Alpha–beta pruning

There are many algorithms for artificial intelligence, or AI for short, in games. One such algorithm is named Alpha-beta pruning. This algorithm has many interesting characteristics that makes it useful in game development.

The Alpha-beta pruning algorithm is a search algorithm that tries to lower the number of nodes that it searches within its search tree by using the minimax algorithm, so the Alpha-beta pruning algorithm is an extension of the maximin algorithm. Minimax is a decision rule for minimizing the possible loss for a worst case scenario. This algorithm was originally for two-player games where the player took alternate turns or the players took actions simultaneously. The pseudocode for this algorithm is:

```
Minimax(node, depth, maximizingPlayer)
        if(depth = 0 || node is terminal node)
                return node heuristic value
        if(maximizingPlayer)
                bestVal = neg-infinity
                for(child node)
                        v == Minimax(child, depth-1, false)
                        bestVal == max(bestVal, v)
                return bestVal
        else
                bestVal = infinity
                for(child node)
                        v == Minimax(child, depth-1, true)
                        bestVal == min(bestVal, v)
                return bestVal
```

This pseudocode is only for the maximin algorithm and not the Alpha-beta pruning algorithm. Although, the minimax algorithm is a major part of the Alpha-beta pruning algorithm it still does not have the full functionality of the Alpha-beta pruning algorithm. So what is the basics of the Alpha-beta algorithm that makes it different.

The Alpha-beta pruning algorithm, which can also be called an adversarial search algorithm, is actually different from the minimax algorithm in a few ways, but there is one big difference. The algorithm is commonly used when a machine is playing a two-player game, such as chess, go, or even tic-tac-toe. The algorithm only stops when it has completely evaluated a move, which means that it finds at least one possibility that proves the move to be worse than a previously examined move. It essentially returns the same move that minimax would, but it also prunes away branches, hence the word pruning in the name, that do not influence the final decision. Now that we have a quick overview of what the algorithm does then it is time to go into more detail.

The Alpha-beta algorithm has two values, alpha and beta. One of the values represents the minimum score that the maximizing player is assured of and the other value is the maximum score that the minimizing player is assured of. At the start of the algorithm the value of alpha is negative infinity and the value of beta is positive infinity. If the maximum score that the minimizing player is assured becomes less than the minimum score that the maximizing player is assured, meaning beta <= alpha, then it means that the parent node should not choose that particular node. The reason for this is because it will make the score for the parent node worse, this also means the other

branches of the node do not have to be explored. The pseudocode for the Alpha-beta algorithm is:

```
AlphaBeta(node, depth, alpha, beta, maximizingPlayer)
        if(depth = 0 || node is terminal node)
                return node heuristic value

        if(maximizingPlayer)
                v = neg-infinity
                for(child node)
                        v == max(v,AlphaBeta(child,depth-1, alpha, beta, false))
                        alpha == max(alpha, v)
                        if(beta <= alpha)
                                break
                return v
        else
                v = infinity
                for(child node)
                        v == min(v, AlphaBeta(child, depth-1, alpha, beta, true))
                        beta == (beta,v)
                if(beta <= alpha)
                        break
                return v
```

Since the other branches of the node do not have to be explored then this makes the search faster and faster the more it figures out which branch not to explore. This is the true advantage for the Alpha-beta algorithm.

The Alpha-beta algorithm has the interesting quality that it can diminish the amount of branches and nodes that it has to search. Its other qualities allow it to be a good algorithm to use when making games. This however is not a full explanation for the Alpha-beta algorithm and only gives the basics of the algorithm.

Resources

https://en.wikipedia.org/wiki/Alpha–beta_pruning
https://en.wikipedia.org/wiki/Minimax