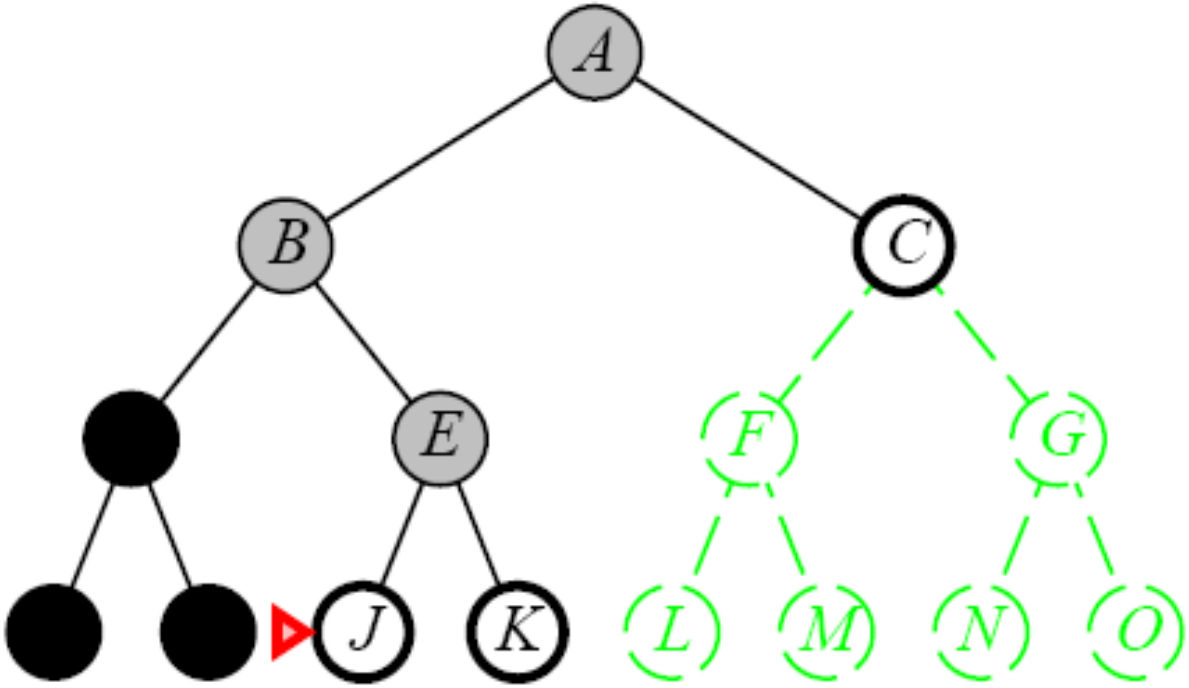
DEPTH-FIRST SEARCH



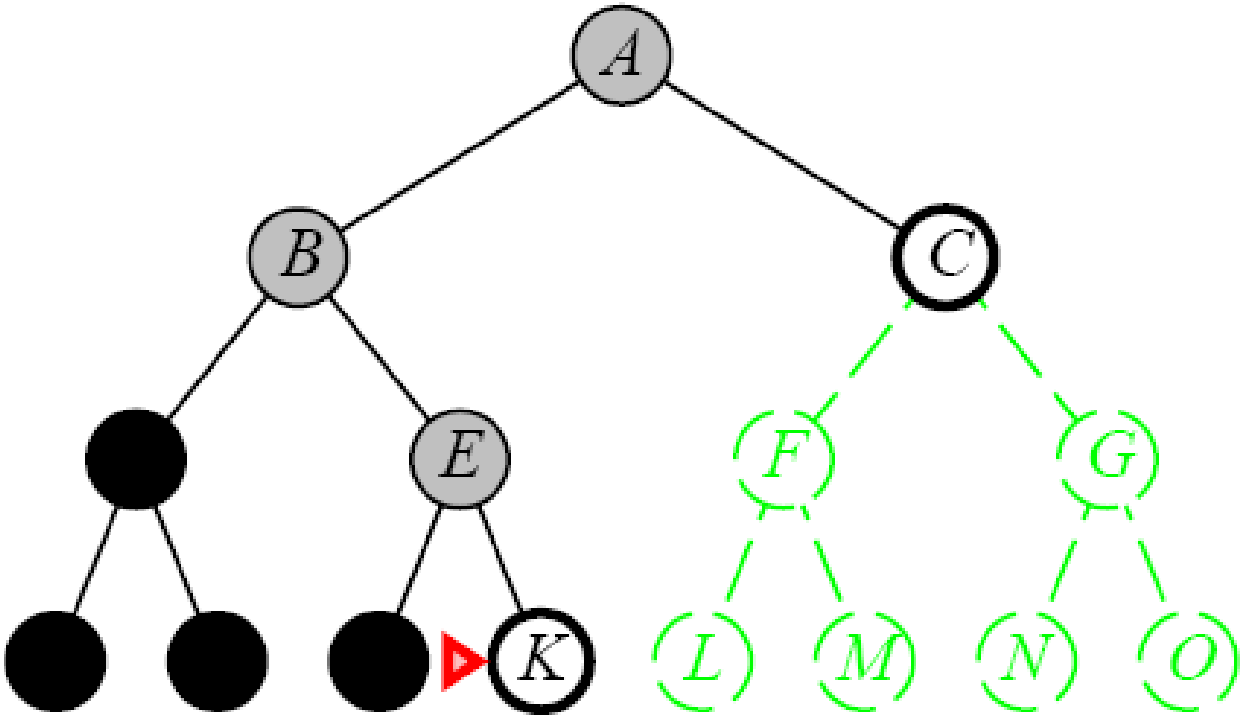
DEPTH-FIRST SEARCH



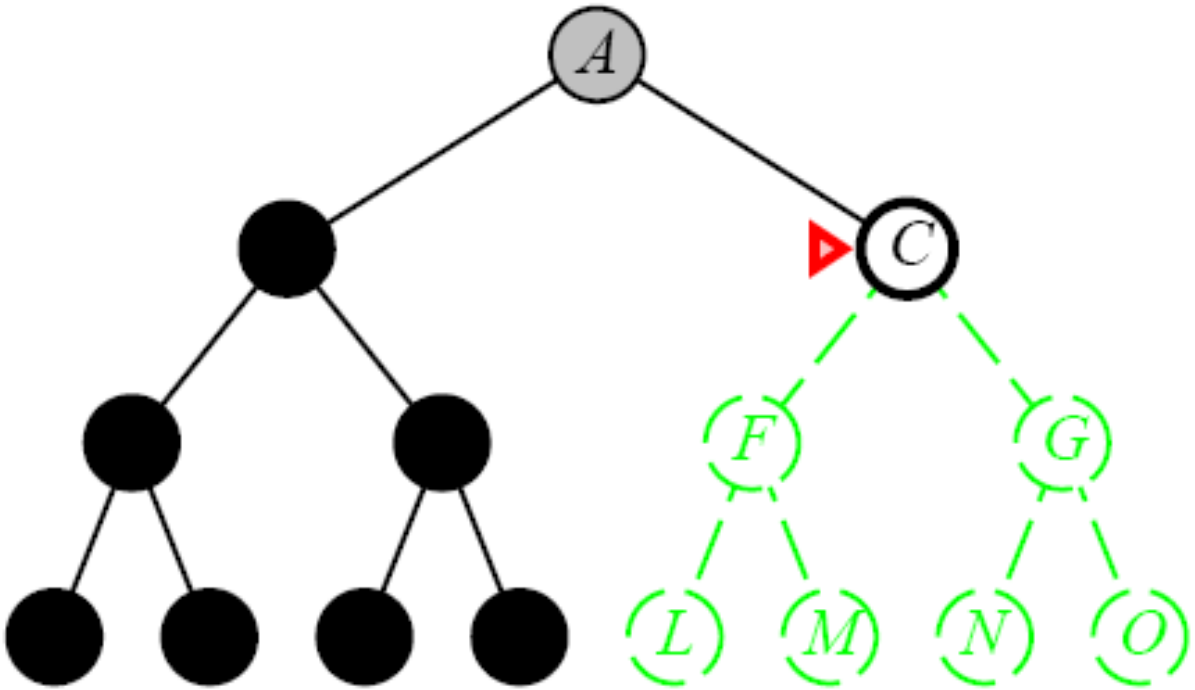
DEPTH-FIRST SEARCH



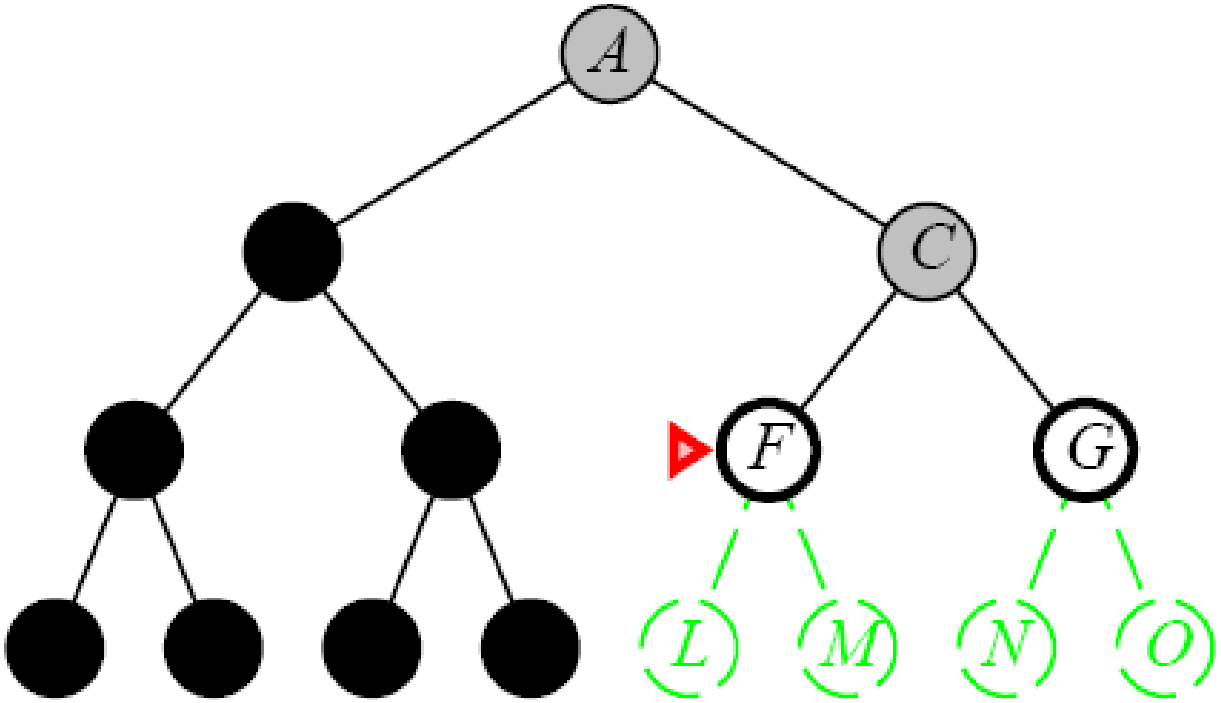
DEPTH-FIRST SEARCH



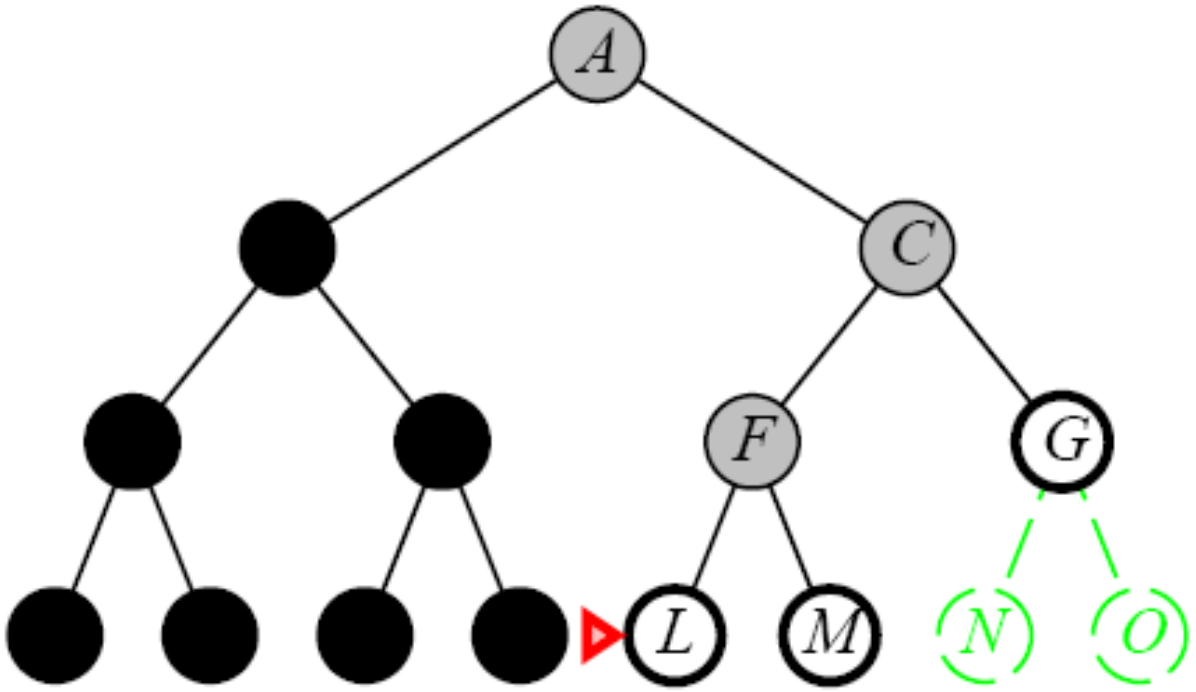
DEPTH-FIRST SEARCH



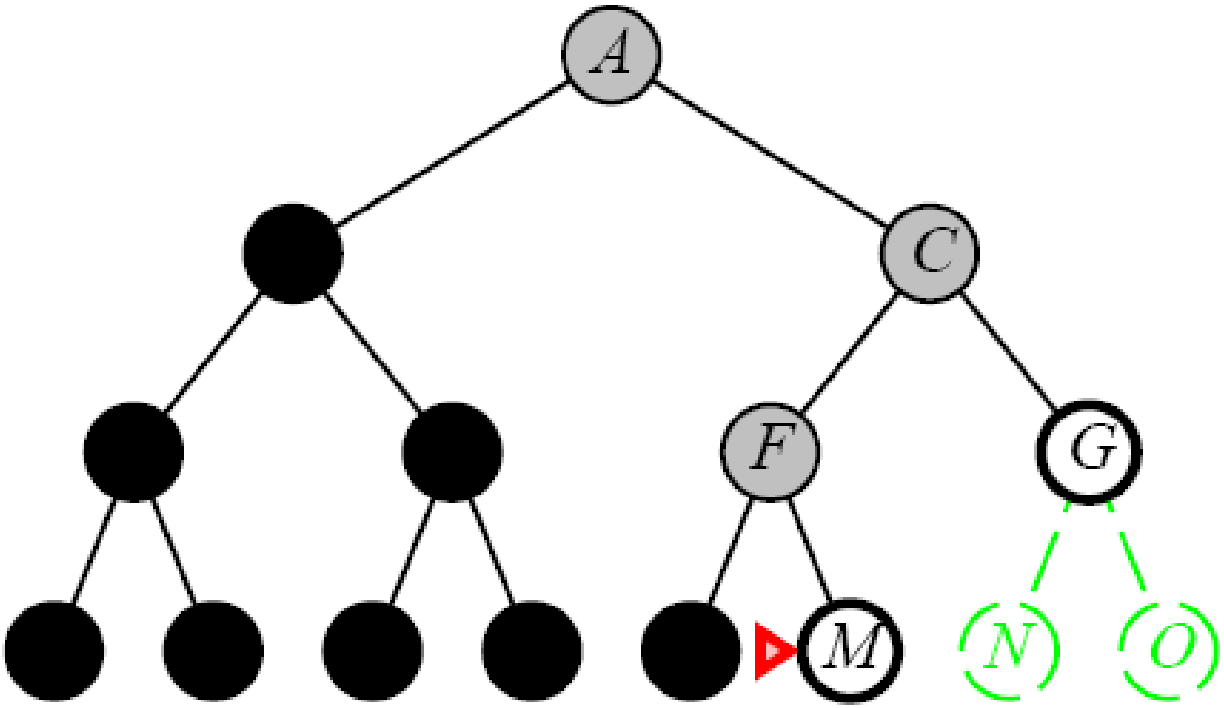
DEPTH-FIRST SEARCH



DEPTH-FIRST SEARCH



DEPTH-FIRST SEARCH

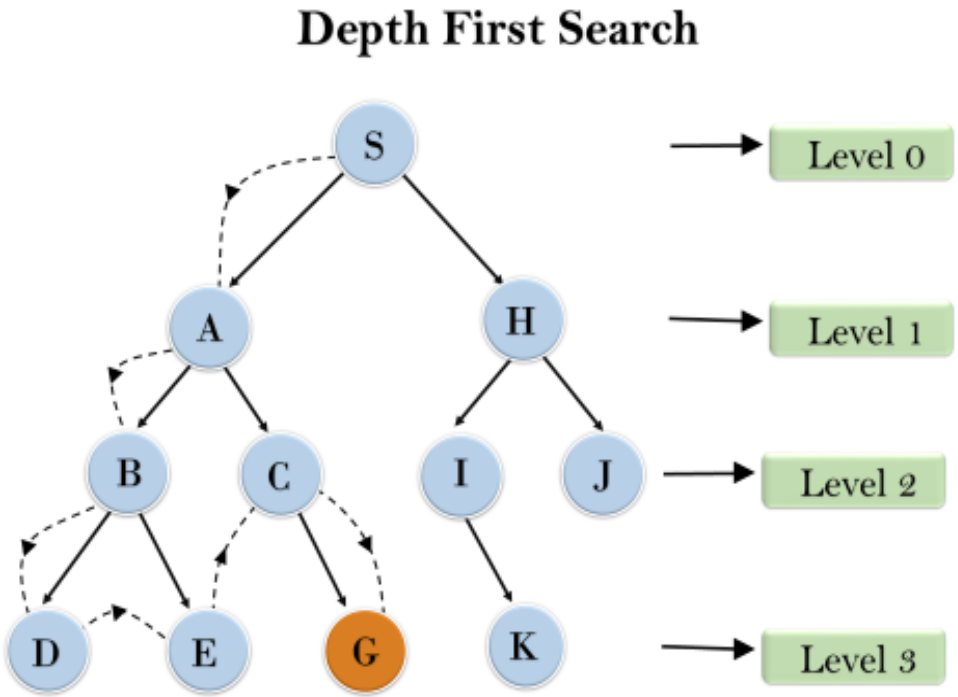


EXAMPLE#01

In the below search tree, we have shown the flow of depth-first search, and it will follow the order as:

Root node--->Left node ----> right node.

It will start searching from root node S, and traverse A, then B, then D and E, after traversing E, it will backtrack the tree as E has no other successor and still goal node is not found. After backtracking it will traverse node C and then G, and here it will terminate as it found goal node.

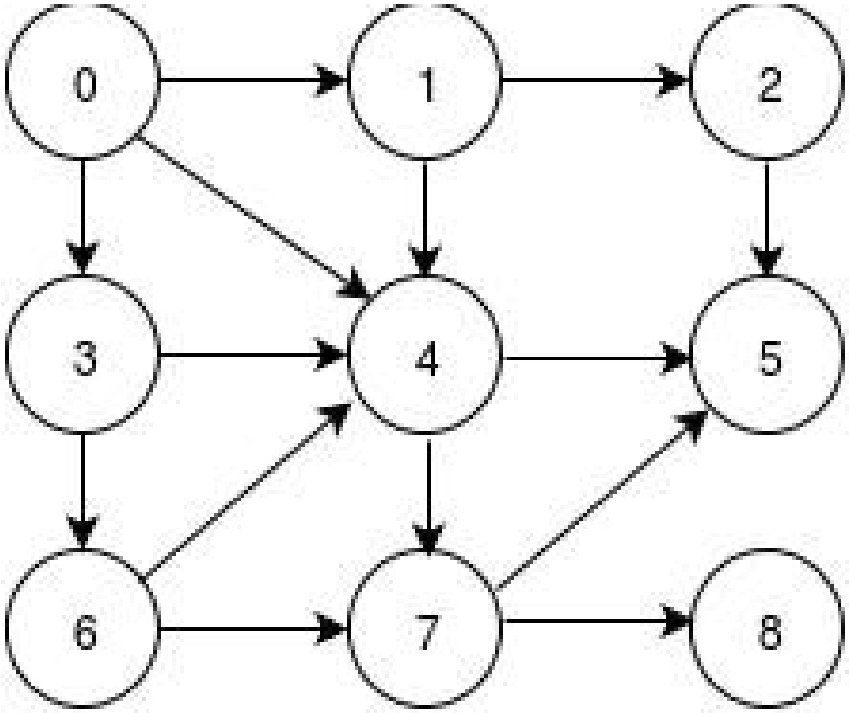


EXAMPLE#02

For the given graph, find the path, number of node tested and expanded node using Depth First Search (DFS) Algorithm

Starting State: 0 (Zero)

Goal State: 8 (Eight)



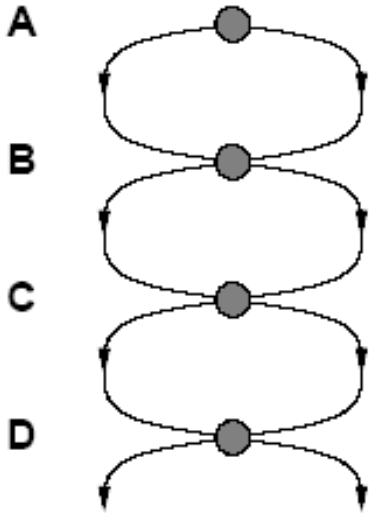
SOLUTION

OPEN LIST	CLOSE LIST
[0]	[ ]
[1,3,4]	[0]
[2,3,4]	[0,1]
[5,3,4]	[0,1,2]
[3,4]	[0,1,2,5]
[6,4]	[0,1,2,5,3]
[7,4]	[0,1,2,5,3,6]
[8,4]	[0,1,2,5,3,6,7]
[4]	[0,1,2,5,3,6,7,8] GOAL ACHIEVED

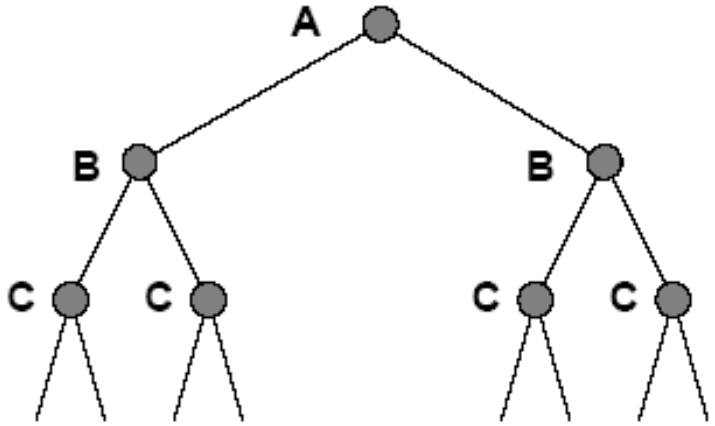
AVOIDING REPEATED STATES

- Complication of wasting time by expanding states that have already been encountered and expanded before
  - Failure to detect repeated states can turn a linear problem into an exponential one
- Sometimes, repeated states are unavoidable
  - Problems where the actions are reversible
    - Route finding
    - Sliding blocks puzzles

AVOIDING REPEATED STATES



State Space



Search Tree

UNIFORM-COST SEARCH

- Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph.
- This algorithm comes into play when a different cost is available for each edge.
- The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost.
- Uniform-cost search expands nodes according to their path costs form the root node.
- It can be used to solve any graph/tree where the optimal cost is in demand.
- A uniform-cost search algorithm is implemented by the priority queue.
- Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.

CONTD….

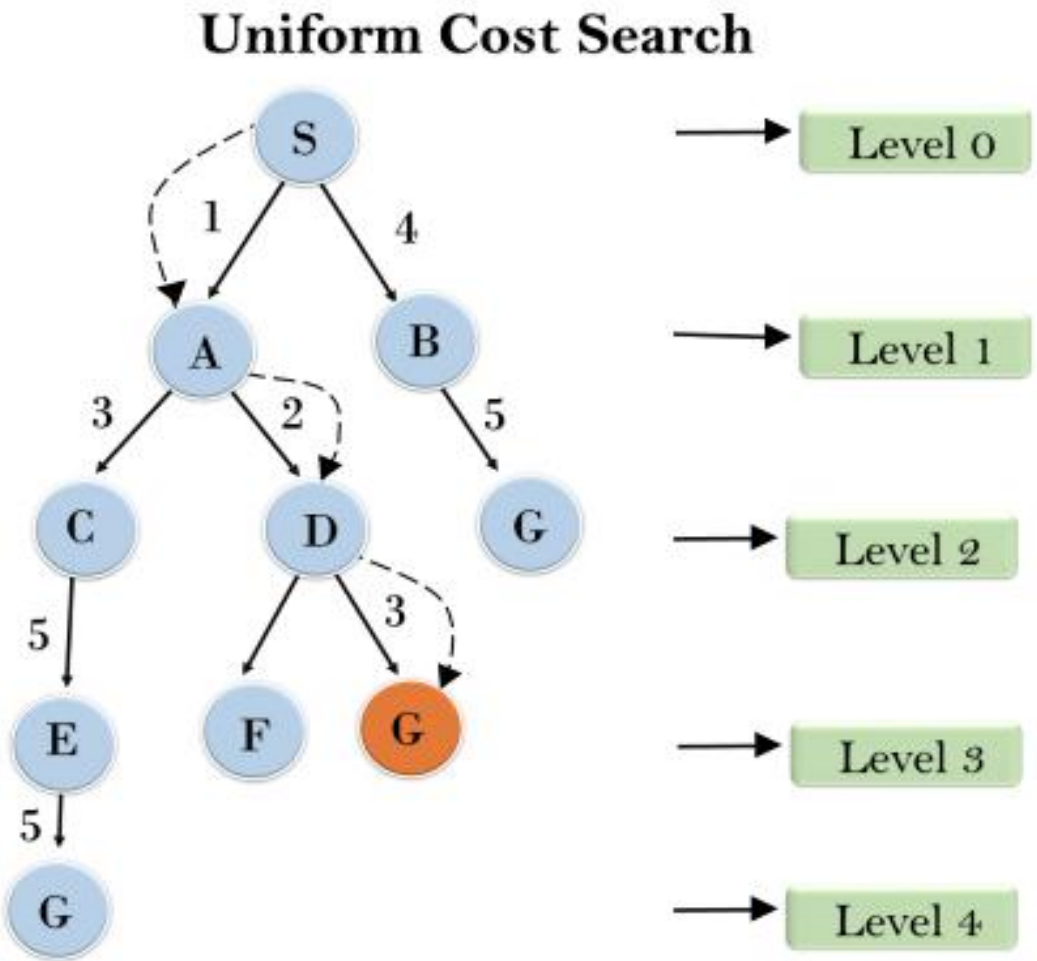
- **Advantages:**

Uniform cost search is optimal because at every state the path with the least cost is chosen.

- **Disadvantages:**

It does not care about the number of steps involve in searching and only concerned about path cost. Due to which this algorithm may be stuck in an infinite loop.

EXAMPLE


Uniform Cost Search
S
A
B
C
D
G
E
F
G
G
1
4
3
2
5
5
3
5
Level 0
Level 1
Level 2
Level 3
Level 4

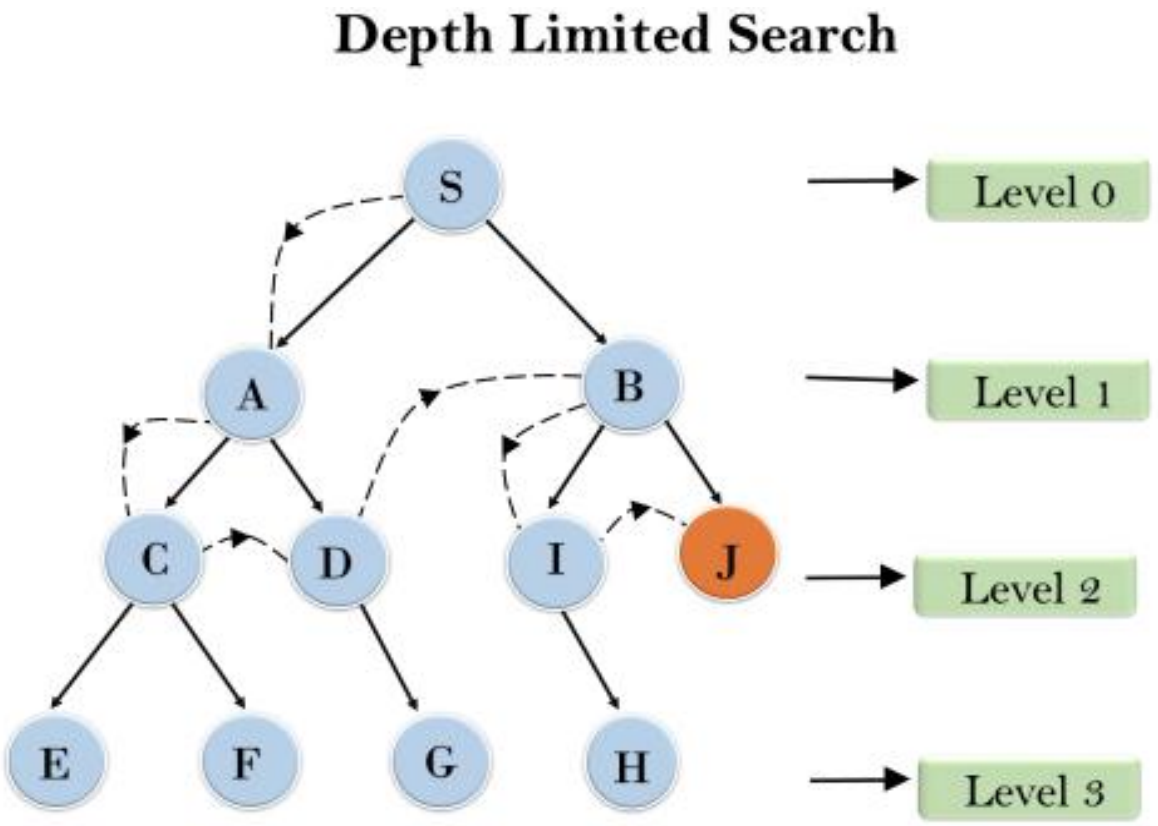
DEPTH-LIMITED SEARCH

- A depth-limited search algorithm is similar to depth-first search with a predetermined limit.
- Depth-limited search can solve the drawback of the infinite path in the Depth-first search.
- In this algorithm, the node at the depth limit will treat as it has no successor nodes further.
- Depth-limited search can be terminated with two Conditions of failure:
  - Standard failure value: It indicates that problem does not have any solution.
  - Cutoff failure value: It defines no solution for the problem within a given depth limit.

CONTD….

- **Advantages:**
  - Depth-limited search is Memory efficient.
- **Disadvantages:**
  - Depth-limited search also has a disadvantage of incompleteness.
  - It may not be optimal if the problem has more than one solution.

EXAMPLE


Depth Limited Search
S
A
B
C
D
I
J
E
F
G
H
Level 0
Level 1
Level 2
Level 3

ITERATIVE DEEPENING SEARCH

- The iterative deepening algorithm is a combination of DFS and BFS algorithms. This search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found.
- This algorithm performs depth-first search up to a certain "depth limit", and it keeps increasing the depth limit after each iteration until the goal node is found.
- This Search algorithm combines the benefits of Breadth-first search's fast search and depth-first search's memory efficiency.
- The iterative search algorithm is useful uninformed search when search space is large, and depth of goal node is unknown.

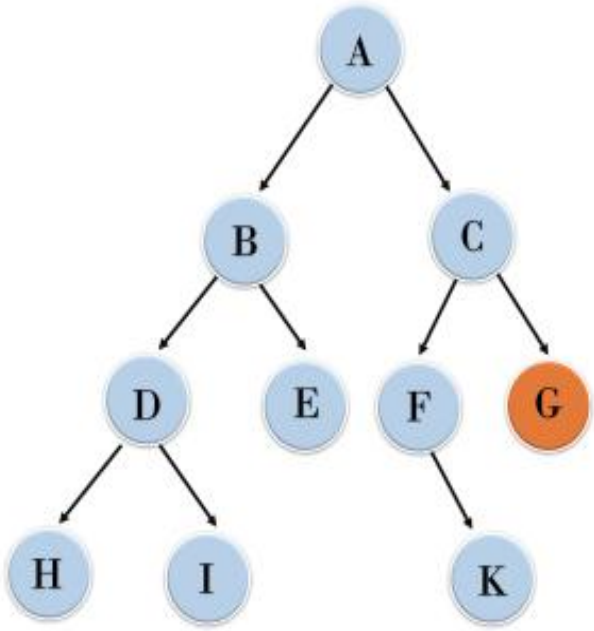
CONTD….

- **Advantages:**
  - It combines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.
- **Disadvantages:**
  - The main drawback of IDDFS is that it repeats all the work of the previous phase.

CONTD….

Following tree structure is showing the iterative deepening search. IDS algorithm performs various iterations until it does not find the goal node. The iteration performed by the algorithm is given as:

Iterative deepening depth first search



Level 0

1'st Iteration-----> A

2'nd Iteration----> A, B, C

Level 1

3'rd Iteration------>A, B, D, E, C, F, G

4'th Iteration------>A, B, D, H, I, E, C, F, K, G

Level 2

In the fourth iteration, the algorithm will find the goal node.

Level 3

SUMMARY

- Uninformed search refers to a class of highly generic search strategies which use no problem-specific structure (other than the ability to generate successor states and evaluate costs and goal conditions).

- These algorithms aren't very effective for complex tasks, but they're good enough for some small problems.