CSE 537 Assignment 2 Game Search

Atul Jha 110350053
atjha@cs.stonybrook.edu

Gerrard Lukacs 108166646
gerrard.lukacs@stonybrook.edu

1. Minimax search has been implemented in `basicplayer.py`. Since Connect Four is a zero sum game, negamax has been used to implement the search. In negamax the evaluation for min nodes is the negative of what it would be for max node.

2. The new evaluation function has been implemented in `basicplayer.py`. It scores each player based on the quality of their potential chains. A potential chain is a chain containing only empty cells and that player's tokens. Potential chain finding has been implemented by modifying `board.chain_cells` and related functions in `connectfour.py`. The quality of a potential chain is determined by the number of actual tokens in it, as well as how it would fare if the opponent fills an empty cell. It gives more significance to tokens in surviving sub-chains. Chain values are also squared, as a single strong chain is more valuable than many weak chains. Note that potential chains smaller than k (4 in Connect Four) are ignored.

3. Alpha-Beta search has been implemented in `basicplayer.py`. It uses the same concept of negamax. But here we drop those subtrees were alpha >= beta. We expect that expanded nodes should be less than minimax.

4.
    a) By introducing one more variable to the constructor of `ConnectFourBoard` object called `connect_k`, we were able to extend the game from connect four to connect k. This required some changes in `new_evaluation` function, clone function, `do_move` function and `is_win_from_cell` function.
    b) We introduced one more variable called `longest_streak` in the constructor of `ConnectFourBoard` to change the winning rule to whoever has the longest streak at the end of the game.

Results:

New Player vs Basic Player
Execution Time: 43.3247
Nodes Expanded: 3096

Alphabeta Player vs Basic Player
Execution Time: 22.5471
Nodes Expanded: 1435