Unit 3

Artificial Intelligence

Uninformed Search Techniques

Search in AI

Search is a commonly used method in Artificial Intelligence for solving problems.

The search technique explores the possible moves that one can make in a space of ‘states’, called the search space.

Two important states are:

- The start state, which embodies the ‘current state of affairs’.
- The goal state, which embodies the ‘desired state of affairs’.

‘Operators’ allow one to move between states.

In search, one tries to find a path from start state to goal state.

- There can be one or many solutions to a given problem, depending on the scenario, as there can be many ways to solve that problem. Think about how do you approach a problem.

- Lets say you need to do something straight forward like a math multiplication. Clearly there is one correct solution, but we have many algorithms to multiply, depending on the size of the input.

- Now, take a more complicated problem, like playing a game.
- In most of these games, at a given point in time, you have multiple moves that you can make, and you choose the one that gives you ***best possible outcome.***
- In this scenario, there is no one correct solution, ***but there is a best possible solution***, depending on what you want to achieve.
- Also, there are multiple ways to approach the problem, based on what strategy you choose to have for your game play.

Searching and AI

- Searching falls under Artificial Intelligence (AI).
- A major goal of AI is to give computers the ability to think, or in other words, mimic human behavior.
- The problem is, unfortunately, computers don't function in the same way our minds do.
- They require a series of ***well-reasoned out*** steps before finding a solution.

Search algorithm Terminologies

- 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.

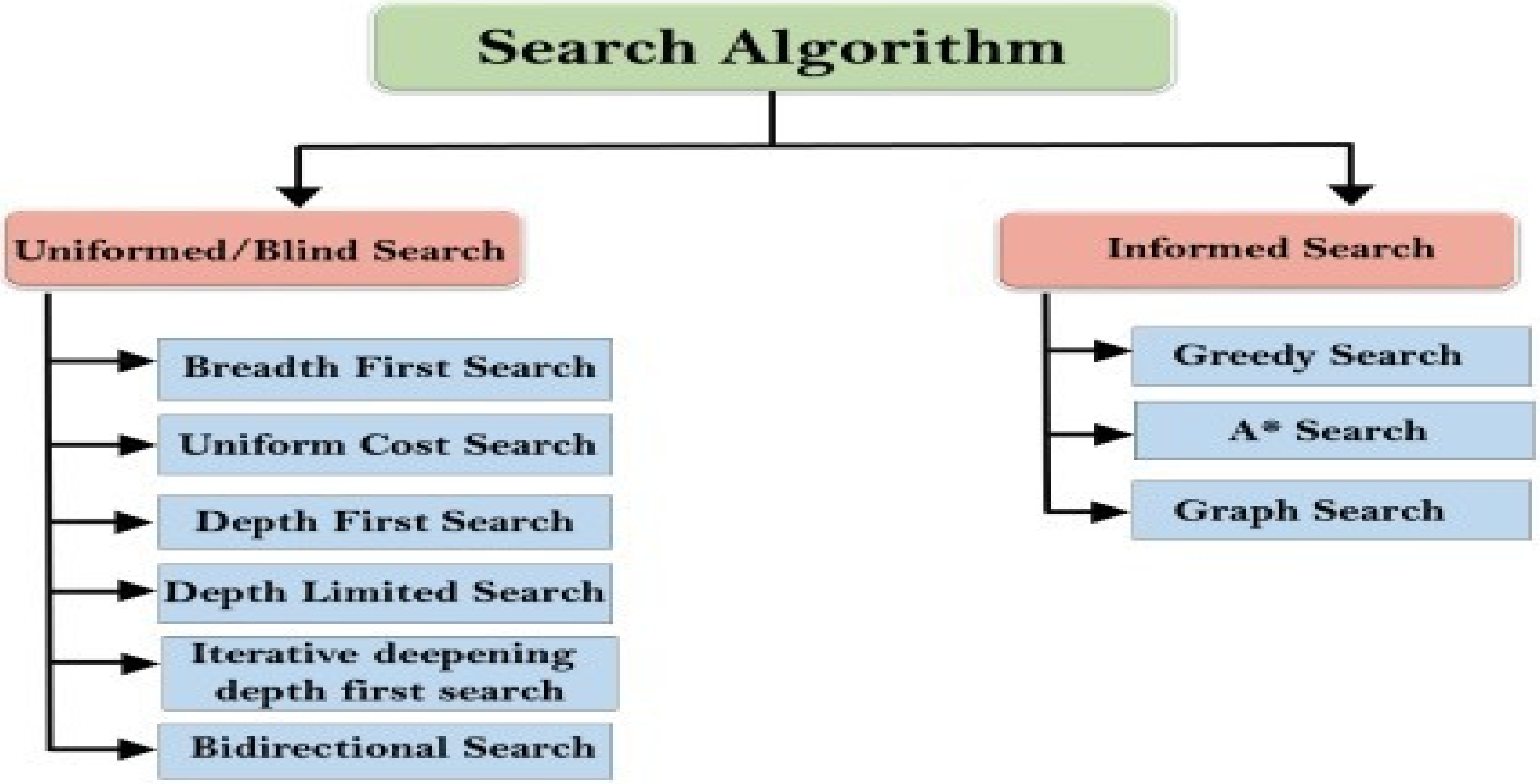
Properties of Search Algorithm

- **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.

Types of Search Algorithm

Based on the search problems we can classify the search algorithms into:

- Uninformed Search (Blind Search)
- Informed search (Heuristic Search)


Search Algorithm
Uniformed/Blind Search
Breadth First Search
Uniform Cost Search
Depth First Search
Depth Limited Search
Iterative deepening depth first search
Bidirectional Search
Informed Search
Greedy Search
A* Search
Graph Search

Uninformed Search/Blind Search

- This type of search ***does not use any domain knowledge.***
- This means that it does not use any information that helps to reach the goal, like closeness or location of the goal.
- The strategies or algorithms, using this form of search, ignore where they are going until they find a goal and report success.
- In this search, ***total search space is looked for the solution.***
- 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.

Informed Search / Heuristic Search

- This type of search ***uses domain knowledge.***
- 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.***
- It generally uses a heuristic function that estimates how close a state is to the goal.
- This heuristic need not be perfect. This function is used to estimate the cost from a state to the closest goal.

Types of Heuristic Search

- **Generate and test**
- **Best first search (Greedy search)**
- **Hill-Climbing search**
- Problem Reduction
- Constraint Satisfaction
- Means end analysis
- Min-max search

Informed Search vs. Uninformed Search

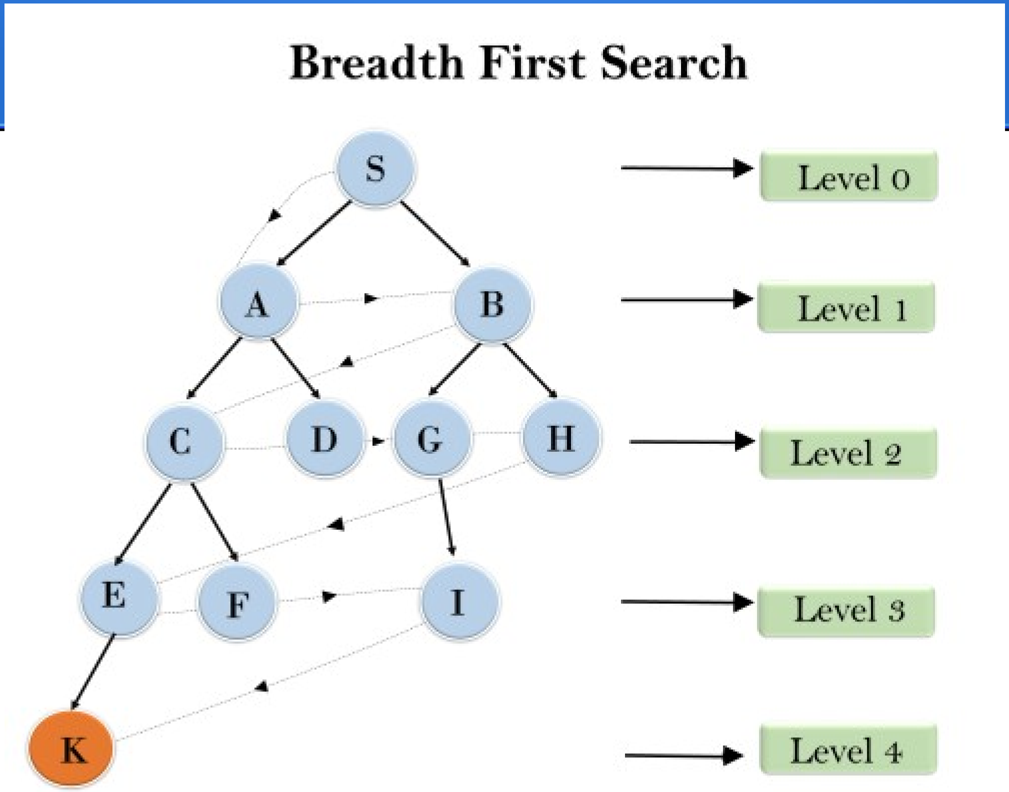
Informed Search	Uninformed Search
It uses knowledge for the searching process.	It doesn't use knowledge for searching process.
It finds solution more quickly.	It finds solution slow as compared to informed search.
It is highly efficient.	It is mandatory efficient.
Cost is low.	Cost is high.
It consumes less time.	It consumes moderate time.
It provides the direction regarding the solution.	No suggestion is given regarding the solution in it.
It is less lengthy while implementation.	It is more lengthy while implementation.
Greedy Search, A* Search, Graph Search	Depth First Search, Breadth First Search

Types of Uninformed Search

- BFS (Breadth First Search): It expands the shallowest node (node having lowest depth) first.
- DFS (Depth First Search): It expands deepest node first.
- DLS (Depth Limited Search): It is DFS with a limit on depth.
- Bidirectional Search

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.



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

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+b2+b3+.......+ bd= O (bd)

Space Complexity: Space complexity of BFS algorithm is given by the Memory size of frontier which is O(bd).

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.

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.

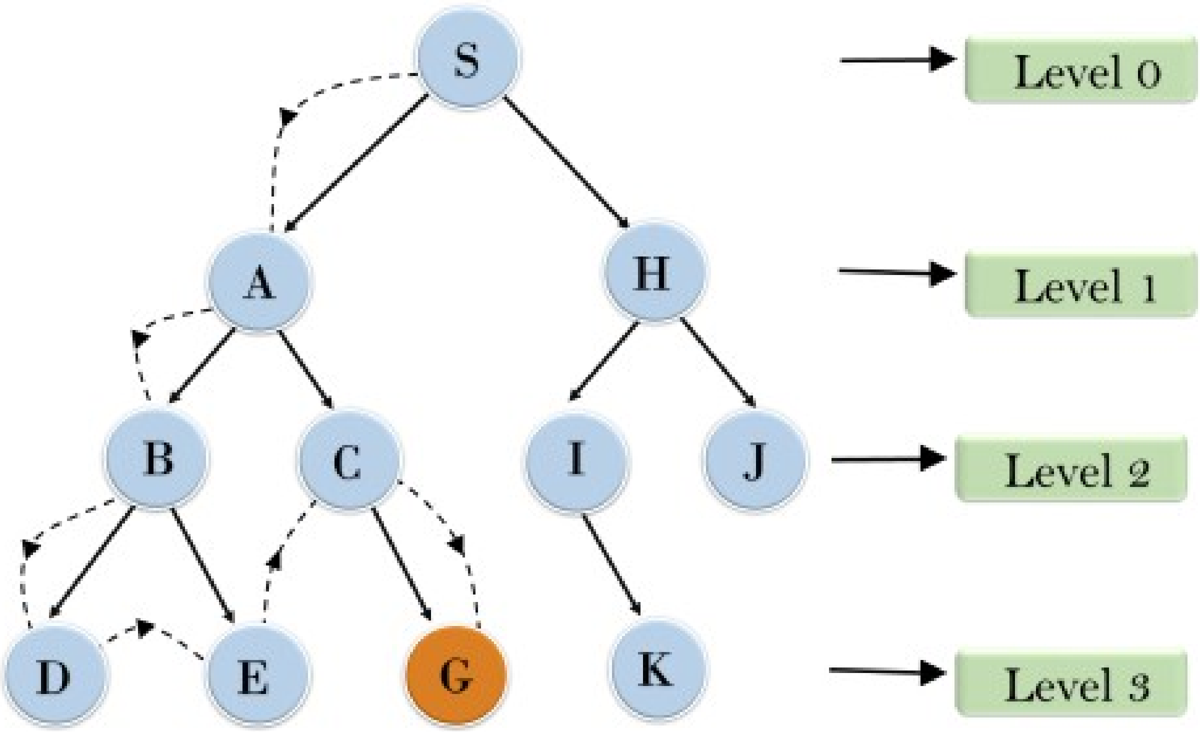
Disadvantages:

- 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.

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



When Goal state is G :

S---> A--->B---->D--->E---->C--->G

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

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

  $$T(n)= 1+ n2+ n3 +.........+ nm=O(nm)$$

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

- **Space Complexity**: DFS algorithm needs to store only single path from the root node, hence space complexity of DFS is equivalent to the size of the fringe set, which is O(bm).

- **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.

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.

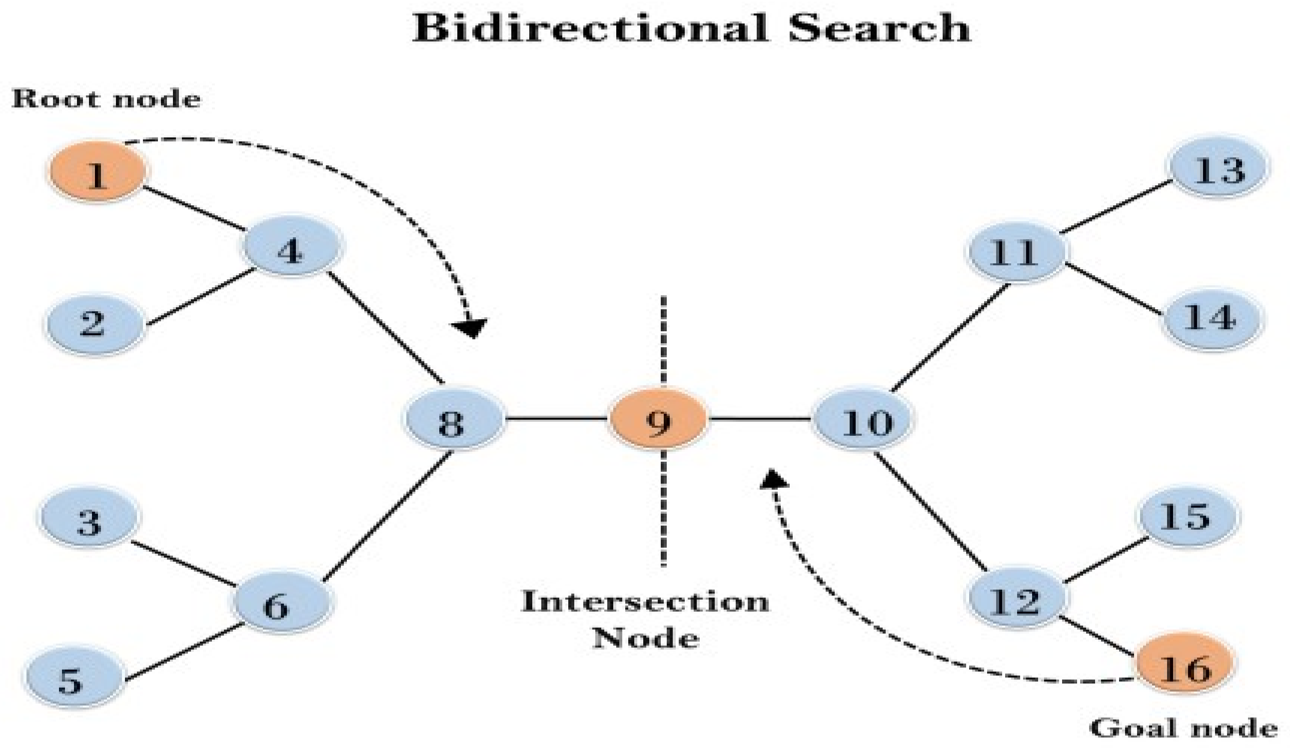
- It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).

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.

Bidirectional Search

- Bidirectional search algorithm runs two simultaneous searches, one form initial state called as forward-search and other from goal node called as backward-search, to find the goal node.
- Bidirectional search replaces one single search graph with two small subgraphs in which one starts the search from an initial vertex and other starts from goal vertex.
- The search stops when these two graphs intersect each other.



This algorithm divides one graph/tree into two sub-graphs. It starts traversing from node 1 in the forward direction and starts from goal node 16 in the backward direction.

The algorithm terminates at node 9 where two searches meet.

- **Completeness**: Bidirectional Search is complete if we use BFS in both searches.

- **Time Complexity**: Time complexity of bidirectional search using BFS is O(bd).

- **Space Complexity**: Space complexity of bidirectional search is O(bd).

- **Optimal**: Bidirectional search is Optimal.

Advantages:

- Bidirectional search is fast.
- Bidirectional search requires less memory

Disadvantages:

- Implementation of the bidirectional search tree is difficult.
- In bidirectional search, one should know the goal state in advance.

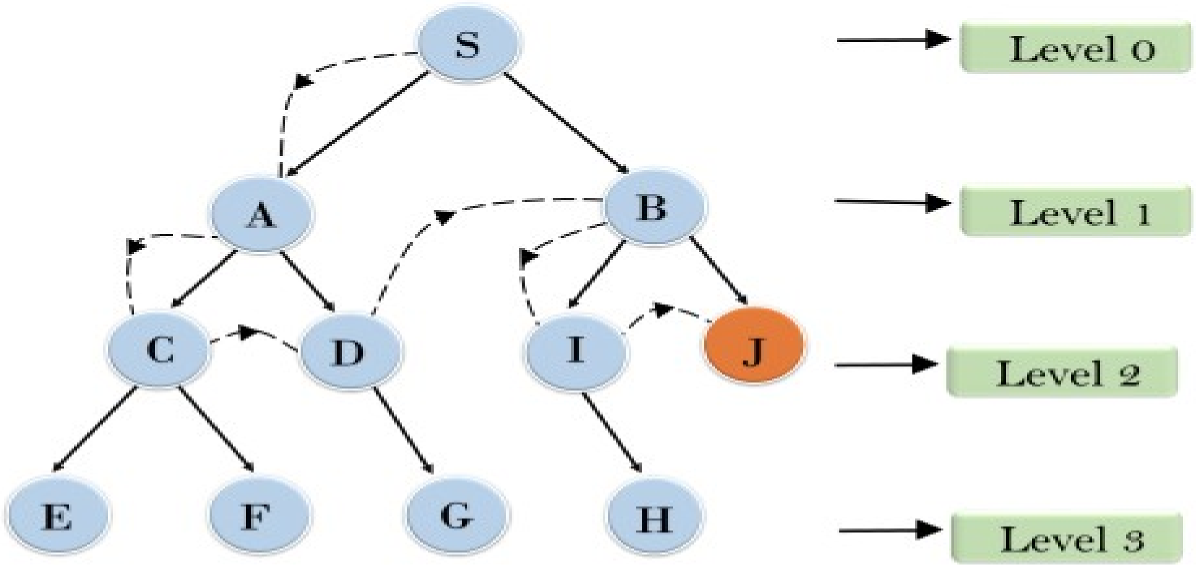
Depth Limited Search

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.

Depth Limited Search



- **Completeness**: DLS search algorithm is complete if the solution is above the depth-limit.
- **Time Complexity**: Time complexity of DLS algorithm is $O(b^{\ell})$.
- **Space Complexity**: Space complexity of DLS algorithm is $O(b \times \ell)$.

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.