Name: Adarsha Poudel

Architecture:

Representation of the board:

I use the struct named *state* to represent the board of each state. The struct contains a pointer to a two-dimensional char array which represents the board in some state.

Classes and Private variables:

The important private variables include *boardSize, AIPiece, HumanPiece, and depth* (of minimax). All of them are self-explanatory terms and serve the same meaning.

Constructors:

The simple purpose of the constructor of gobang class is to initialize boardSize, AIPiece, and the depth of minimax algorithm.

Classes used to for the heuristics:

These are the classes used for heuristics purposes: horizontal, *vertical, diagonal, countPattern, and evalFunc.* So, the functions horizontal, vertical, and diagonal gets the board representation of same states, and uses the board to get its horizontal, vertical, and diagonal representation. Then, these classes use *countPattern* function to count specific threat patterns for this game. And finally, *evalFunc* simply adds value achieved from horizontal, vertical, and diagonal functions and returns the evaluated value for specific move.

Classes used for implementing minimax:

These are the classes used: *AIOptimalMove, minmax*. Every time it is AI's turn to make move, *AIOptimalMove* is called. It takes in current state of board, and applies *minmax* algorithm to available pieces to get evaluation values for each available piece, and finally acts as a maximizer to get a move which has a highest evaluated value. This is how AI's move is determined.

Important Utilities function:

gameEnds: check whether game ends.

printBoard: prints the board

hasPiecesNearThem: checks whether a specific position in the board has pieces near them with the distance of less than 2.

Search

For evaluating possible optimal move for AI, I have classified several specific threats pattern and scored the move accordingly. For example, some of the most common threatful pattern are listed below.

Pattern b = black, w = white	Score
"bbbbb" => The Straight four	Highest
"XbbbbX" => Non-refutable four	Highest
"Xbbbbw" => refutable four	Very high
"XbXbbX" => refutable broken three	High

I have also given a very low score for consecutive ones and twos.

To find the optimal move for AI, I used minimax algorithm. Basically, I start as a maximizer and apply depth first search until the state reaches certain depth or the state is a winning and Tie condition. After I reach the leaf node, I use my evaluation function to score that state, and recursively back-track whist saving the maximizing value and visit remaining nodes.

Optimization:

I have also applied alpha-beta pruning to my minimax algorithm. Initially values of alpha and delta are -infinity and infinity. When I do DFS, I track the values of alpha and beta, and whenever the alpha >= beta, I stop searching that branch.

On top of that instead of searching all the available moves for some state, I only search for relevant moves, i.e. moves near the pieces. Specifically, I have chosen the moves that have some piece – black or white – present near the distance of two. This has helped optimized my algorithm.

Challenges:

Coming with a good heuristic was a huge problem. So, I had to read several research papers on Gomoku to figure to a good heuristic for my AI.

Weakness:

I still cannot figure out why my AI does not outperform the baseline while I increase my depth which is very counterintuitive. However, when the depth is 2, my AI easily beats the baseline.