UNIVERSITY INSTITUTE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGG.

Bachelor of Engineering (Computer Science & Engineering)

Principles of Artificial Intelligence (20CST-258)

Uninformed Search

DISCOVER . LEARN . EMPOWER

Outline

- Search Algorithms
- Search Algorithm Terminologies
- Properties of Search Algorithms
- Type of Search Strategies
  - Uninformed
  - Informed

Search Algorithm Terminologies

- **Search:** Searching is a step by step procedure to solve a search-problem in a given search space. A search problem can have three main factors:
    - **Search Space:** Search space represents a set of possible solutions, which a system may have.
    - **Start State:** It is a state from where agent begins the search.
    - **Goal test:** It is a function which observe the current state and returns whether the goal state is achieved or not.
- **Search tree:** A tree representation of search problem is called Search tree. The root of the search tree is the root node which is corresponding to the initial state.
- **Actions:** It gives the description of all the available actions to the agent.
- **Transition model:** A description of what each action do, can be represented as a transition model.
- **Path Cost:** It is a function which assigns a numeric cost to each path.
- **Solution:** It is an action sequence which leads from the start node to the goal node.
- **Optimal Solution:** If a solution has the lowest cost solution among all solutions.

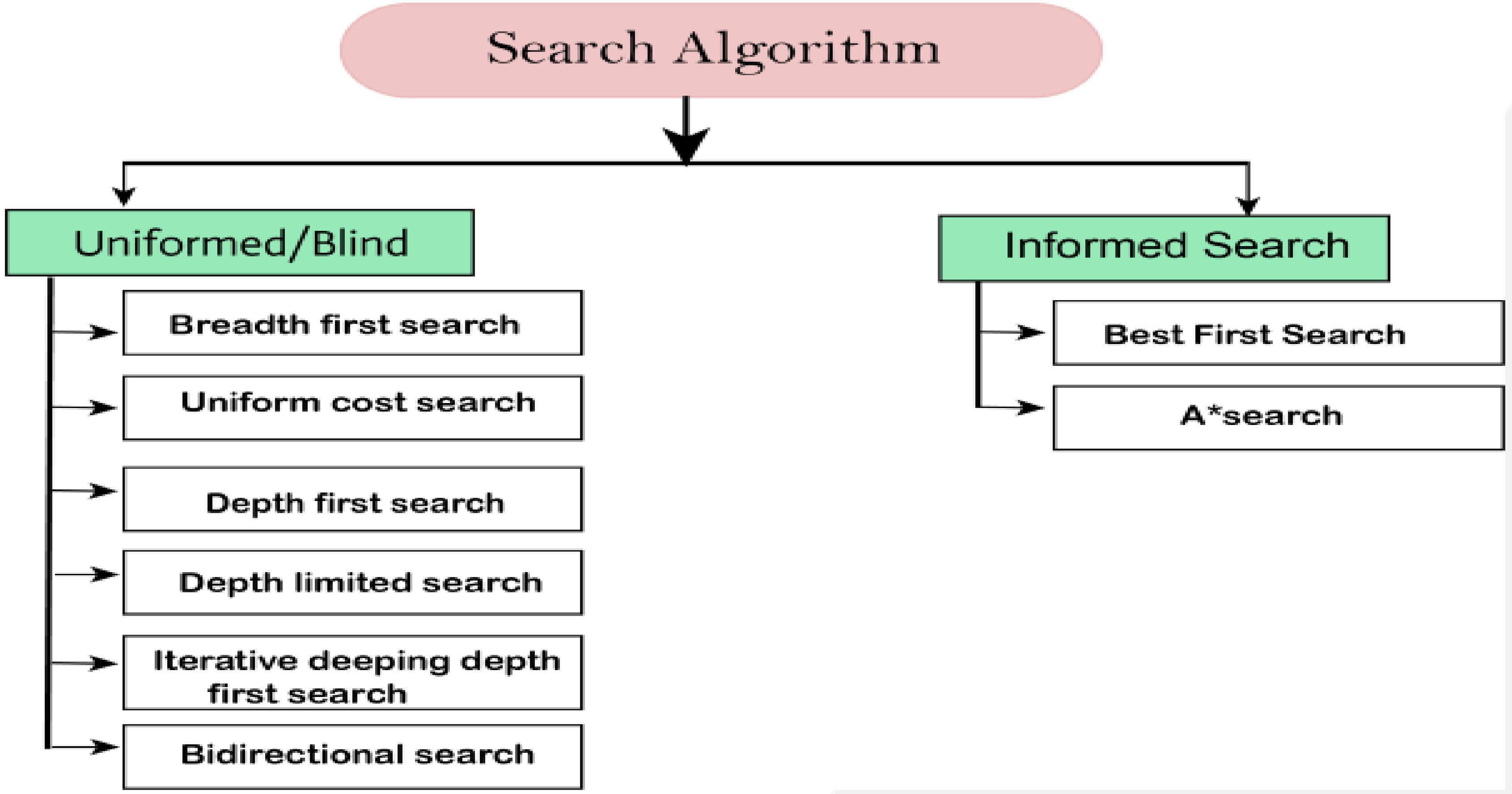
Properties of Search Algorithms

- **Completeness:** A search algorithm is said to be complete if it guarantees to return a solution if at least any solution exists for any random input.
- **Optimality:** If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions, then such a solution for is said to be an optimal solution.
- **Time Complexity:** Time complexity is a measure of time for an algorithm to complete its task.
- **Space Complexity:** It is the maximum storage space required at any point during the search, as the complexity of the problem.

Type of Search Strategies

- Uninformed search (Also known as blind search) –Uninformed search algorithms have no additional information on the goal node other than the one provided in the problem definition
- Informed search (Also known as heuristic search) – Search strategies know whether one state is more promising than another.

Types of Search Algorithms



The Uninformed Search

- Uninformed search algorithms have no additional information on the goal node other than the one provided in the problem definition.
- The plans to reach the goal state from the start state differ only by the order and/or length of actions.
- Uninformed search is also called Blind search.

These types of algorithms will have:

- A problem graph, containing the start node S and the goal node G.
- A strategy, describing the manner in which the graph will be traversed to get to G.
- A fringe, which is a data structure used to store all the possible states (nodes) that you can go from the current states.
- A tree that results while traversing to the goal node. A solution plan, which the sequence of nodes from S to G.

Uninformed Search Types

- It can be divided into five main types:
    - Depth-first search
    - Breadth-first search
    - Iterative deepening depth-first search
    - Uniform cost search
    - Bidirectional Search

Depth First Search

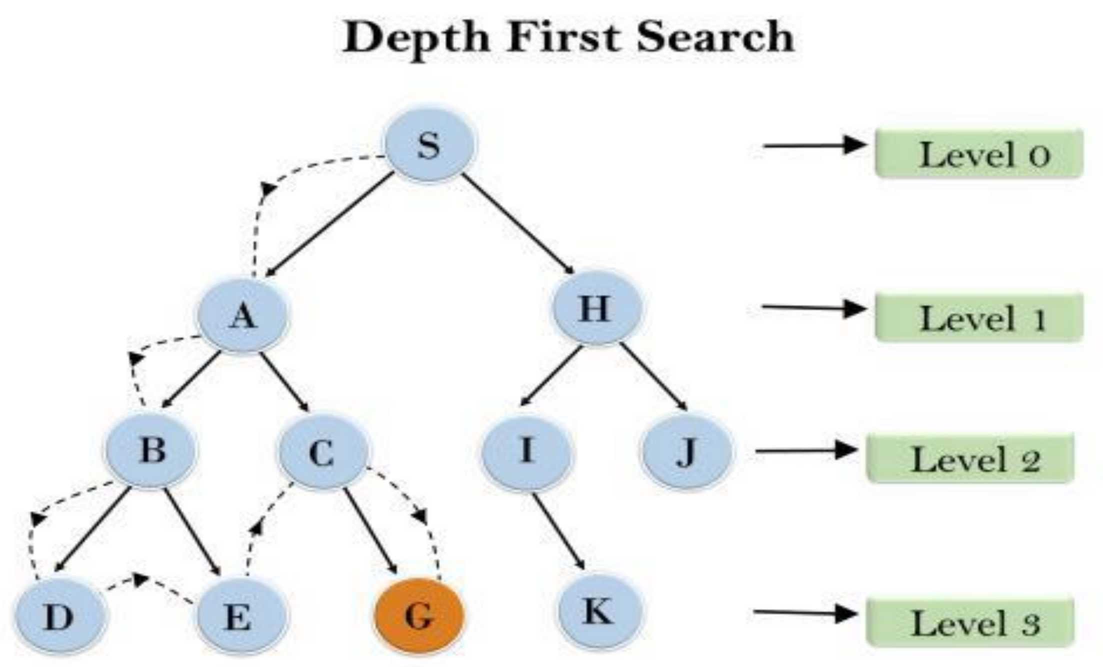
- The DFS algorithm is a recursive algorithm that uses the idea of backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking.
- Here, the word backtrack means that when you are moving forward and there are no more nodes along the current path, you move backwards on the same path to find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path will be selected.
- This recursive nature of DFS can be implemented using stacks.

DFS Contd..

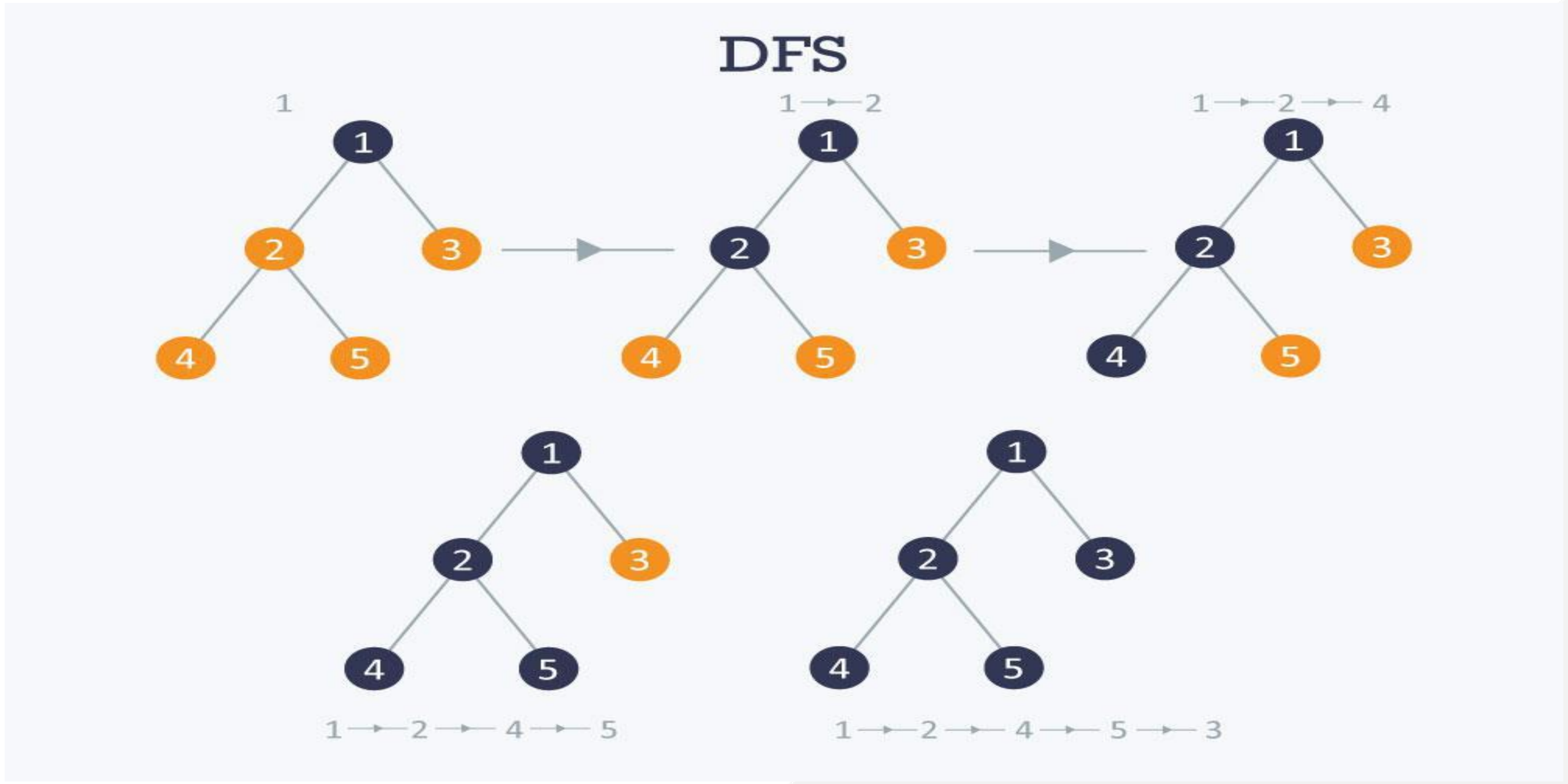
- The basic idea is as follows:
  - Pick a starting node and push all its adjacent nodes into a stack.
  - Pop a node from stack to select the next node to visit and push all its adjacent nodes into a stack.
  - Repeat this process until the stack is empty.
- However, ensure that the nodes that are visited are marked. This will prevent you from visiting the same node more than once. If you do not mark the nodes that are visited and you visit the same node more than once, you may end up in an infinite loop.

Example 1

NOTE:-Follow the dotted arrows to determine the sequence.



Example 2



Key Points

- **Time complexity:** T(n)= 1+ n2+ n3 +.........+ nm =O(nm)

  Where, m= maximum depth of any node

  n= no. of nodes in each level
- **Completeness:** DFS will provide completeness feature when state space is finite
- **Space complexity:** DFS algorithm needs to store only single path from the root node **O (bm)**
- **Optimality:** DFS is not optimal, meaning the number of steps in reaching the solution, or the cost spent in reaching it is high.

Advantages of DFS

- Memory requirement is Linear with respect to Nodes.
- Less time and space complexity rather than BFS.
- Solution can be found out by without much more search.

Disadvantages of DFS

- Not guaranteed that it will give you solution.
- Cut-off depth is smaller so time complexity is more
- Determination of depth until the search has proceeds.

THANK YOU