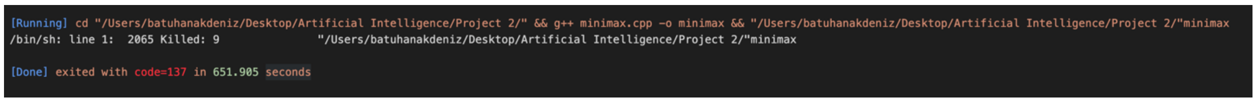
AI-Optimal-Decision-in-Games

Comparing the Basic minimax algorithm and alpha beta pruning

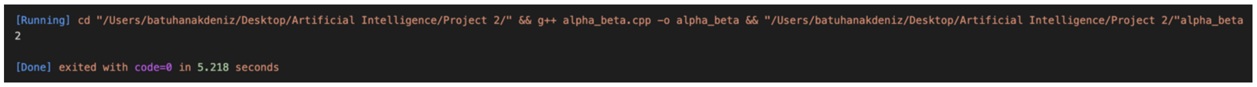


```
[Running] cd "/Users/batuhanakdeniz/Desktop/Artificial Intelligence/Project 2/" && g++ minimax.cpp -o minimax && "/Users/batuhanakdeniz/Desktop/Artificial Intelligence/Project 2/"minimax
/bin/sh: line 1:  2065 Killed: 9               "/Users/batuhanakdeniz/Desktop/Artificial Intelligence/Project 2/"minimax

[Done] exited with code=137 in 651.905 seconds
```

Figure 1- Basic minimax result

This is the result for basic minimax algorithm. This algorithm can not be runned because time and memory space which it takes give an error from my computer.

In basic minimax algorithm all the nodes that can be possible are created. So, it can not be imlemented for big problems because it will take a lot of time and memory space. Because time complexity of this type of minimax algorithm is O(b!).



```
[Running] cd "/Users/batuhanakdeniz/Desktop/Artificial Intelligence/Project 2/" && g++ alpha_beta.cpp -o alpha_beta && "/Users/batuhanakdeniz/Desktop/Artificial Intelligence/Project 2/"alpha_beta
2

[Done] exited with code=0 in 5.218 seconds
```

Figure 2 – Alpha Beta Pruning Approach Minimax Result

On the other hand, alpha beta style minimax algorithm is faster and provides less memory space usage. This new approach of minimax is give an result for input_2-2.txt and it is shown on figure 2.

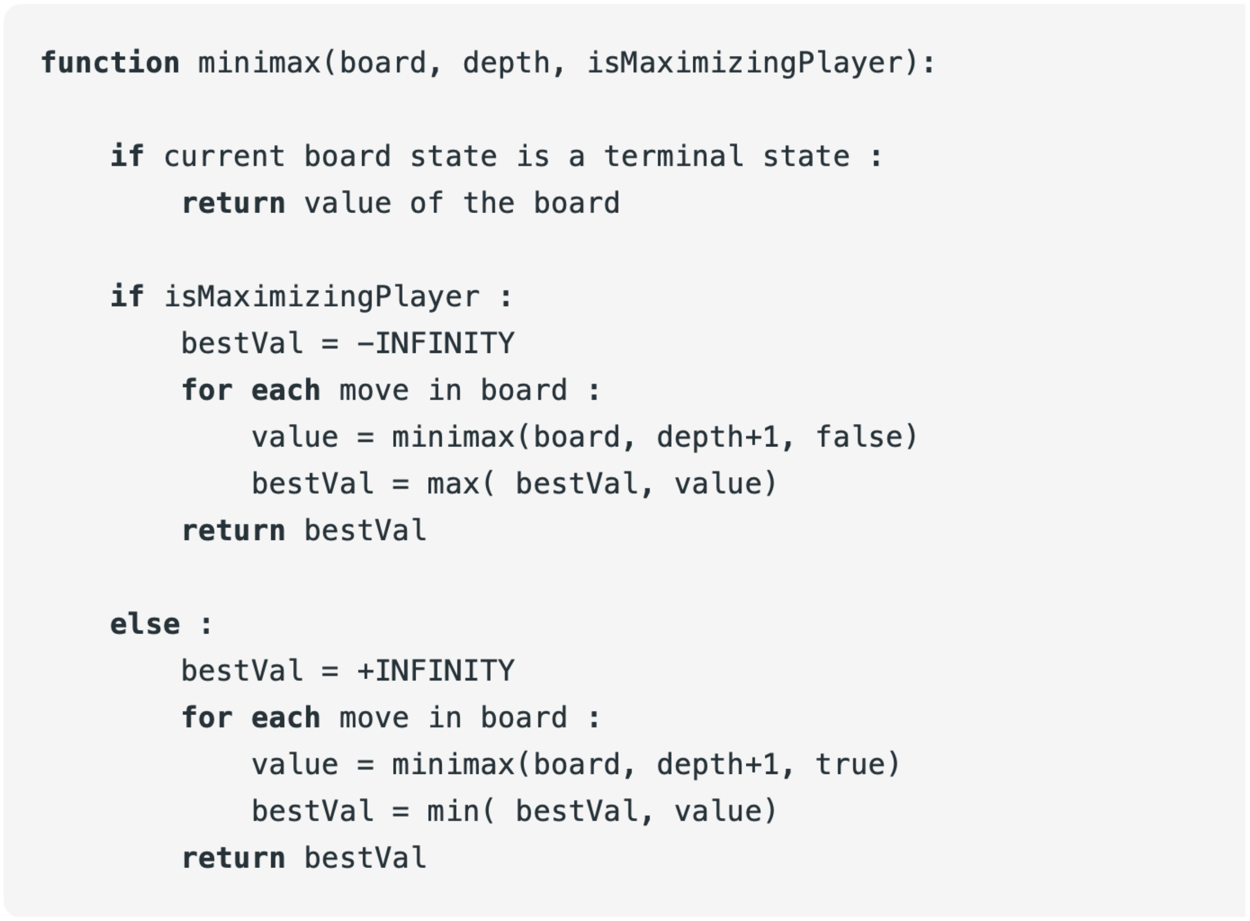
1) Number of created Nodes

For the basic minimax algorithm every nodes on tree is searched. So, it can take a lot of time. But alpha beta pruning approach searching every nodes in tree is not necessary. Because is there any better option in tree the worse side of the will not be searched. So, it takes less time obviously thanks to this method.

2) Running time

According to running time there is huge different. In figure 1, basic minimax algorithm's result is shown and after the 651 second it is not ended. Besides, alpha beta pruning approach minimax found the result in 5 second as shown in figure 2.

Minimax Algorithm

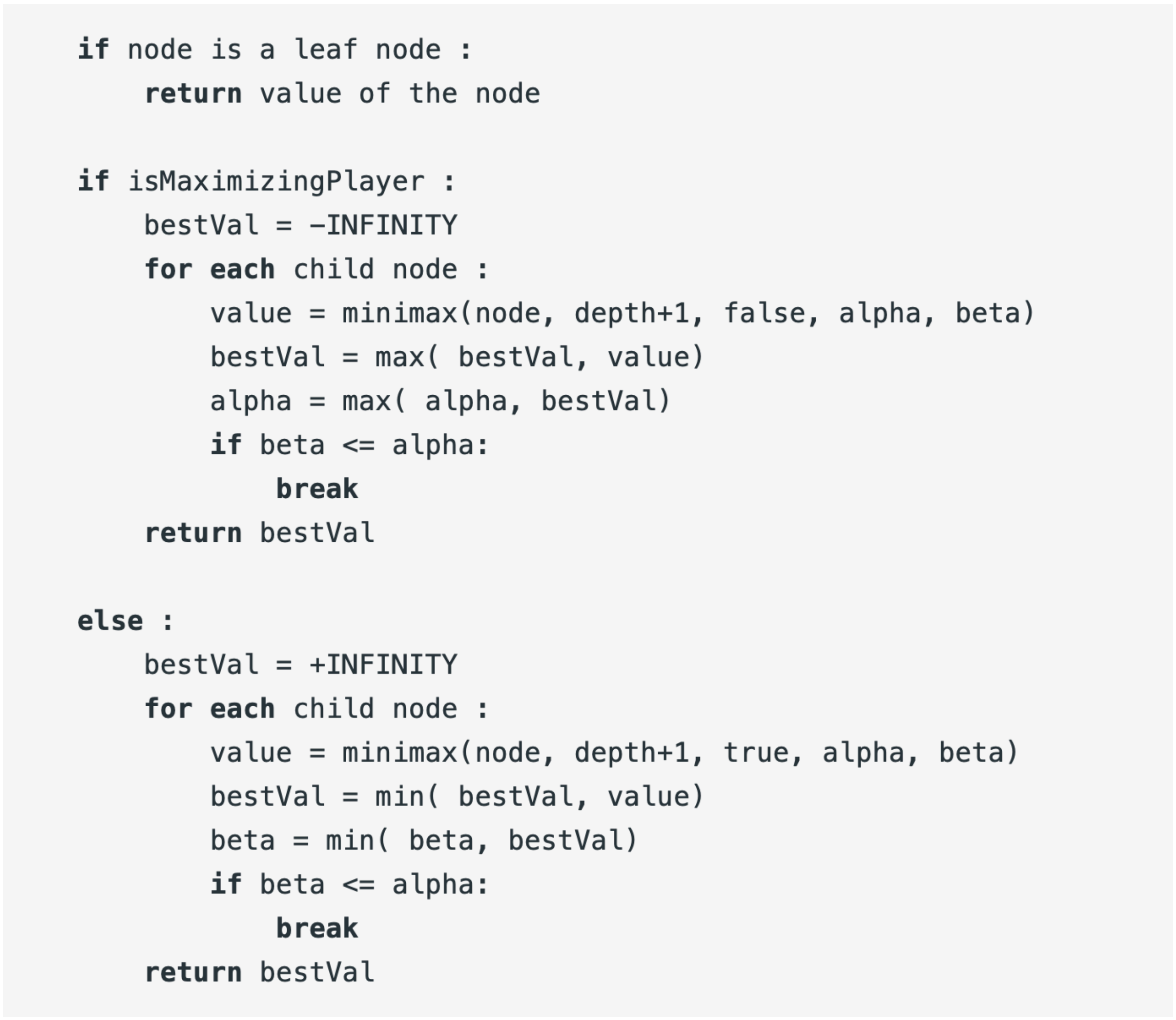


```
function minimax(board, depth, isMaximizingPlayer):

    if current board state is a terminal state :
        return value of the board

    if isMaximizingPlayer :
        bestVal = -INFINITY
        for each move in board :
            value = minimax(board, depth+1, false)
            bestVal = max( bestVal, value)
        return bestVal

    else :
        bestVal = +INFINITY
        for each move in board :
            value = minimax(board, depth+1, true)
            bestVal = min( bestVal, value)
        return bestVal
```

Figure 3 – Basic minimax algorithm pseudocode

Minimax algrithm is implemented. This pseudocode is taken from https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move/?ref=rp . In this pseudocode, it can be said this function works as recursively. In my implementation firstly available children are created. Then check for the node which is coming from agrument of minimax function is end node. Then if is Maximum player's turn it will take best value as an incredibly negative value. Then recursively minimax function is called. In this calling isMax variable is set as false because max player is done and the turn is minimum player's. Thanks to this approach function creates nodes and check this nodes in backtracking style.

Alpha Beta Approach



```
if node is a leaf node :
    return value of the node

if isMaximizingPlayer :
    bestVal = -INFINITY
    for each child node :
        value = minimax(node, depth+1, false, alpha, beta)
        bestVal = max( bestVal, value)
        alpha = max( alpha, bestVal)
        if beta <= alpha:
            break
    return bestVal

else :
    bestVal = +INFINITY
    for each child node :
        value = minimax(node, depth+1, true, alpha, beta)
        bestVal = min( bestVal, value)
        beta = min( beta, bestVal)
        if beta <= alpha:
            break
    return bestVal
```

Figure 4 – Alpha Beta approach's pseudocode

Alpha Beta style minimax algorithm is implemented and this pseudocode is taken from same website. In this implementation new parameters are added which is called as beta and alpha. With these variables if there is any not necessary nodes to search, it will ignored. This new approach to minimax algortihm if there is any better way in tree it is not searched and it gives less time to computation and less memory space to tree.

Compile and run

Use terminal to compile and run this code. Use these commands. Files and input files must be in same directory.

```
gcc -o minimax minimax.cpp
./minimax input_file_name
gcc -o alpha_beta alpha_beta.cpp
./alpha_beta input_file_name
```