De La Paz, Jhoanna Alexandra C.
J3A - AI
Midterm Activity 2

Minimax Tic-Tac-Toe (Player vs. AI): Documentation

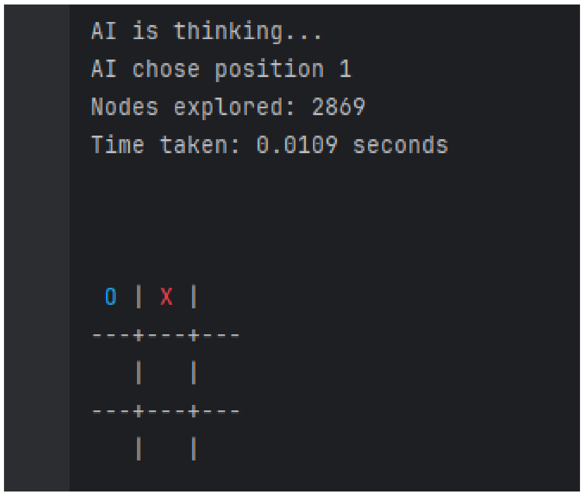
Explanation of Minimax

The Minimax algorithm is a way for computers to make decisions in games like Chess or in this case, Tic-Tac-Toe. It works by assuming that one player wants to get the highest score (MAX) while the other wants to lower the other one's score (MIN) to prevent them from winning, and the algorithm alternates between those two components by their turns. The algorithm looks at all possible moves and their results, then picks the move that gives the best result even if the opponent plays perfectly. In short, Minimax helps a computer think ahead by evaluating the best move to play based on future possibilities.

Alpha-Beta Pruning Effect

The Alpha-Beta Pruning improves the Minimax algorithm by skipping unnecessary nodes that cannot possibly affect the final decision. This means it doesn't have to check every single possibility, which saves time and produces faster move computation. For example, without pruning, the algorithm might explore all nodes (example: 1000+ nodes), but with pruning, it could explore only around 200-500 nodes while still choosing the best moves.

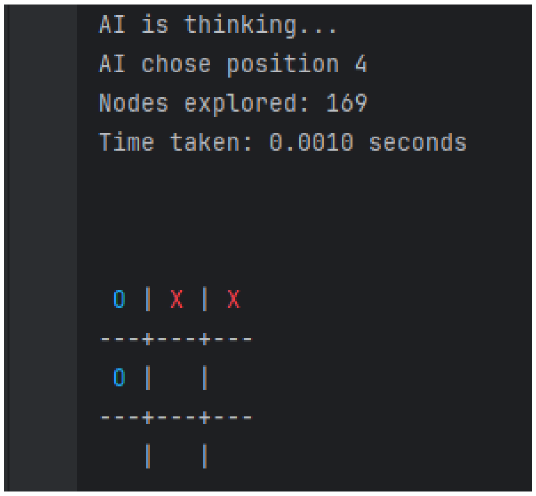
In my program, I have displayed the nodes and timing with Alpha Beta Pruning, and it proves that pruning makes the Minimax algorithm work faster and more efficiently, especially when the game has many possible moves or a large number of branches to explore



```
AI is thinking...
AI chose position 1
Nodes explored: 2869
Time taken: 0.0109 seconds



 O | X |
---+---+---
   |   |
---+---+---
   |   |
```



```
AI is thinking...
AI chose position 4
Nodes explored: 169
Time taken: 0.0010 seconds



 O | X | X
---+---+---
 O |   |
---+---+---
   |   |
```

Reflection

One thing I learned is how convenient Alpha Beta Pruning is. It is also very important for a program to have good optimization for it to perform better and faster. The challenges I faced would probably be about how Minimax and Alpha Beta Pruning really works, especially how to apply them in terms of code. I understand the main concepts, but I'm still a bit confused about the detailed process. I get that Minimax helps choose the best move by looking ahead and that pruning skips unnecessary paths, but I find it tricky to fully follow how the values update and when exactly branches are cut off during the search.