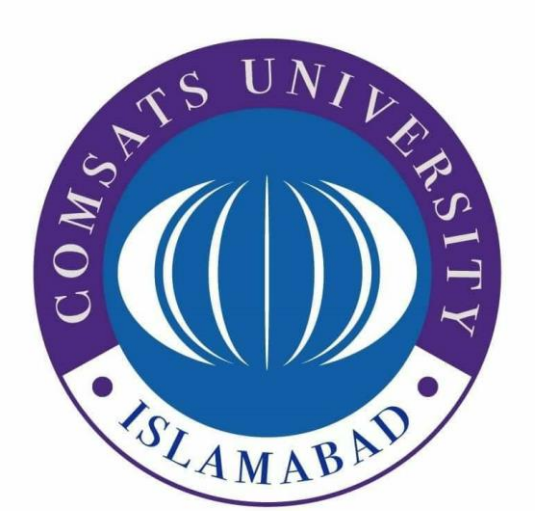


SUBMITTED BY:

Hammad Tufail

Kulsoom Khurshid

REGISTRATION #:

SP20-BCS-028

SP20-BCS-044

COURSE: Artificial Intelligence

SUBMITTED TO: Ma'am Shahela Saif

ASSIGNMENT #: Lab Assignment 1 & 2

Mini-Max Algorithm:

It is a backtracking algorithm which is used for decision making and game theory. It suggests an optimal moves to the player considering the move of other player also optimal. Both the players compete as the opponent gets minimum benefit while they can have maximum benefit. It uses DFS for the complete exploration of the game tree. The shortcoming of this algorithm is that it is time consuming for complex games since such games have huge branching factor and many choices have to be made. To overcome this issue, alpha-beta pruning is used.

Introduction to Alpha-Beta Pruning:

Mini-Max algorithm is modified as an Alpha-Beta pruning. Since the Mini-Max Algorithm explore the game states exponentially, so it cannot be eliminated but we can cut it in half.

Pruning can compute the Mini-Max decision without checking each node. Two threshold parameters are used known as **Alpha** and **Beta.**

- **Alpha:** Along the path of Maximizer it is the best choice that is found at any point. Its initial value is minus infinity.
- **Beta:** Along the path of Minimizer it is the best low value found at any point. Its initial value is positive infinity

Alpha- Beta pruning returns the same move as the Mini-Max algorithm but it removes the nodes that does not affect the final decision infact make the process slow.

Condition for Alpha-Beta pruning:

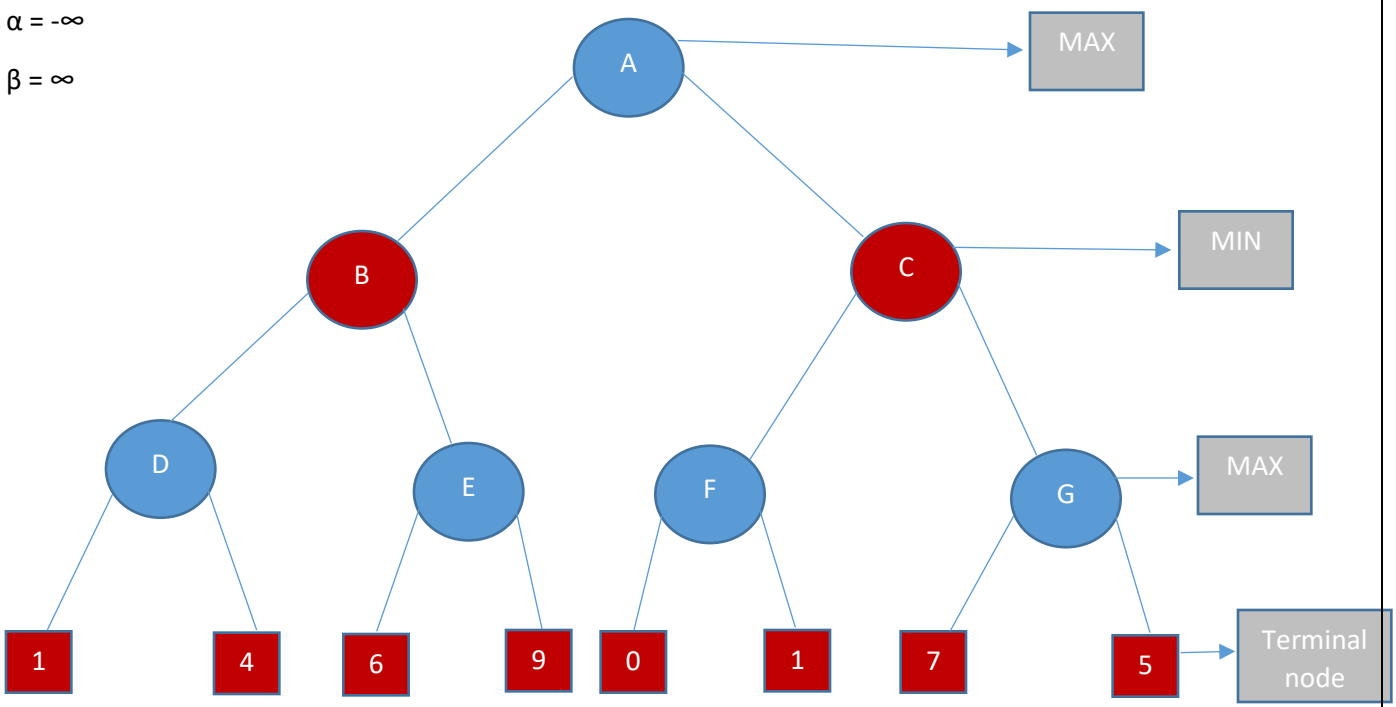
Alpha should always be greater or equal to Beta, i.e. $\alpha \geq \beta$

- Alpha is updated by Max player.
- Beta is updated by Min player.
- Node value is passed to upper node while backtracking instead the value of alpha and beta.
- The child node gets the value of alpha and beta.

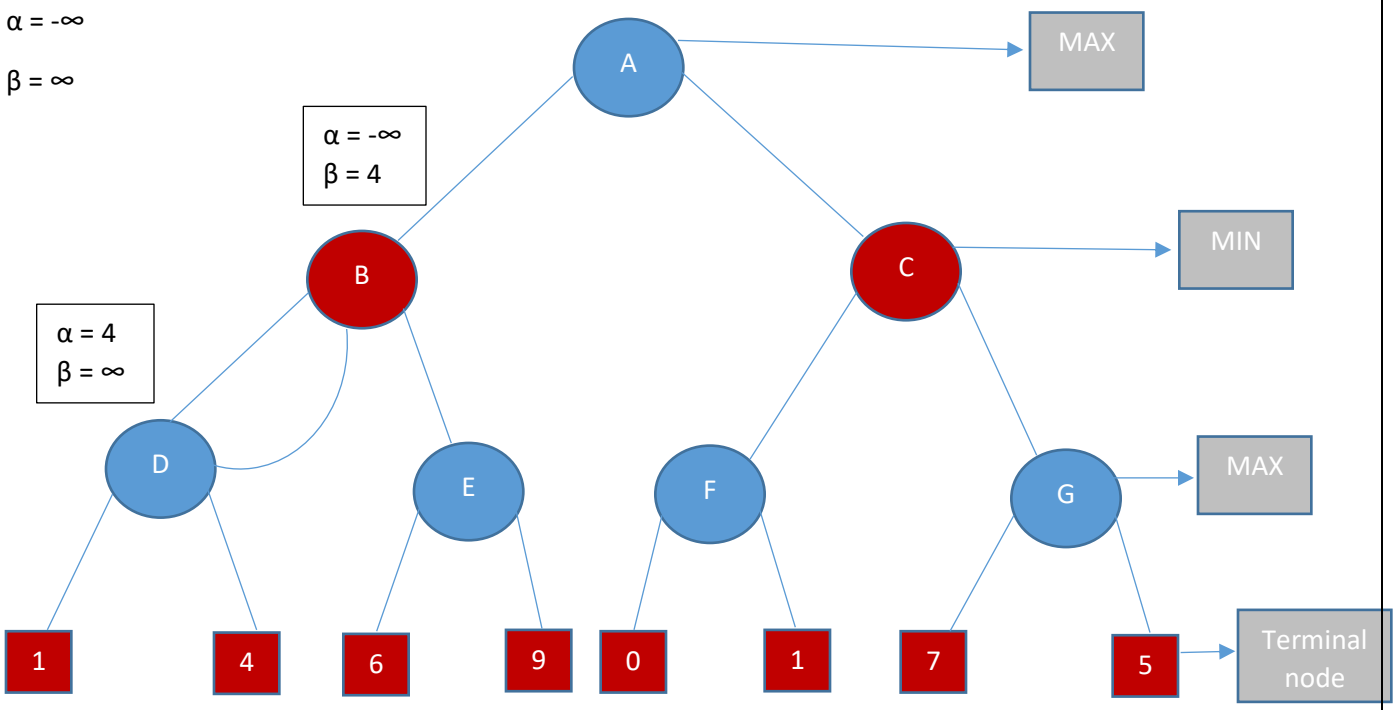
Working of Alpha Beta Pruning:

Consider the following example for Alpha Beta pruning.

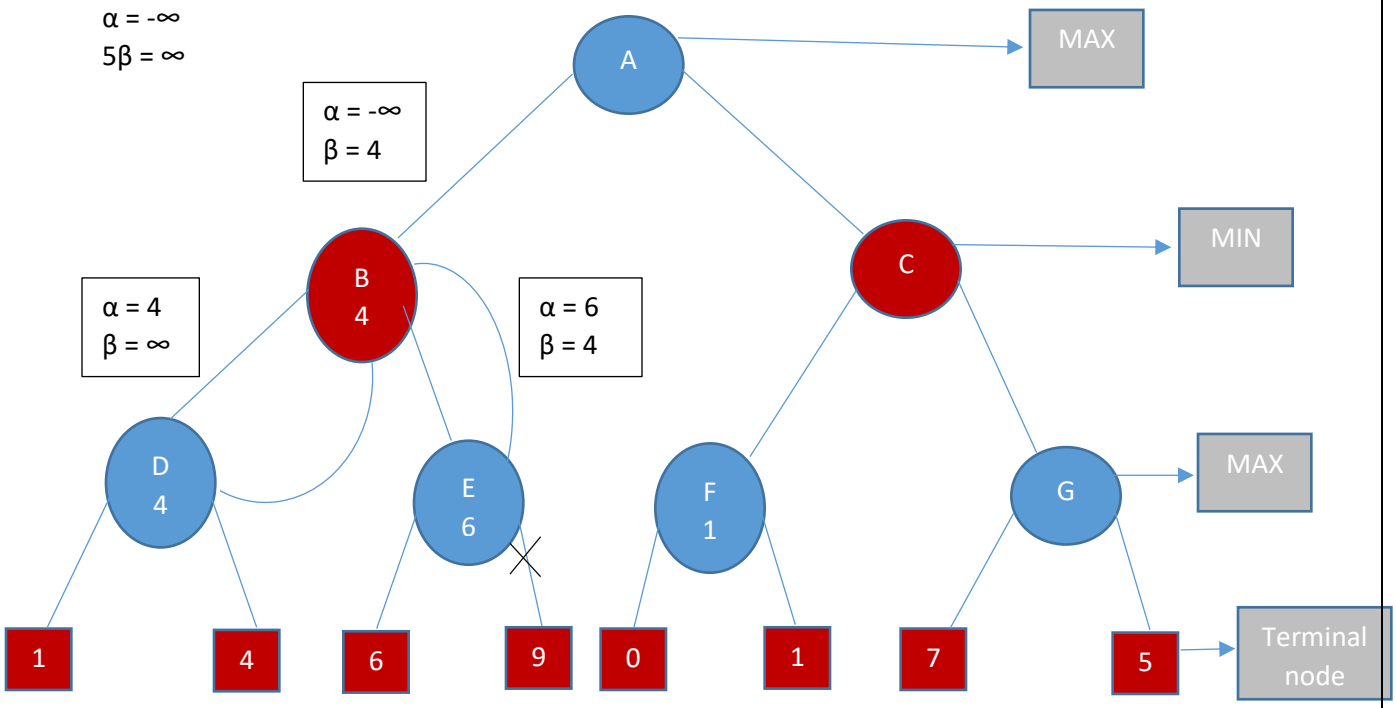
1. Initially the value of Alpha is $-\infty$ and Beta is ∞. The first move will be made by MAX player i.e., Node A which will pass down the values of α and β to its child node. Node B passes down the same value to Node D.



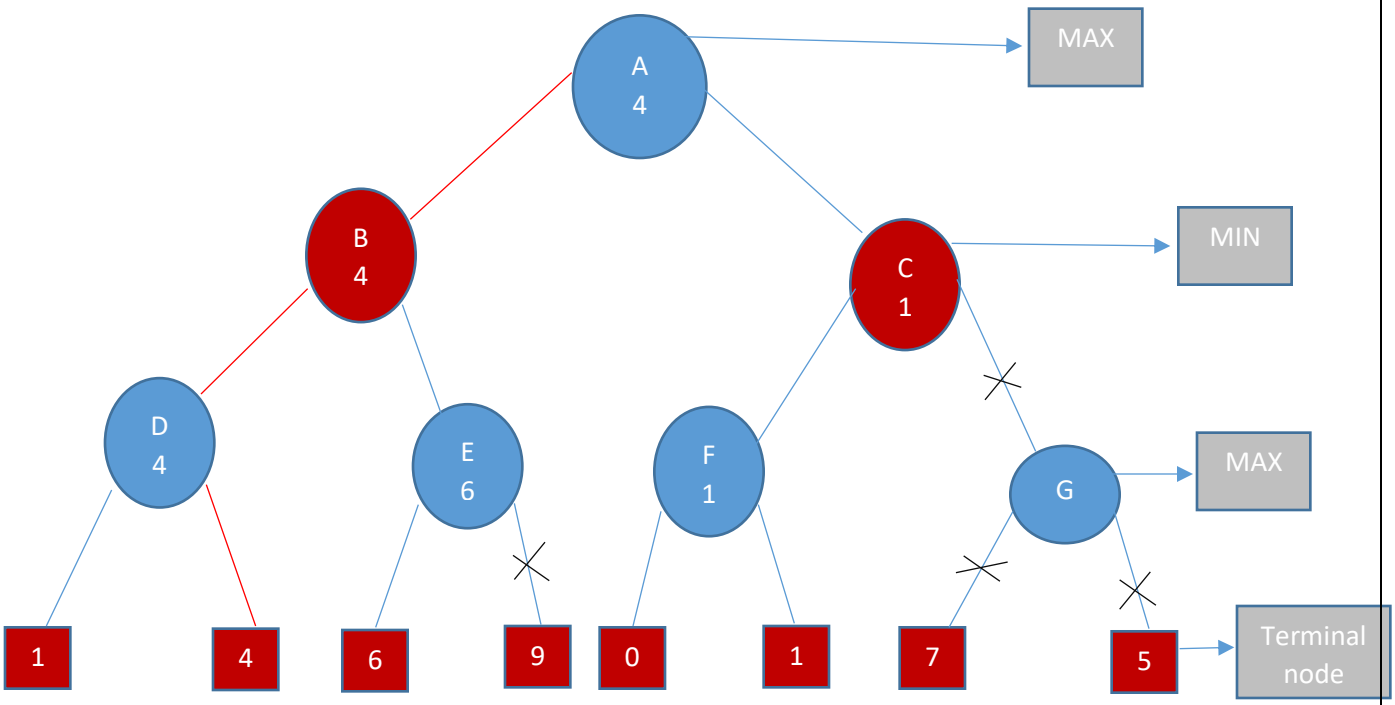
2. Node D will calculate the value of α as it is a MAX turn. This values is compared with 1 and then 4. Max(1,4) will be assigned to alpha at Node D. In this case, max(1,4) = 4. So, α = 4.
3. The algorithm backtracks and move to Node B from node D. in this case, value of β is calculated as it is the turn of Min player. So at D, β = ∞ and α = 4. Min(∞,3) = 3. At node B, α = -∞ and β = 4. The next successor E of B is traversed β = 4 and α = -∞.



4. Its turn for Max, the value of alpha will be updated. Max(-∞, 6) = 6 so α = 6 and β = 4. According to condition, α ≥ β => 6 ≥ 4. Since the condition satisfied, the right successor is pruned.



5. From node B it backtracks to Node A, the value of alpha is updated. The maximum available value is 4. So max(-∞, 4) = 4 and β = ∞ is passed down to its successor node C. From C it is passed down to F.
6. The value of alpha is first compared to 0 and then to 1. Max(0, 4) = 4 and max(1, 4) = 4. Hence, α = 4. Node F will have value 1.
7. Node C has α= 4 and β= +∞, it is Min player's turn hence value of beta will be updated, min (∞, 1) = 1. Hence, α=4 and β= 1, according to condition it is true α>=β, so the right successor of C will be pruned.
8. Node C returns 1 to A, max (4, 1) = 4. Following is the final game tree that shows which nodes are computed and which one is pruned. Hence the optimal value for the maximizer is 4.



Alpha-Beta pruning in Tic Tac Toe:

Utility function:

In case of Max(), -1 is loss, 0 is tie and 1 is win.

Whereas for Min(), 1 is loss, 0 is tie and -1 is win.

Max() is AI whereas Min() is human.

Maximizer:

Initially maxv = -2 which is the worst case, the values of alpha beta are passed down to child nodes which are basically Min. So, min sends the minimum value after comparing to both sides child. Which is stored in m when it returns to max. If m is > maxv, the value of maxv is replaced and the indices are stored in variables which will be returned. The condition is checked if $\alpha \geq \beta$ the other side is pruned else α is updated.

Minimizer:

Initially minv = 2 which is the worst case, the values of alpha beta are passed down to child nodes which are basically Max. So, max sends the maximum value after comparing to both sides child. Which is stored in m when it returns to max. If m is < minv, the value of minv is replaced and the indices are stored in variables which will be returned. The condition is checked if $\alpha \geq \beta$ the other side is pruned else β is updated.

Basically, if the AI is winning human is trying to minimize the win and if human is winning AI has to minimize human's win.