**MODULE 3**

**Solving Problems by Searching**

**{3.1}**

1. **State space representation**

[more](https://studyresearch.in/2018/03/11/state-space-representation-consist-of-defining-an-initial-state/#:~:text=State%20Space%20Representation%20consist%20of,formed%20set%20of%20possible%20states.)

1. State space search is a process in which successive states of an instance are considered with the goal of finding the desired goal state and property.
2. Before an Al problem can be solved it must be represented as a state space. The state space is then searched to find a solution to the problem.
3. A state space essentially consists of a set of nodes representing each state of the problem, arcs between nodes representing the legal moves from one state to another, an initial state and a goal state.
4. Each state space takes the form of a tree or a graph.
5. Factors that determine which search algorithm or technique will be used include the type of the problem and how the problem can be represented.
6. Search techniques that will be examined in the course include

* Depth First Search
* Depth First Search with Iterative Deepening
* Breadth First Search
* Best First Search
* Hill Climbing
* Branch and Bound Techniques
* A\* Algorithm

1. **Problem as a state space search**

State space search is a process used in the field of computer science, including artificial intelligence (AI).

(ii) In which successive 'configurations' or 'states' of an instance are considered, with the intention of finding a goal state with the desired property.

(iii) The set of states forms a graph where two states are connected if one state can be transformed to another state by the given operation and states.

(iv) State space search differs from traditional computer science search methods because the state space is implicit.

(v) The typical state space graph is much too large to generate.

(vi) Instead nodes are generated as they are explored, and

then discarded.

(vii) A solution may consist of the goal state itself, or of a path from some initial state to the goal state.

1. **Problem formulation**

It is one of the core steps of problem-solving which decides what action should be taken to achieve the formulated goal.

In AI this core part is dependent upon a software agent which consists of the components to formulate the associated problem.

Every algorithm demands problems in a specific form.

Before problem formulation it is very important to know components of the problem.

Components of problem are

* Problem statement
* Problem solution
* Solution space
* Operators

Examples of problem formulation

Mouse path problem

Water jug problem

1. **Well-defined problems :**

Problem - solving consists of using generic or ad-hoc methods in an orderly manner.

Problems can be classified into two different types :

ill defined and well-defined.

1. Ill defined problems are those that do not have clear goals, solution paths, or expected Solutions.

2. Well - defined problems have specific goals, clearly defined solution paths, and clear expected solutions.

These problems also allow for more initial planning than ill -defined problems.

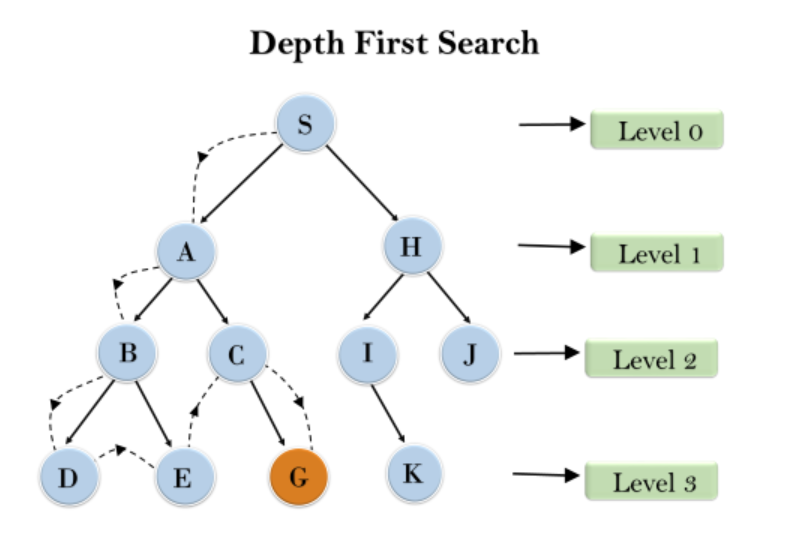
**{3.3}**

**Uninformed Search:**

Uninformed search is a class of general-purpose search algorithms which operates in brute force-way. Uninformed search algorithms do not have additional information about state or search space other than how to traverse the tree, so it is also called blind search.

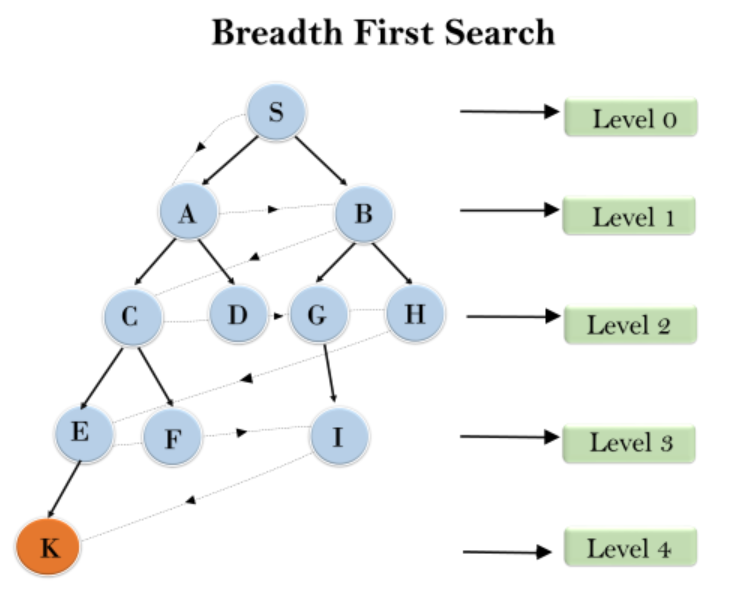
**Depth First Search:**

1. Depth-first search is a recursive algorithm for traversing a tree or graph data structure.
2. 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.
3. DFS uses a stack data structure for its implementation.
4. The process of the DFS algorithm is similar to the BFS algorithm but takes less time to reach the goal.



**Breadth First Search**

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

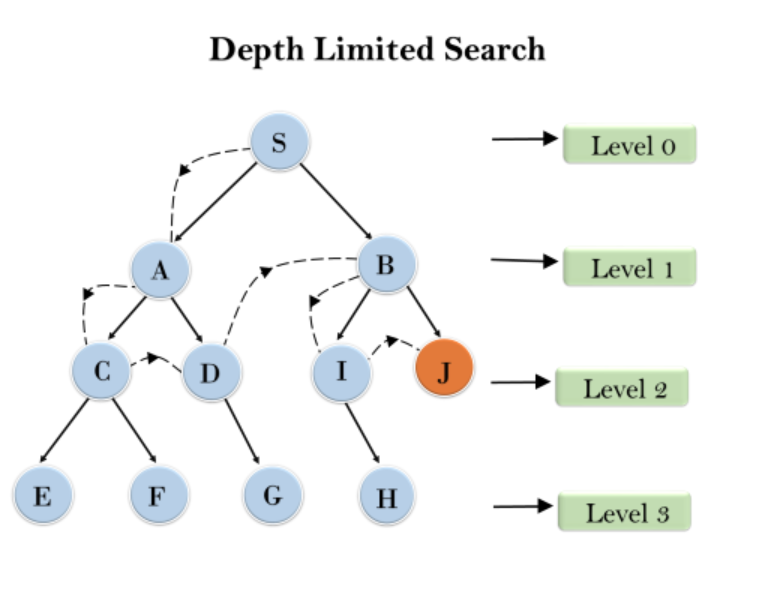


**Depth Limited Search**

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 limit will be treated as it has no successor nodes further.

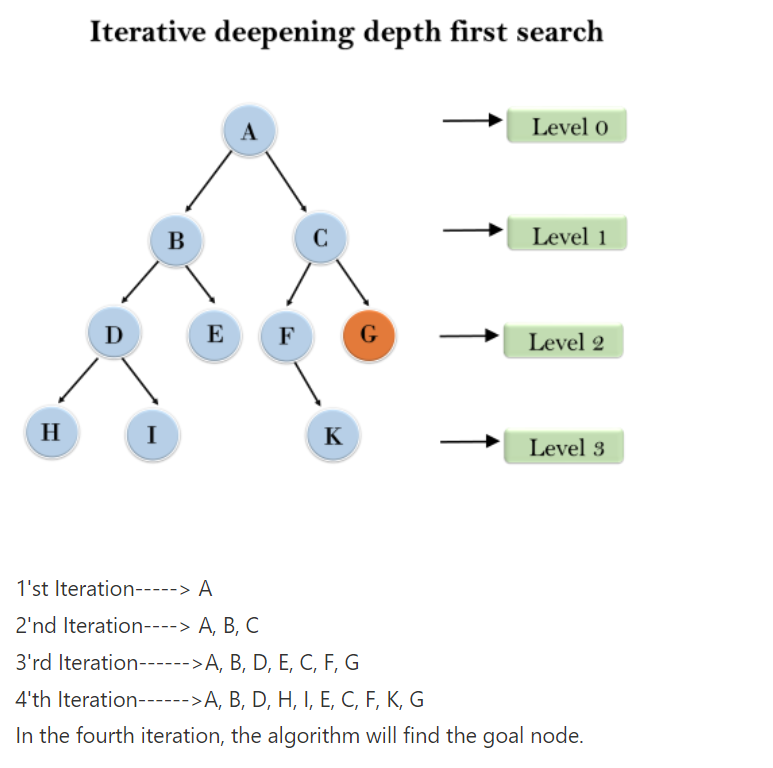
Depth-limited search can be terminated with two Conditions of failure:

* Standard failure value: It indicates that the problem does not have any solution.
* Cutoff failure value: It defines no solution for the problem within a given depth limit



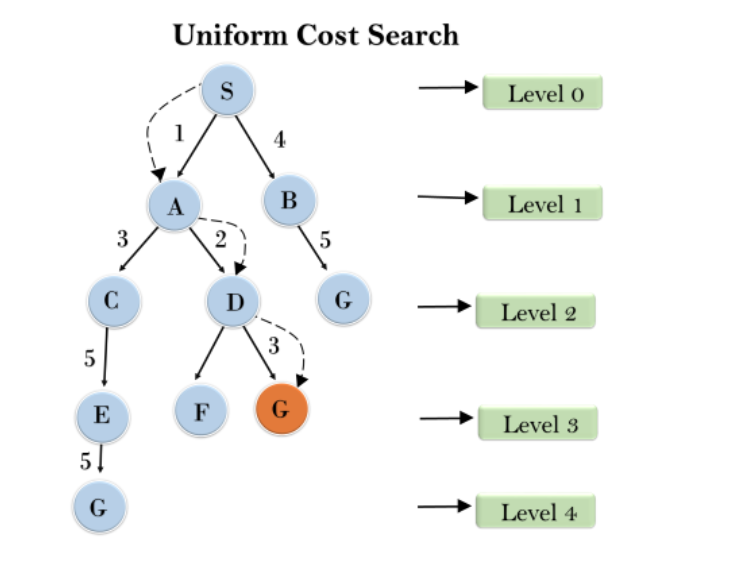
**Iterative Deepening Search**

1. The iterative deepening algorithm is a combination of DFS and BFS algorithms.
2. This search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found.
3. This Search algorithm combines the benefits of Breadth-first search's fast search and depth-first search's memory efficiency.
4. 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



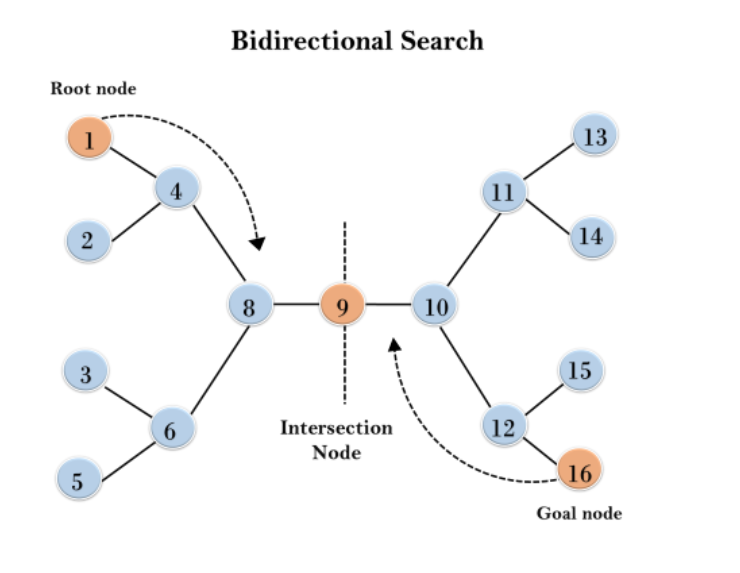
**Uniform Cost Search**

1. Uniform-cost search is a searching algorithm used for going through a weighted tree or graph.
2. The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost.
3. It can be used to solve any graph/tree where the optimal cost is in demand
4. A uniform-cost search algorithm is implemented by the priority queue.



**Bidirectional Search**

1. Bidirectional search algorithm runs two simultaneous searches, one from initial state called as forward-search and other from goal node called as backward-search, to find the goal node.
2. 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.
3. The search stops when these two graphs intersect each other.
4. Bidirectional search can use search techniques such as BFS, DFS, DLS, etc.



**{3.4}**

**Informed Search:**

1. an informed search algorithm contains an array of knowledge such as how far we are from the goal, path cost, how to reach the goal node, etc.
2. This knowledge helps agents to explore less to the search space and find more efficiently the goal node.
3. The informed search algorithm is more useful for large search spaces.
4. Informed search algorithm uses the idea of heuristic, so it is also called Heuristic search

**Heuristic Function**

1. Heuristic is a function which is used in Informed Search, and it finds the most promising path.
2. It takes the current state of the agent as its input and produces the estimation of how close the agent is from the goal.
3. The heuristic method, however, might not always give the best solution, but it guaranteed to find a good solution in reasonable time.
4. Heuristic function is represented by h(n), and it calculates the cost of an optimal path between the pair of states. The value of the heuristic function is always positive.

**Admissible Heuristic**

Admissibility of the heuristic function is given as: h(n) <= h\*(n)

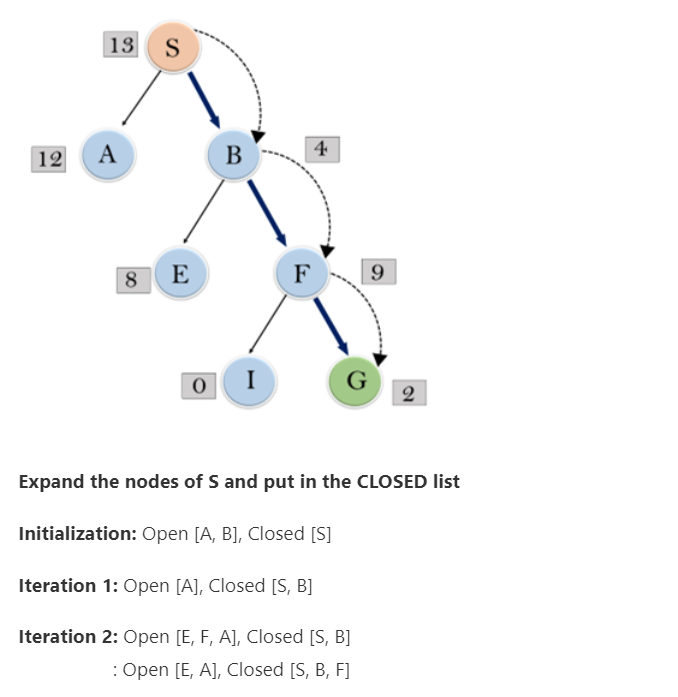
Here h(n) is heuristic cost, and h\*(n) is the estimated cost. Hence heuristic cost should be less than or equal to the estimated cost.

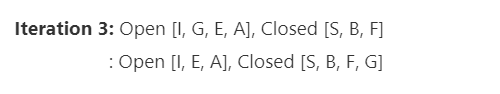
**Informed Search Techniques**

**Informed Search** algorithms have information on the goal state which helps in more efficient searching. This information is obtained by a function that estimates how close a state is to the goal state

**Greedy Best First Search**

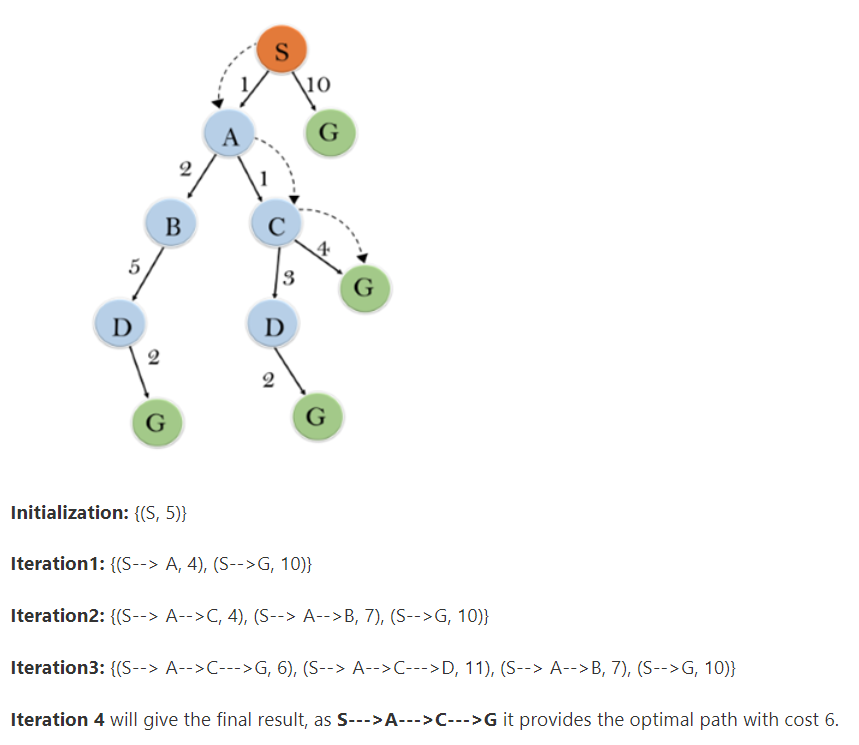
1. Greedy best-first search algorithm always selects the path which appears best at that moment.
2. It is the combination of depth-first search and breadth-first search algorithms.
3. It uses the heuristic function and search.
4. With the help of best-first search, at each step, we can choose the most promising node





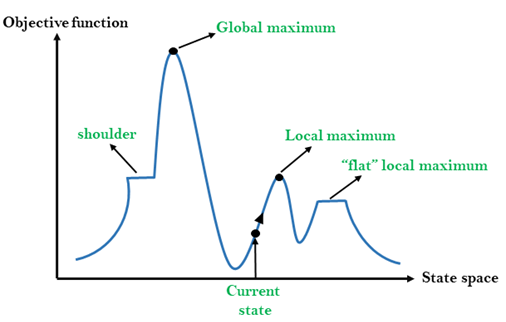
**A\* Search**

1. A\* search is the most commonly known form of best-first search.
2. It uses heuristic function h(n), and cost to reach the node n from the start state.
3. A\* search algorithm finds the shortest path through the search space.
4. This search algorithm expands the search tree and provides optimal results faster.
5. A\* algorithm returns the path which occurred first, and it does not search for all remaining paths.



**Local Search:**

1. **Hill Climbing Search:**
2. Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing value to find the best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.
3. Hill climbing algorithm is a technique which is used for optimizing mathematical problems. One of the widely discussed examples of Hill climbing algorithms is the Traveling-salesman Problem in which we need to minimize the distance traveled by the salesman.
4. It is also called greedy local search, it only looks to its neighboring state not beyond that.
5. A node in a hill climbing algorithm has two components: state and value.
6. In this algorithm we don’t need to keep a search tree or graph in the memory as it only keeps the single current state.

* **Features of Hill Climbing:**
  + **Generate and Test variant:** Hill Climbing is the variant of Generate and Test method. The Generate and Test method produce feedback which helps to decide which direction to move in the search space.
  + **Greedy approach:** Hill-climbing algorithm search moves in the direction which optimizes the cost.
  + **No backtracking:** It does not backtrack the search space, as it does not remember the previous states.
* **State Space landscape:**
* ****

**Simulated Annealing Search**

The simulated annealing optimization algorithm can be thought of as a modified version of stochastic hill climbing.

Stochastic hill climbing maintains a single candidate solution and takes steps of a random but constrained size from the candidate in the search space. If the new point is better than the current point, then the current point is replaced with the new point. This process continues for a fixed number of iterations.

Simulated annealing executes the search in the same way. The main difference is that new points that are not as good as the current point (worse points) are accepted sometimes.

A worse point is accepted probabilistically where the likelihood of accepting a solution worse than the current solution is a function of the temperature of the search and how much worse the solution is than the current solution

A hill-climbing algorithm that never makes “downhill” moves toward states with lower value

(or higher cost) is guaranteed to be incomplete, because it can get stuck on a local maximum.

In contrast, a purely random walk—that is, moving to a successor chosen uniformly at random from the

The set of successors—is complete but extremely inefficient.

Therefore, it seems reasonable to combine hill climbing with a random walk in some way that yields both

efficiency and completeness.

Idea: escape local maxima by allowing some “bad” moves but gradually decrease their size

and frequency.

The simulated annealing algorithm, a version of stochastic hill climbing where some downhill moves are allowed.

Annealing: the process of gradually cooling metal to allow it to form stronger crystalline structures

Simulated annealing algorithm: gradually “cool” search algorithm from Random Walk to First-

Choice Hill Climbing

[**Optimization: Genetic Algorithm**](https://www.geeksforgeeks.org/genetic-algorithms/)

**{3.5}**

**GAME PLAYING**

**MINIMAX Algorithm**

Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that the opponent is also playing optimally.

The Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various two-players games. This Algorithm computes the minimax decision for the current state.

In this algorithm two players play the game, one is called MAX and other is called MIN.

The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.

The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

**Adversarial Search Techniques-**

There might be some situations where more than one agent is searching for the solution in the same search space, and this situation usually occurs in game playing.

The environment with more than one agent is termed a multi-agent environment, in which each agent is an opponent of another agent and playing against each other.

Searches in which two or more players with conflicting goals are trying to explore the same search space for the solution, are called adversarial searches.

**Alpha-Beta Pruning-**

Alpha-beta pruning is a modified version of the minimax algorithm.

As we have seen in the minimax search algorithm, the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half

Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called **pruning**.

This involves two threshold parameters Alpha and Beta for future expansion, so it is called **alpha-beta pruning**.

Alpha-beta pruning can be applied at any depth of a tree

The two-parameter can be defined as:

1. **Alpha:** The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is **-∞**.
2. **Beta:** The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is **+∞**
3. The Max player will only update the value of alpha.
4. The Min player will only update the value of beta.