B. Sc. (H) Computer Science Semester VI
Core Paper XIII – Artificial Intelligence

Topic:

- **Mini-Max Algorithm**
- **Alpha-Beta Pruning**

Mini-Max Algorithm in Artificial Intelligence

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.
Mini-Max algorithm uses recursion to search through the game-tree.
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.
In this algorithm two players play the game, one is called MAX and other is called MIN.
Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
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.

Working of Min-Max Algorithm

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.
In this example, there are two players one is called Maximizer and other is called Minimizer.
Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.
At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs.

Alpha-beta Pruning

Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
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.

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.

The two-parameter can be defined as:
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 -∞.
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 +∞.

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 fast.
Note: To better understand this topic, kindly study the minimax algorithm.

Condition for Alpha-beta pruning:

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

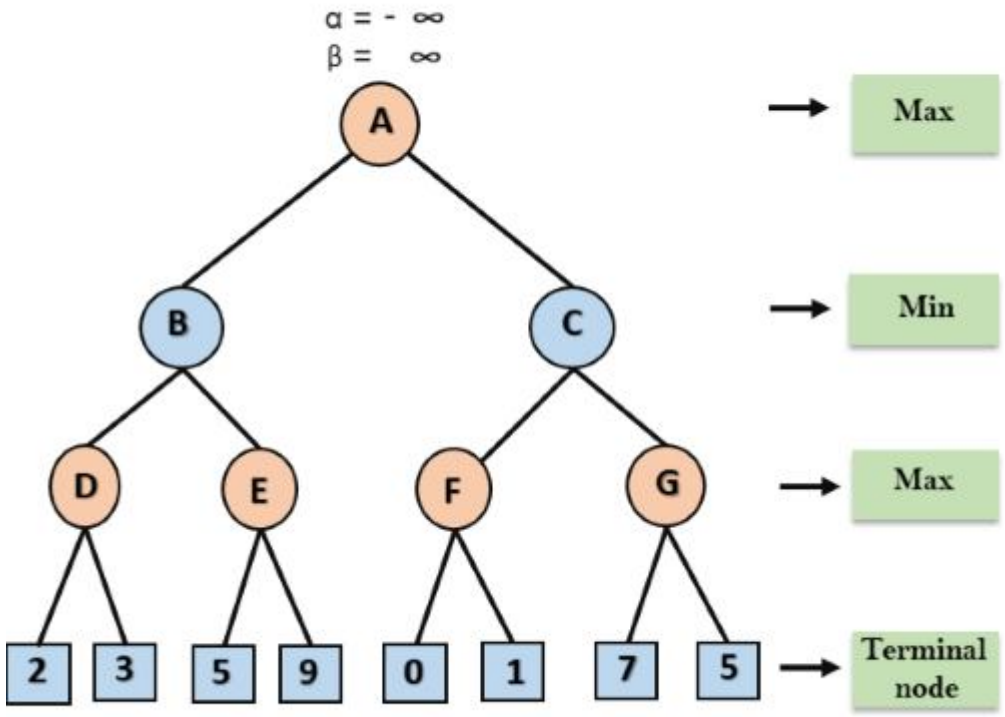
$\alpha >= \beta$

Key points about alpha-beta pruning:

The Max player will only update the value of alpha.
The Min player will only update the value of beta.

While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
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

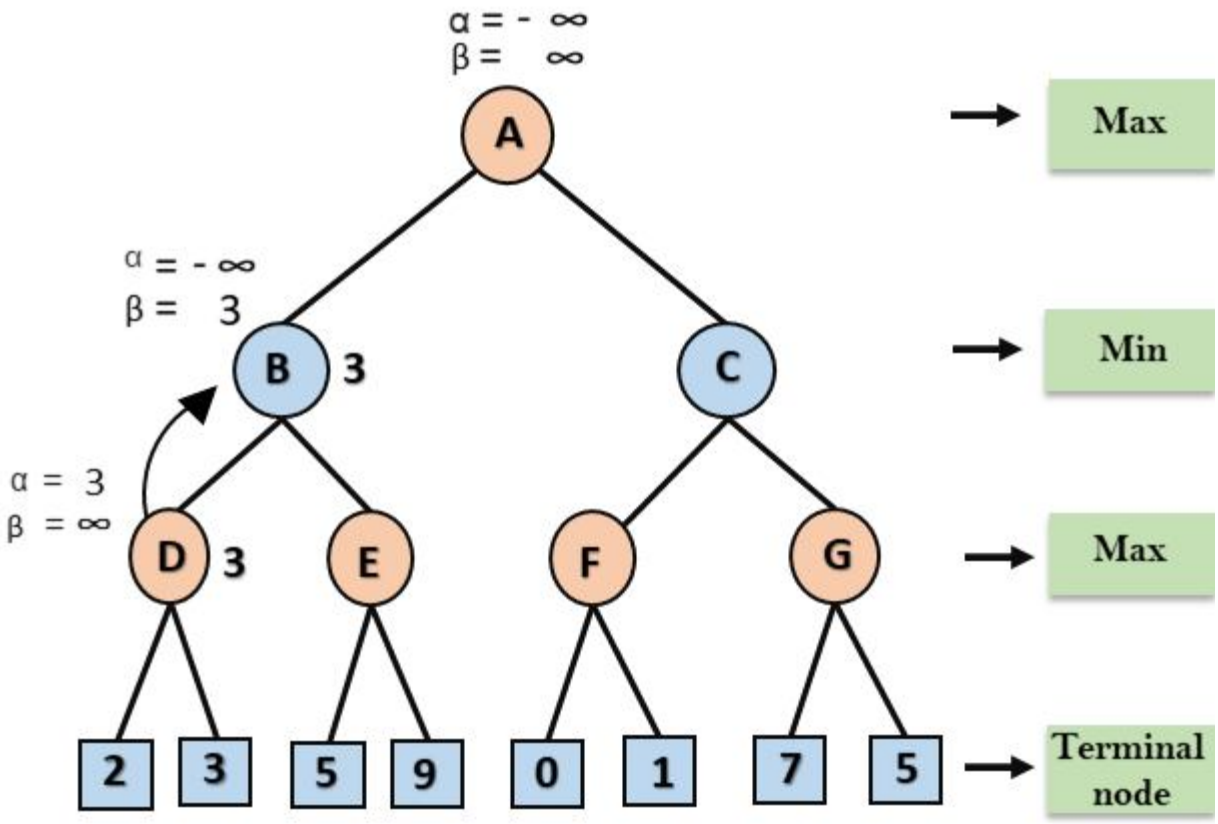
Step 1: At the first step the, Max player will start first move from node A where $\alpha = -\infty$ and $\beta = +\infty$, these value of alpha and beta passed down to node B where again $\alpha = -\infty$ and $\beta = +\infty$, and Node B passes the same value to its child D.



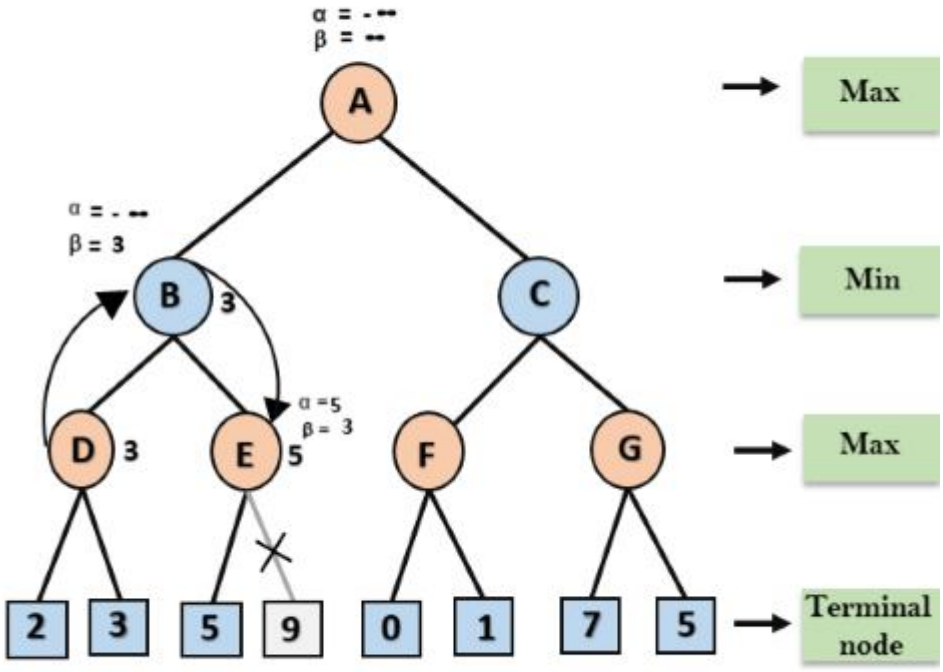
Step 2: At Node D, the value of α will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and node value will also 3.

Step 3: Now algorithm backtrack to node B, where the value of β will change as this is a turn of Min, Now β= +∞, will compare with the available subsequent nodes value, i.e. min (∞, 3) = 3, hence at node B now α= -∞, and β= 3.

In the next step, algorithm traverse the next successor of Node B which is node E, and the values of α= -∞, and β= 3 will also be passed.



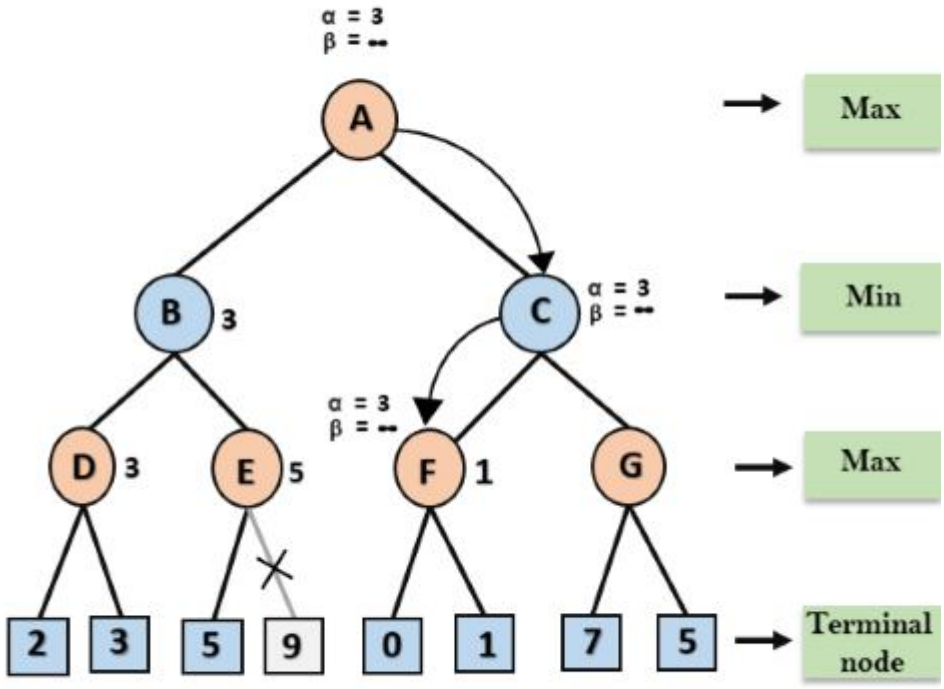
Step 4: At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so max (-∞, 5) = 5, hence at node E α= 5 and β= 3, where α>=β, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.



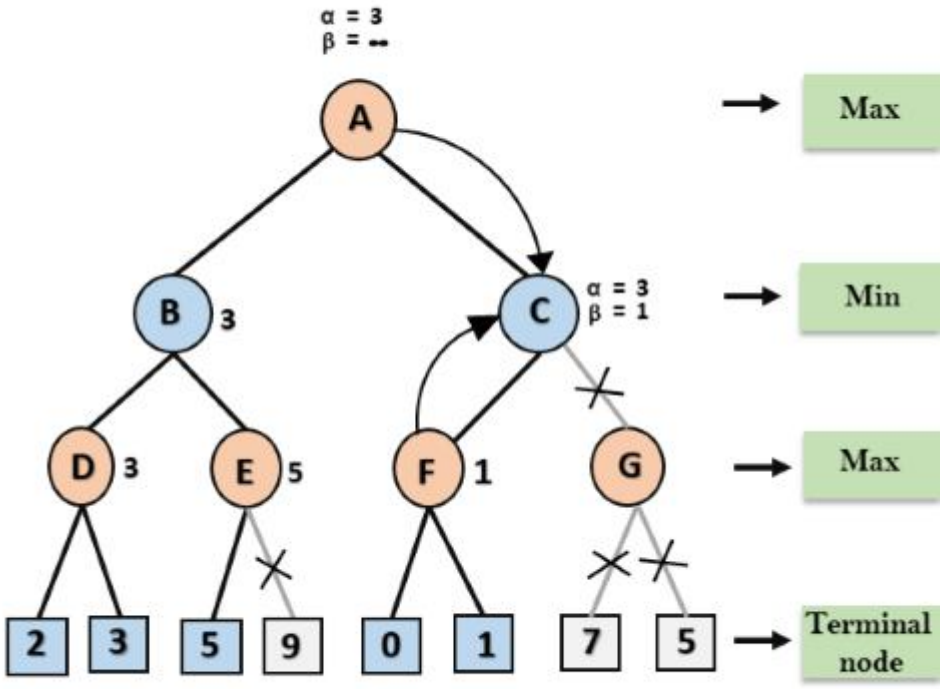
Step 5: At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max (-∞, 3)= 3, and β= +∞, these two values now passes to right successor of A which is Node C.

At node C, α=3 and β= +∞, and the same values will be passed on to node F.

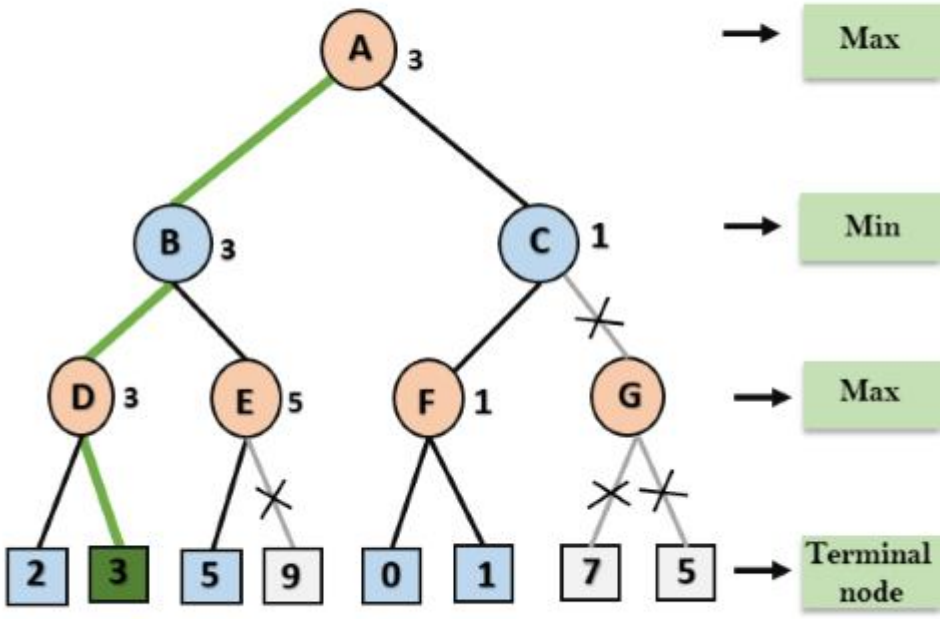
Step 6: At node F, again the value of α will be compared with left child which is 0, and max(3,0)= 3, and then compared with right child which is 1, and max(3,1)= 3 still α remains 3, but the node value of F will become 1.



Step 7: Node F returns the node value 1 to node C, at C α= 3 and β= +∞, here the value of beta will be changed, it will compare with 1 so min (∞, 1) = 1. Now at C, α=3 and β= 1, and again it satisfies the condition α>=β, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.



Step 8: C now returns the value of 1 to A here the best value for A is max (3, 1) = 3. Following is the final game tree which is the showing the nodes which are computed and nodes which has never computed. Hence the optimal value for the maximizer is 3 for this example.



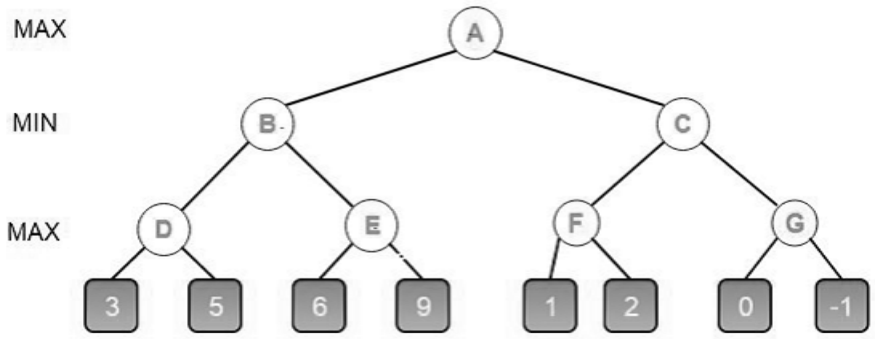
References/Resources

1. Dan. W. Patterson, Artificial Intelligence and Expert Systems, Prentice Hall, 2004
2. Elaine Rich, Kevin Knight, & Shivashankar B Nair, Artificial Intelligence, McGraw Hill, 3rd ed.,2009
3. https://www.javatpoint.com
4. http://www.vssut.ac.in
5. https://career.guru99.com

Assignment

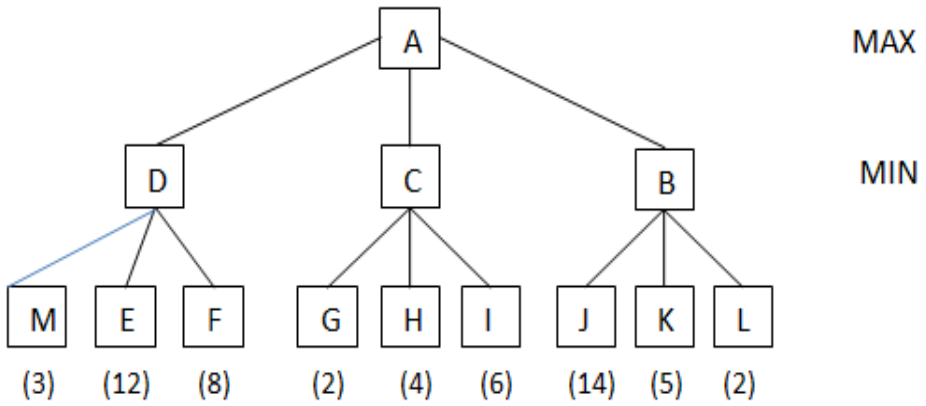
Q1.

In the following two-ply game tree, the terminal nodes show the utility values computed by the utility function. Use the Minimax algorithm to compute the utility values for other nodes in the given game tree.



Q2.

In the following two-ply game tree, the terminal nodes show the utility values computed by the utility function. Use the Minimax algorithm to compute the utility values for other nodes in the given game tree.



Q3. Explain the utility of alpha and beta cuts in Minimax problem.

Q4. What does a production rule consist of?

Q5. Which search method takes less memory?

Q6. Which is the best way to go for Game playing problem?