

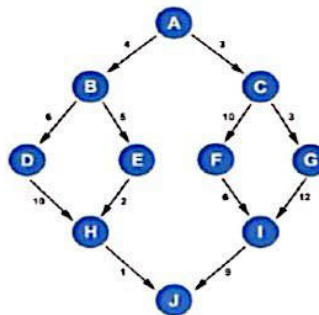
The course for terminal includes: Agents, searches (Adversarial Search, GBFS, Hill Climbing, GA, Heuristically informed Methods, Uniform Cost Search, A\*), First Order Logic and Knowledge-Based Agents, Uncertainty, Planning and Constraint Satisfaction Problems.

\*\*\*\*\*

**Q1. Express the following sentences using first-order logic (FOL). Alongside universal and/or existential quantifiers, use appropriate symbols for objects and functions.**

| Sentences                                              | FOL                                                                                                                |
|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| Frank bought a dvd                                     | Bought(Frank, dvd)                                                                                                 |
| Someone bought everything                              | $\exists x \forall y. \text{bought}(x, y)$                                                                         |
| There is a student who is loved by every other student | $\exists x. (\text{Student}(x) \wedge \forall y. (\text{Student}(y) \wedge \neg(x=y) \supset \text{Loves}(x, y)))$ |
| Every student takes at least one course.               | $\forall x. (\text{Student}(x) \supset \exists y. (\text{Course}(y) \wedge \text{Takes}(x, y)))$                   |
| Not every dog is white                                 | $\neg \forall x. \text{Dog}(x) \supset \text{White}(x)$                                                            |
| Bill has at most one sister.                           | $\forall x \forall y. (\text{SisterOf}(x, \text{Bill}) \wedge \text{SisterOf}(y, \text{Bill})) \supset x = y$      |

**Q2: For each of the following search strategies, give the path that would be returned from “a to j”, or write none if no path will be returned. If there are any ties, assume alphabetical tiebreaking (i.e., nodes for states earlier in the alphabet are expanded first in the case of ties).**



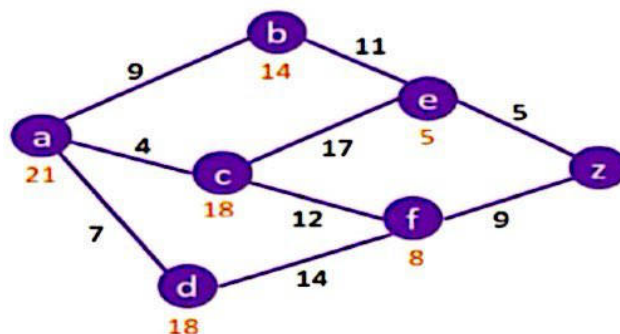
| Search Type                                 | List of States |
|---------------------------------------------|----------------|
| Greedy Search algorithm (Highest cost)      | A, B, D, H, J  |
| Uniform Cost Search (Path with lowest cost) | A, C, G, I, J  |
| A-Star Search ( Path with highest cost)     | A, B, D, H, J  |

**Question 3. Express the following sentences using first-order logic (FOL). Alongside universal and/or existential quantifiers, use appropriate symbols for objects and functions.**

| Sentences                               | FOL                                                                                              |
|-----------------------------------------|--------------------------------------------------------------------------------------------------|
| No students love Bill                   | $\neg \exists x. (\text{Student}(x) \wedge \text{Loves}(x, \text{Bill}))$                        |
| Every student takes at least one course | $\forall x. (\text{Student}(x) \supset \exists y. (\text{Course}(y) \wedge \text{Takes}(x, y)))$ |

|                              |                                                                                          |
|------------------------------|------------------------------------------------------------------------------------------|
| Every Dog has a Tail         | $\forall x. \text{Dog}(x) \supset \exists y (\text{PartOf}(x, y) \wedge \text{Tail}(y))$ |
| Everyone bought something    | $\forall x \exists y. \text{bought}(x, y)$                                               |
| There exists a smart student | $\exists x. (\text{Student}(x) \wedge \text{Smart}(x))$                                  |
| Bill is a student            | $\text{Student}(\text{Bill})$                                                            |

**Q4:** For each of the following search strategies, give the path that would be returned from "a to z", or write none if no path will be returned. If there are any ties, assume alphabetical tiebreaking (i.e., nodes for states earlier in the alphabet are expanded first in the case of ties).



| Search Type                                 | List of States |
|---------------------------------------------|----------------|
| Greedy Search algorithm (Highest cost)      | None           |
| Uniform Cost Search (Path with lowest cost) | A, C, F, Z     |
| A-Star Search ( Path with highest cost)     | None           |

**Iteration Techniques working:**

|                         |                                                                                                                                                                                                                                                                                           |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Greedy Search Algorithm | Iterate by seeing at the heuristic cost.<br>Its Follow DFS algorithm.                                                                                                                                                                                                                     |
| Uniform Cost Search     | Iterate by seeing the Path Cost (Actual Cost).<br>Cost sum after each iteration I -e Cumulative cost<br>By comparing cumulative cost at point, we find optimal path                                                                                                                       |
| A Star                  | Actual Cost + Heuristic Cost                      So,<br>$f(n) = g(n) + h(n)$ where<br>$g(n)$ = cost of traversing from one node to another. This will vary from node to node<br>$h(n)$ = heuristic approximation of the node's value. This is not a real value but an approximation cost |

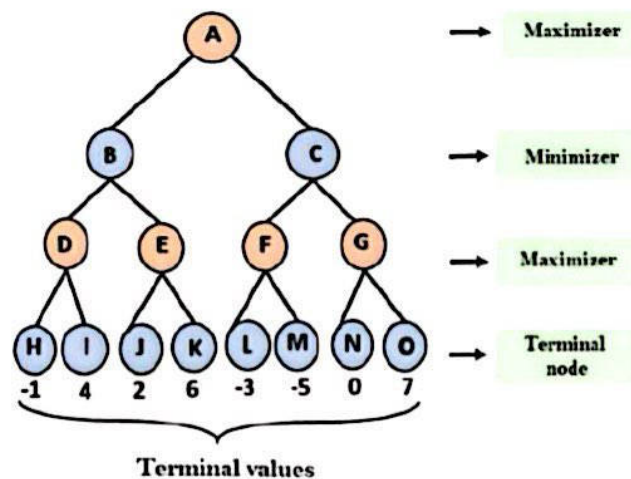
### Mini-Max Algorithm in Artificial Intelligence

- i. 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 opponent is also playing optimally.
- ii. Mini-Max algorithm uses recursion to search through the game-tree.
- iii. Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.
- iv. In this algorithm two players play the game, one is called MAX and other is called MIN.
- v. Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
- vi. Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
- vii. The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
- viii. The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

### Working of Min-Max Algorithm:

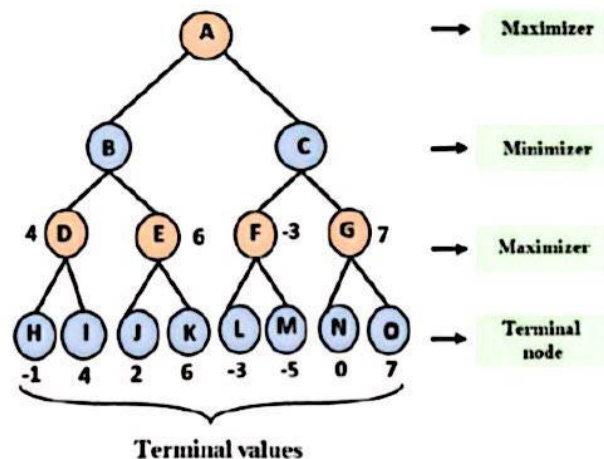
- i. The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.
- ii. In this example, there are two players one is called Maximizer and other is called Minimizer.
- iii. Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
- iv. This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.
- v. At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs. Following are the main steps involved in solving the two-player game tree:

**Step-1:** In the first step, the algorithm generates the entire game-tree and apply the utility function to get the utility values for the terminal states. In the below tree diagram, let's take A is the initial state of the tree. Suppose maximizer takes first turn which has worst-case initial value = - infinity, and minimizer will take next turn which has worst-case initial value = +infinity.



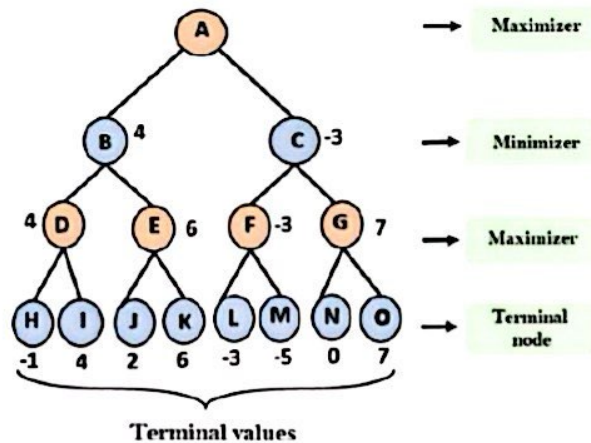
**Step 2:** Now, first we find the utilities value for the Maximizer, its initial value is  $-\infty$ , so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.

- i. For node D  $\max(-1, -\infty) \Rightarrow \max(-1, 4) = 4$
- ii. For Node E  $\max(2, -\infty) \Rightarrow \max(2, 6) = 6$
- iii. For Node F  $\max(-3, -\infty) \Rightarrow \max(-3, -5) = -3$
- iv. For node G  $\max(0, -\infty) = \max(0, 7) = 7$



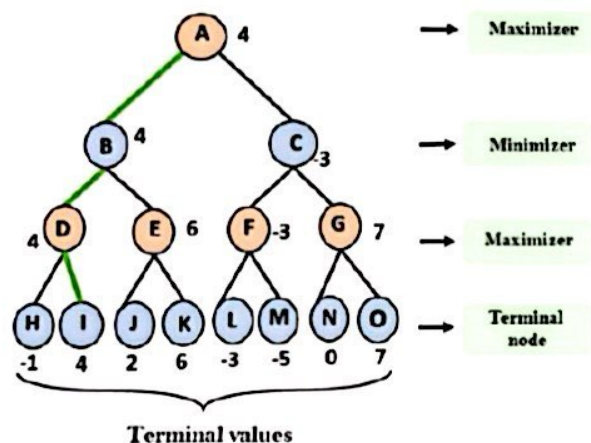
**Step 3:** In the next step, it's a turn for minimizer, so it will compare all nodes value with  $+\infty$ , and will find the 3<sup>rd</sup> layer node values.

- i. For node B  $\min(4, 6) = 4$
- ii. For node C  $\min(-3, 7) = -3$



**Step 4:** Now it's a turn for Maximizer, and it will again choose the maximum of all nodes value and find the maximum value for the root node. In this game tree, there are only 4 layers, hence we reach immediately to the root node, but in real games, there will be more than 4 layers.

- i. For node A  $\max(4, -3) = 4$



That was the complete workflow of the minimax two player game.

Properties of Mini-Max algorithm:

- i. **Complete-** Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
- ii. **Optimal-** Min-Max algorithm is optimal if both opponents are playing optimally.
- iii. **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is  $O(b^m)$ , where  $b$  is branching factor of the game-tree, and  $m$  is the maximum depth of the tree.
- iv. **Space Complexity-** Space complexity of Mini-max algorithm is also similar to DFS which is  $O(bm)$ .

### Alpha-Beta Pruning

- i. Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
- ii. As we have seen in the minimax search algorithm that 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 parameter Alpha and beta for future expansion, so it is called **alpha-beta pruning**. It is also called as **Alpha-Beta Algorithm**.
- iii. Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.
- iv. 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  $-\infty$ .
  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  $+\infty$ .
- v. The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm

### Condition for Alpha-beta pruning:

The main condition which required for alpha-beta pruning is:

1.  $\alpha \geq \beta$

Key points about alpha-beta pruning:

- i. The Max player will only update the value of alpha.
- ii. The Min player will only update the value of beta.
- iii. While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
- iv. We will only pass the alpha, beta values to the child nodes.

### Working of Alpha-Beta Pruning:

Let's take an example of two-player search tree to understand the working of Alpha-beta pruning

**Step 1:** At the first step the, Max player will start first move from node A where  $\alpha = -\infty$  and  $\beta = +\infty$ , these value of alpha and beta passed down to node B where again  $\alpha = -\infty$  and  $\beta = +\infty$ , and Node B passes the same value to its child D.