

**Artificial intelligence**

**Major task**

**Phase 2 report**

**Ali Khaled Ragy 20P1342**

**Tamer Ihab 20P5567**

**Usama Ahmed 18Q9584**

**Presented to**

**Dr. Mahmoud Khalil**

**Eng. Hazem Ayman**

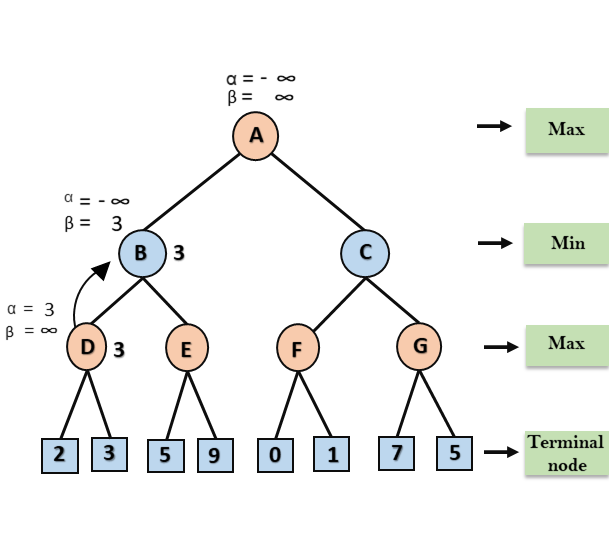
How our program works?

Once the game starts, the agent takes a copy of the board state on the spot, then it starts iterating on all of the possible moves that it could take on the next turn. The program firstly checks if the move is valid, and if it was valid, the program starts giving scores to every possible move it could take based on the the alpha beta pruning function and stores the scores in a table to re-check them. After giving scores to every possible move, the agent compares the scores with each other, and decides to take the move of the highest score.

How the alpha beta pruning works generally ?

Alpha beta pruning is an algorithm that gives scores to every possible move that can be played to get the best scenario. But Alpha beta pruning is more advanced program than the minimax algorithm. As it does not iterate on all the possible moves to get the best one, but it starts to compare the scores with the previously examined moves during in this level. And it starts to prune the moves that are guaranteed to be worse than the previously eliminated moves. This ensures that the program would be much faster and more optimized and time-efficient.

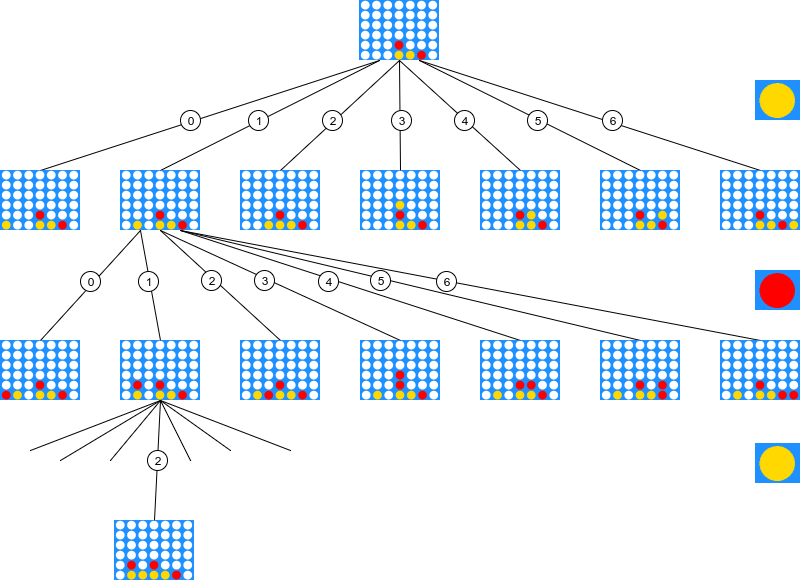
How alpha beta pruning prunes the undesired tree branches ?

The program starts with an initial alpha value of negative infinity and a beta value of positive infinity. These values represent the best scores found for the maximizing and minimizing player, respectively.

Then the algorithm begins the recursive search through the game tree, evaluating nodes in a depth-first manner. At each depth of the tree, the algorithm alternates between the maximizing and minimizing player.

For the maximizing player, the alpha value is updated if a higher value is found during the search. Alpha represents the best score found so far for the maximizing player.

For the minimizing player, the beta value is updated if a lower value is found during the search. Beta represents the best score found so far for the minimizing player.

Before exploring a child node, the the alpha and beta values are compared. If alpha is greater than or equal to beta, it means that the current branch cannot influence the final decision, and pruning can be applied. The algorithm can stop evaluating further nodes in that branch and return the alpha or beta value back to the parent. As the search moves up the tree, the alpha and beta values are propagated to higher levels. For the maximizing player, the alpha value is updated with the maximum value found among its child nodes. For the minimizing player, the beta value is updated with the minimum value found among its child nodes.

By pruning branches that are determined to be worse than previously examined moves, alpha-beta pruning can significantly reduce the number of nodes that need to be evaluated, leading to a more efficient search and faster decision-making in game-playing algorithms.

We can simply sum up the minimax and alpha beta pruning algorithms by the following. The maximizing player always has the goal to maximize his score to the max level, while the minimizing player has the goal to minimize the maximizing player’s score.

Evaluation Function

A screenshot of a computer code

Description automatically generatedOther than the Alpha beta pruning algorithm, The program also depends on an evaluation function. The evaluation function simply goals to give further and more wide-sighted evaluation to the game board situation right now.

The evaluation function is not generally used in the program, it is only used when the alpha beta pruning algorithm is stuck in 2 or more moves that have similar scores.

In our program for example, the evaluation function works on several approaches,

And here are two examples.

Firstly, our evaluation function checks if the opponent has 2 or more discs adjacent

to each other by any means, or the agent itself is having 2 or more discs adjacent to each other.

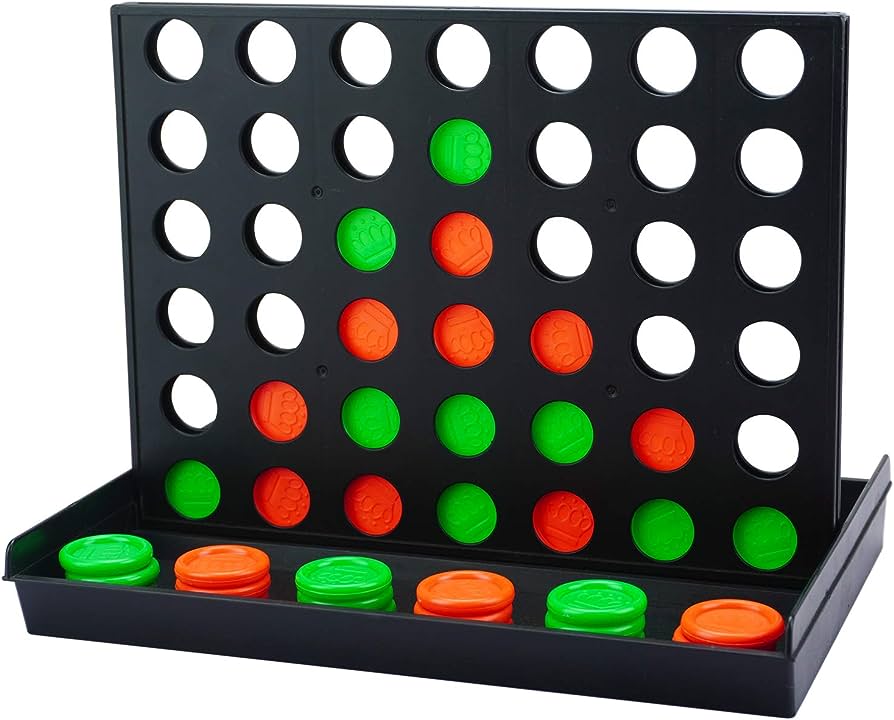
Secondly, the evaluation function checks if there are discs adjacent to each other and there is just an empty space between them, and it gives this situation a score of a smaller weight as It is less risky.

A screen shot of a computer code

Description automatically generated And based on the situations and their risk, the evaluation function gives a score to the overall situation and gives scores to the possible upcoming moves in a different approach than the alpha beta pruning algorithm. And based on this evaluation function, the agent determines the best move.

Moreover, the evaluation function checks for threats and double threats, the threat is simply when there are 3 discs adjacent to eachother and have a chance to win from one side, while the double threat is when there is a chance to make a winning move from 2 sides.

Finally, an additional score is given when the agent takes the middle colummn as it is definitely more valuable in the game process.



Double threat situation