Homework 1 - Documentation for Omok AI

아니스 카티자 (2022320060)

1. Search Algorithm

The AI implements Iterative Deepening Heuristic Alpha-Beta Search using several optimization techniques to efficiently select the best possible move within a given time limit

1. Iterative Deepening, which the AI searches incrementally deeper in depth until the time limit is reached. This ensures the AI always has a *best\_move* ready, even if interrupted. Implemented in *iterative\_deepening\_search()*
2. Alpha-Beta Pruning is used to optimize the minimax algorithm by cutting off branches that cannot affect the final decisions. It also uses alpha, which is the best score for maximizing player, and beta for the best score for the minimizing player, to prune unnecessary searches. This can be found in the function *alpha\_beta\_search()*
3. The search for the best decisions is cut off when the allocated time per move *time\_limit* is exhausted. This is to ensure the AI’s real-time responsiveness and a faster move
4. Transposition Tables, which uses a dictionary to cache previously evaluated positions to avoid redundant calculations
5. Killer Move Heuristics remembers strong moves from previous searches . Killer moves are prioritized in subsequent searches to improve efficiency through better move ordering

2. Heuristic Evaluation Function

1. Pattern Recognition: the evaluation function scores board positions based on recognized patterns

|  |  |  |
| --- | --- | --- |
| Pattern | Score | Description |
| 5-in-a-row | 1,000,000 | Immediate win |
| Open four (4 with 2 open ends) | 100,000 | Guaranteed win next turn |
| Half-open four (4 with 1 open end) | 50,000 | Needs one more to win |
| Open three (3 with 2 open ends) | 10,000 | Strong potential to make four |
| Half-open three (3 with 1 open end) | 2,000 | Potential to make four |

1. Strategic Features
2. Center Control: Bonuses for central positions (decreasing with distance from center)
3. Simultaneous Evaluation: Both player and AI are evaluated to handle threats and opportunities
4. Pattern Matching: Checks all directions, whether horizontal, vertical, or both diagonals for continuous sequences
5. Open End Consideration: Detects whether the patterns are open on both ends, one end or closed. Patterns with two open ends are prioritized
6. Evaluation Process
7. Check for immediate wins or losses
8. Scores all stone patterns for both players
9. Add bonuses for center proximity
10. Return final score for use in search algorithm

3. How To Compile and Run the Program

1. Run the game using the terminal or command line:

*python main.py*

1. Since the implementation is for player versus AI, you will play as the Black stone (○), while the AI will play as White (●)
2. Enter your desired coordinate and best luck in winning against the AI!

4. Final Thoughts

I am not good at playing the Omok game, so I have never won against this AI I made myself. I do wish for a better player to play against my AI so that I can observe its tactics in playing Omok, and how far it can go.