Extreme TicTacToe Bot

Team name: Filthy Pijuns (Team 9)
Team members: Neal Karpe, Gunjan Karamchandani

1. Heuristic

Board - The entire *16*16* game board.
Block - A *4*4* sub-matrix of the board.
Cell - A *1*1* box on the board.

- The overall idea is to calculate a score for each block and combine these scores to obtain the entire heuristic.
- Our heuristic is *offensive.* It fully aims to win and does not try to accumulate points.

Weightage Matrix

Following is the *position_weight* matrix.

2	3	3	2
3	4	4	3
3	4	4	3
2	3	3	2

- *position_weight[i][j]* is determined by the number of winning patterns that include position *(i,j)*.
- For example, *position_weight[1][1] = 4* because there are two diamonds, one row and one column that pass through *(1,1)*.

Block Score

- A block won by us is given a score of 30.
- A block which has been drawn or won by the opponent is marked with -1.
- For a block that is still under contention, the score is calculated as follows -
    - **Patterns -** Contribution of all partially filled winning patterns (*total 12* **-** 4 rows, 4 cols, 4 diamonds) is added to the block heuristic.
        - If there is any opponent marker at all in the pattern, it is discarded (since there is no chance of winning that pattern).

        - If the pattern doesn't contain any opponent marker, then the contribution of the pattern to the block heuristic is-
            - **3 -** If ½ of the pattern is full
            - **11 -** If ¾ of the pattern is full
            - **0 -** If pattern is empty or ¼ full. (Note that ¼ full means just one cell, and that is already taken care of in the **cells** contribution)
    - **Cells -** Contribution of all cells (which we have placed our marker on) to the block heuristic.
        - Contribution of the cell at position (i,j) in the block contributes 0.1**position_weight[i][j]* to the block score.

Board Heuristic

- Block Score was calculated by considering the patterns and positions of cells in a particular block. Now that we have all the Block Scores, we consider the patterns and positions of these blocks to get the complete Board Heuristic.
    - **Patterns -** Contributions of partially filled winning patterns of blocks are added to the board heuristic.
        - If there is a block that has been drawn or won by opponent in the pattern, it is discarded.
        - If none of the blocks in the pattern have been drawn or won by opponent, then the contribution of the pattern is calculated by *multiplier*(sum of scores of blocks on the pattern)*. We select *multiplier* as-
            - **1.1 -** If two blocks in the pattern have already been **won** by us.
            - **2.3 -** If three blocks in the pattern have already been **won** by us.
            - **1 -** If zero of one block in the pattern have already been **won** by us. (Note that having one block is anyways included in the **blocks** contribution)
        - If the pattern consists of 4 blocks won by us, that means we have won the game ;)
    - **Blocks** - Contribution of each block at position *(i,j)* that has not been drawn or won by opponent:
        - *0.02 * block_score[i][j] * position_weight[i][j]*

2. Search Technique

- We used minimax algorithm with alpha beta pruning to find the best move at each turn.
- We used Iterative Deepening Search (IDS) to progressively increase the search depth, starting from 3.
- After each iteration search depth increases by 1, and the best move is updated to the newly found best move.
- This process is aborted after 15 seconds.

3. Other Optimizations

- **Transposition table with Zobrist Hash:** We used Zobrist Hashing to store board states and block states in the transposition table. If the same block or board is seen again in the future, its heuristic need not be calculated all over again.
    - Another optimisation was - at every state in the minimax recursion, the entire Zobrist Hash of the state is not calculated again. Instead we keep a global Hash - when you enter the state its value is added to the hash (XOR operation), and when you return, the value of the state is removed from the hash (XOR operation).
- **No deepcopy:** In the minimax recursion, the same board is used everywhere (it is passed as a parameter). When you enter a state, the board is updated with the new marker & before leaving the state, that marker is removed from the board.