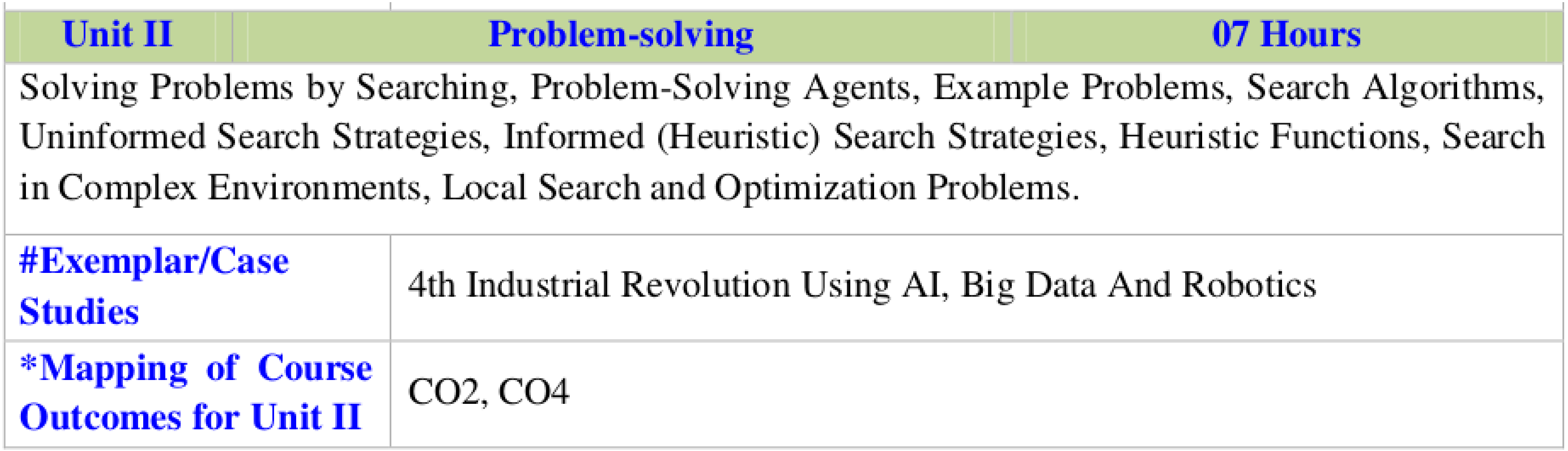
UNIT II

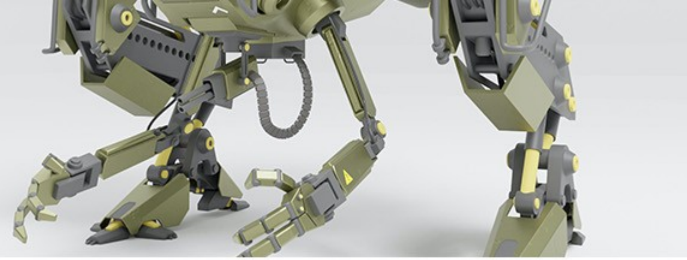
Introduction

Contents



Unit II	Problem-solving	07 Hours
Solving Problems by Searching, Problem-Solving Agents, Example Problems, Search Algorithms, Uninformed Search Strategies, Informed (Heuristic) Search Strategies, Heuristic Functions, Search in Complex Environments, Local Search and Optimization Problems.		
#Exemplar/Case Studies	4th Industrial Revolution Using AI, Big Data And Robotics	
***Mapping of Course Outcomes for Unit II**	CO2, CO4	

Problem Solving



Search: Searchingis 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.

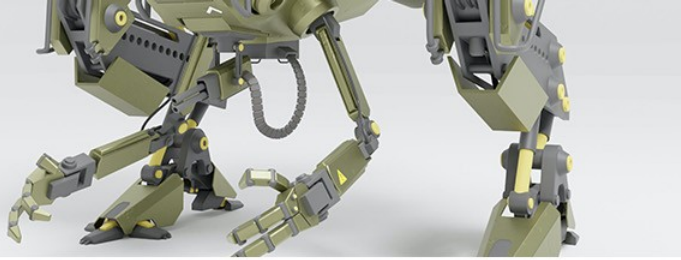
Transition model: A description of what each action do, can be represented as a transition model.

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 among all solutions.

Properties of Search Algorithms:

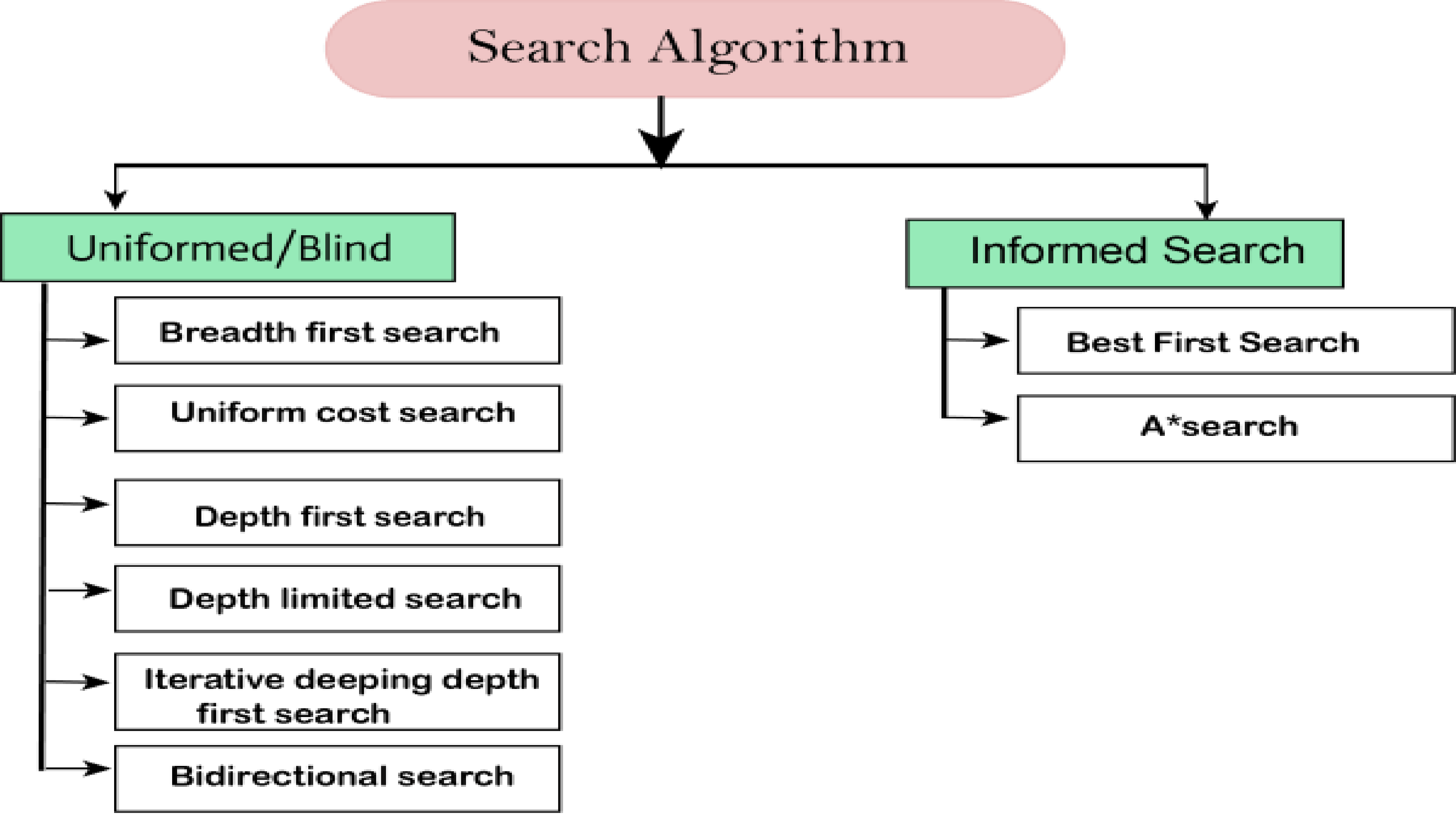


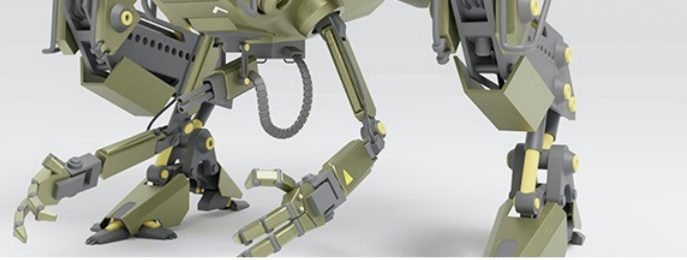
Completeness: A search algorithm is said to be complete if it guarantees to return a solution if at least any solution exists for any random input.

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.


Search Algorithm
Uniformed/Blind
Informed Search
Breadth first search
Uniform cost search
Depth first search
Depth limited search
Iterative deeping depth first search
Bidirectional search
Best First Search
A*search



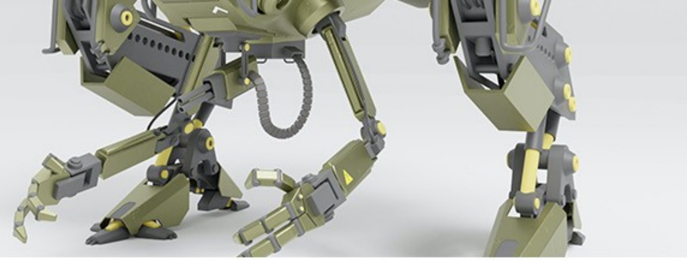
Types of search algorithms

Uninformed/Blind Search:

The uninformed search does not contain any domain knowledge such as closeness, the location of the goal. It operates in a brute-force way as it only includes information about how to traverse the tree and how to identify leaf and goal nodes. Uninformed search applies a way in which search tree is searched without any information about the search space like initial state operators and test for the goal, so it is also called blind search.It examines each node of the tree until it achieves the goal node.

1) Breadth-first search
2) Uniform cost search
3) Depth-first search
4) Iterative deepening depth-first search
5) Bidirectional Search

Types of search algorithms

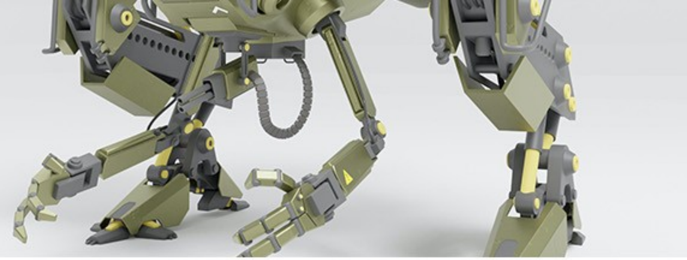


Informed Search

Informed search algorithms use domain knowledge. In an informed search, problem information is available which can guide the search. Informed search strategies can find a solution more efficiently than an uninformed search strategy. Informed search is also called a Heuristic search.

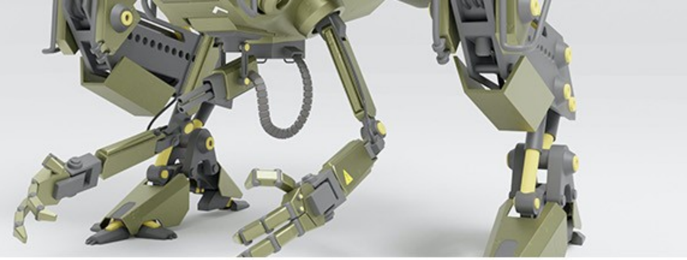
A heuristic is a way which might not always be guaranteed for best solutions but guaranteed to find a good solution in reasonable time. Informed search can solve much complex problem which could not be solved in another way.

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.

Breadth-first Search:



- **Advantages:**

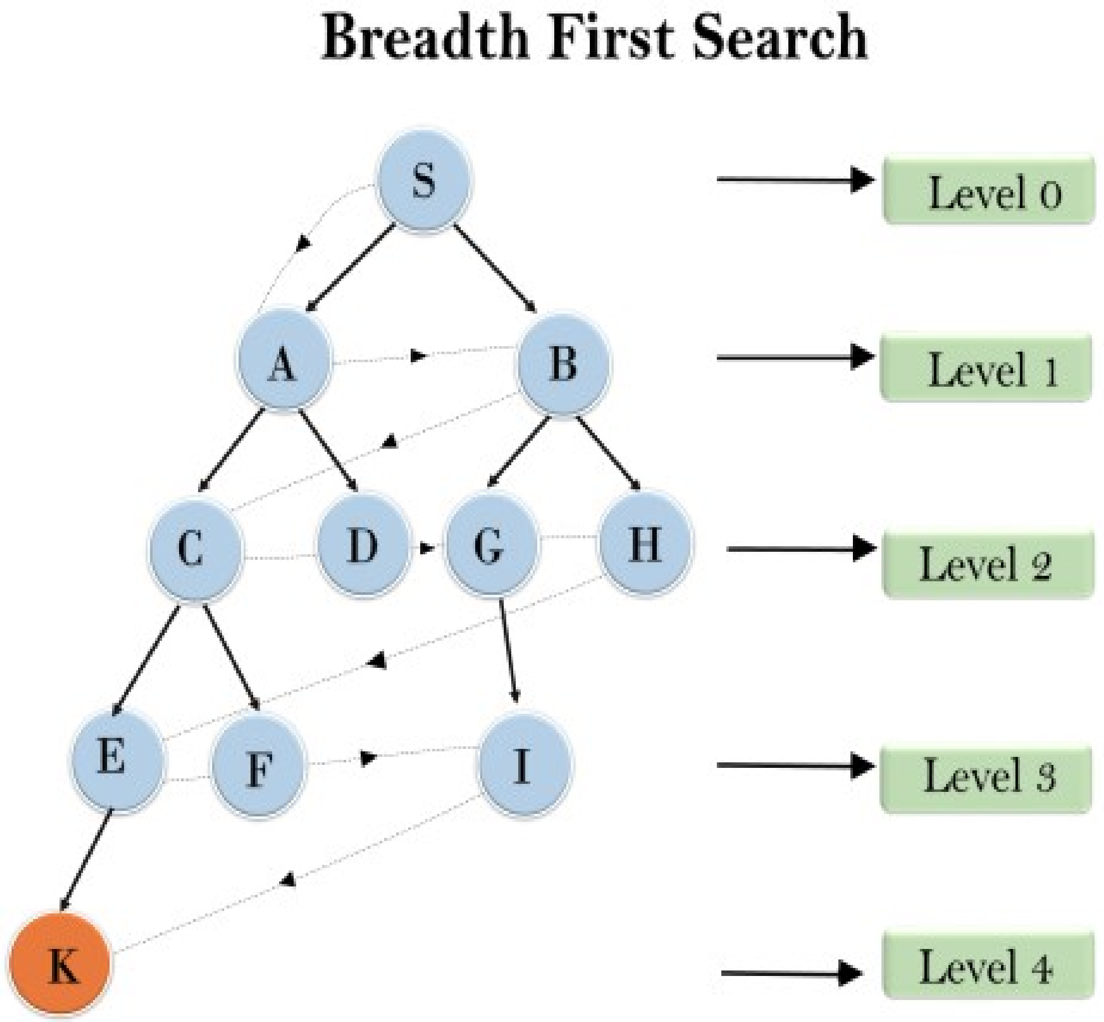
1. BFS will provide a solution if any solution exists.
2. If there are more than one solutions for a given problem, then BFS will provide the minimal solution which requires the least number of steps.

- **Disadvantages:**

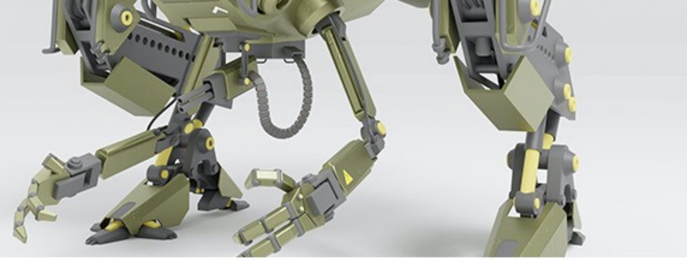
1. It requires lots of memory since each level of the tree must be saved into memory to expand the next level.
2. BFS needs lots of time if the solution is far away from the root node.

Breadth-first Search:

- Example: 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**



Breadth-first Search:

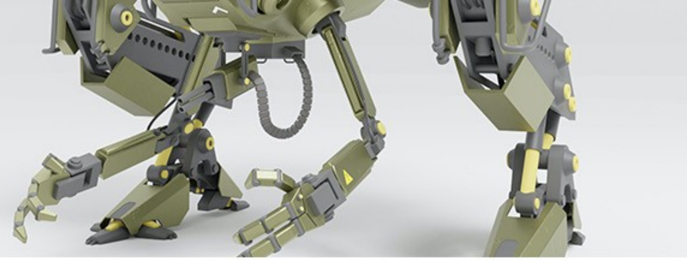


- Time Complexity: Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest Node.
- Where the d= depth of shallowest solution and b is a node at every state.

$$T(b) = 1+b^2+b^3+.......+ b^d= O(b^d)$$

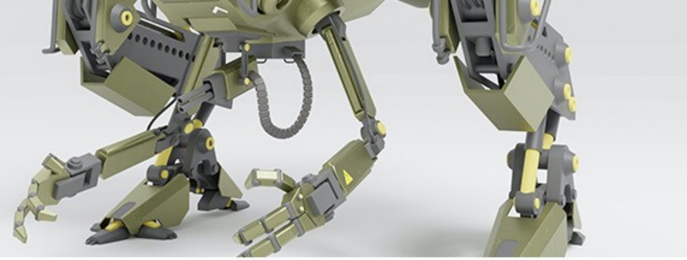
- Space Complexity: Space complexity of BFS algorithm is given by the Memory size of frontier which is $O(b^d)$.
- Completeness: BFS is complete, which means if the shallowest goal node is at some finite depth, then BFS will find a solution.
- Optimality: BFS is optimal if path cost is a non-decreasing function of the depth of the node.

Depth-first Search



- Depth-first search isa 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

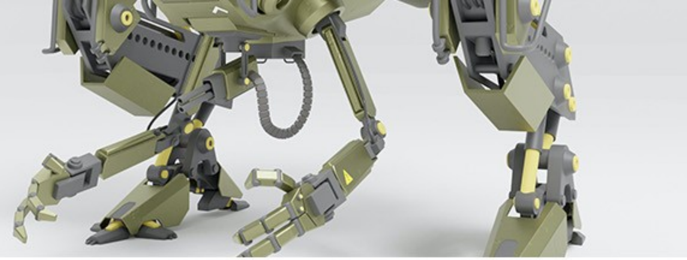


- **Advantage:**

1. 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.
2. It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).

- **Disadvantage:**

1. There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
2. DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

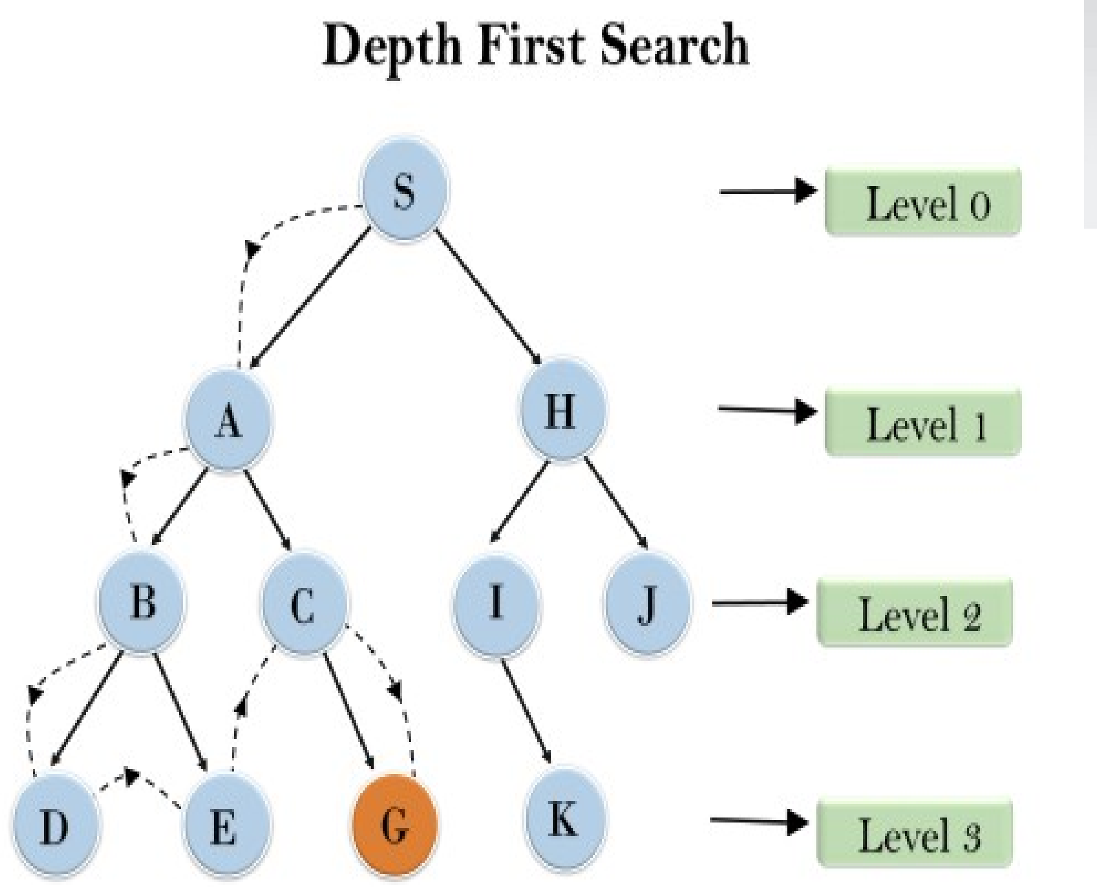


Depth-first Search

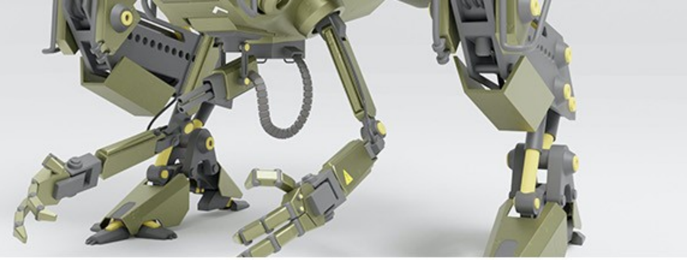
- **Example:**
- In the below search tree, we have shown the flow of depth-first search, and it will follow the order as:
  - **Root node--->Left node ----> right node.**
- It will start searching from root node S, and traverse A, then B, then D and E, after traversing E, it will backtrack the tree as 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.

Depth-first Search

- **Completeness:** DFS search algorithm is complete within finite state space as it will expand every node within a limited search tree.



Depth-first Search



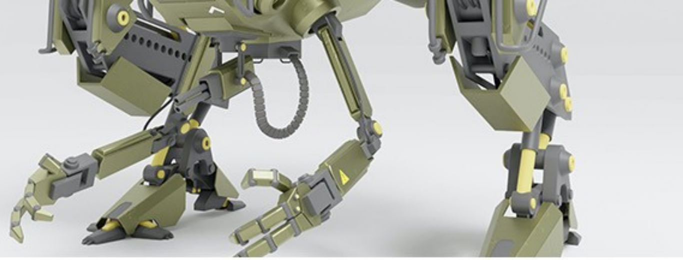
- **Time Complexity**: Time complexity of DFS will be equivalent to the node traversed by the algorithm. It is given by:

  - $$\mathbf{T(n)= 1+ n^2+ n^3 +.........+ n^m=O(n^m)}$$

- Where, m= maximum depth of any node and this can be much larger than d (Shallowest solution depth)

- **Optimal:** DFS search algorithm is non-optimal, as it may generate a large number of steps or high cost to reach to the goal node.

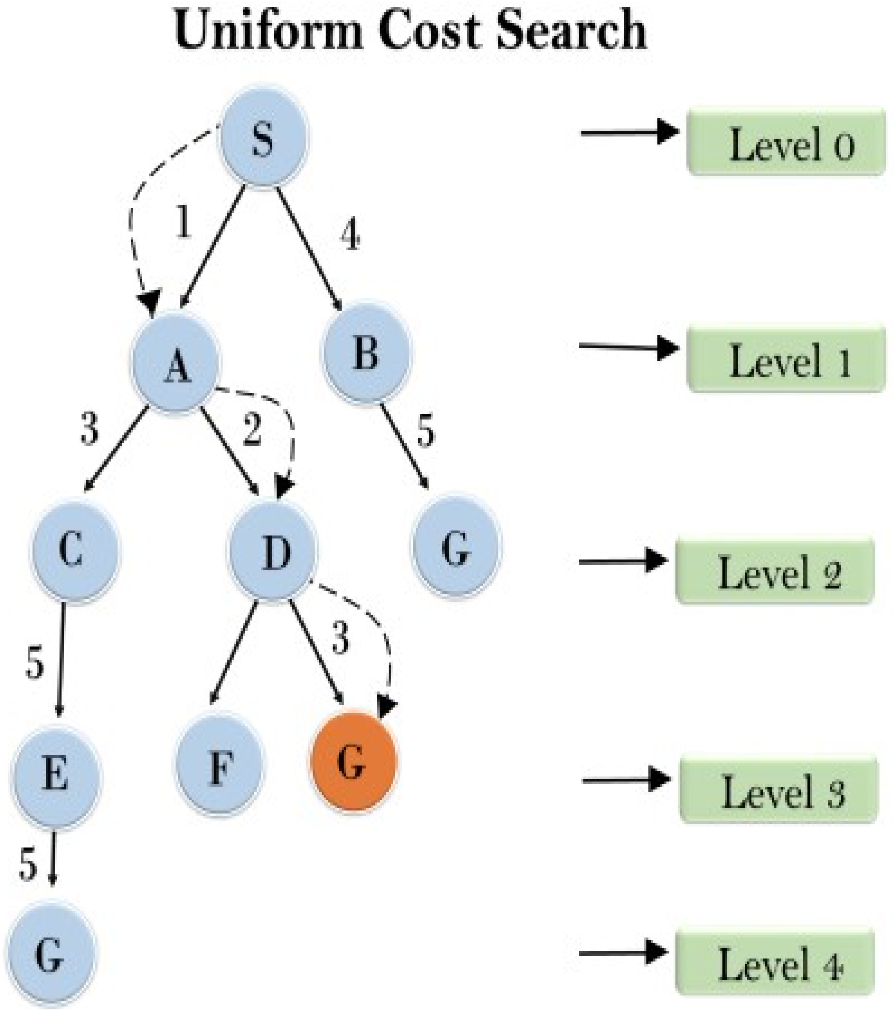
Uniform-cost Search Algorithm:



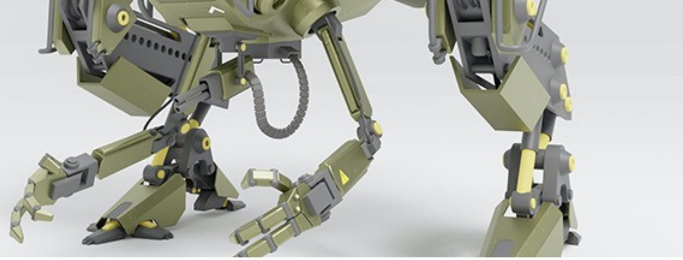
- Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph.
- This algorithm comes into play when a different cost is available for each edge.
- The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost.
- Uniform-cost 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.
- It gives maximum priority to the lowest cumulative cost.
- Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.

Uniform-cost Search:

- **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.



Uniform-cost Search:



- **Completeness:**
  Uniform-cost search is complete, such as if there is a solution, UCS will find it.
- **Time Complexity:**
  Let C* is Cost of the optimal solution, and ε is each step to get closer to the goal node.
  Then the number of steps is = C*/ε+1.

  Here we have taken +1, as we start from state 0 and end to C*/ε.
  The worst-case time complexity of Uniform-cost search isO(b1 + [C*/ε])

- **Optimal:** Uniform-cost search is always optimal as it only selects a path with the lowest path cost.