TURTLE JUMP

ARTIFICIAL INTELLIGENCE PROJECT

Akshay Agrawal (B15CS004)

Aditya Agrawal (B15CS002)

TURTLE JUMP

PROBLEM STATEMENT

Turtles are at war, however they are too nice to kill, and can only convince each other to switch sides. Convincing is a special skill, and in case of turtles, all it takes to convince is a jump. If a turtle can jump over another turtle the, the one jumped over, is convinced to switch sides. Each player as a set of turtles lined up along the opposite sides of a field. The turtles can march in any direction along the lines. They can also jump over turtles of the other color, and thereby enslave them. Turtles cannot jump over turtles who are in the same side. The game ends, when one player has
a) No turtles left or b) has no moves left for the remaining turtles

SOFTWARE DESCRIPTION

Software is designed using pyGame library in Python. A full UI is implemented for better user experience. Both min-max and alpha-beta pruning is implemented for move selection along with proper heuristics.

FEATURES

1. A single player game (Human vs CPU).
2. Game is being played in 9X8 board, with each player having 16 turtles (pieces) with him arranged in 2 rows.
3. Human's turtle are of Red color while CPU turtle are assigned Green color.
4. **Each turtle can move in any of the 8 direction [BONUS Feature].**

Heuristic 1: First heuristic chosen was pretty simple, we just counted the number of turtles of each player and took the difference of them.
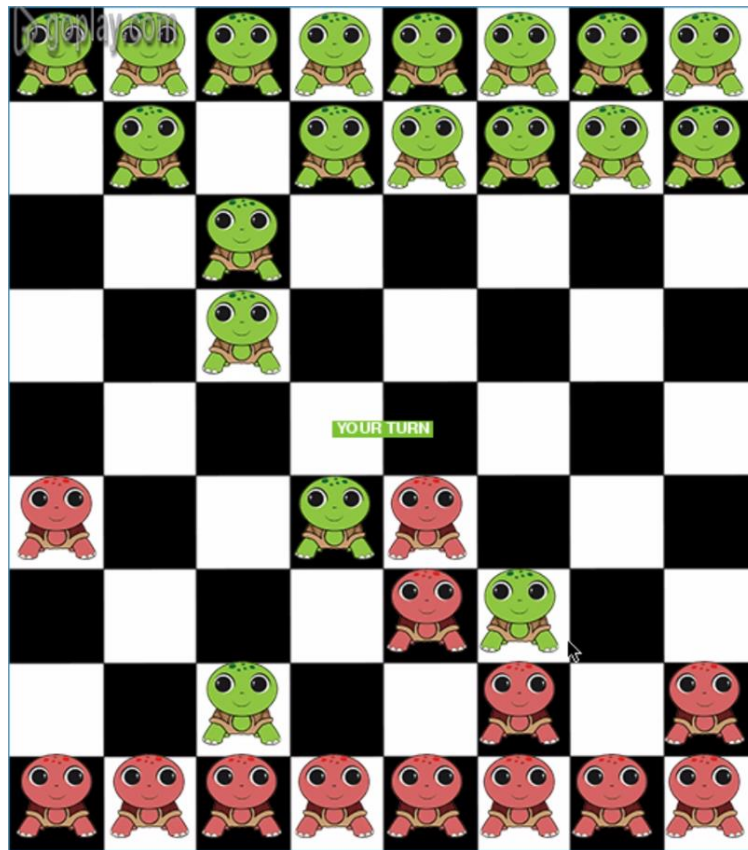
Explanation: In the above image, count (green) =19, count (red) = 13, difference (green – red) = 6, the value for each count is assigned as 100. If player having red turtle is evaluating heuristic he's in a weak position, so his heuristic value will be -600(6 X -100) whereas for opponent it'll be +600 indicating strong position.

Limitation: When both player will have equal number of turtles left, this heuristic will say 0 to both players irrespective of configuration of the pieces which might be in favor of some for next turn. So our next heuristic was based on this limitation.

Heuristic 2: Learning from our previous limitation, we implemented heuristic based on turtle's configuration. This heuristic will also take into account the opponent's turtles position. It checked if an opponent's turtle is adjacent to it in any of the 8 directions, if it is then that board state will lead to capture of that turtle by the opponent in next move. So that state is considered of less importance.
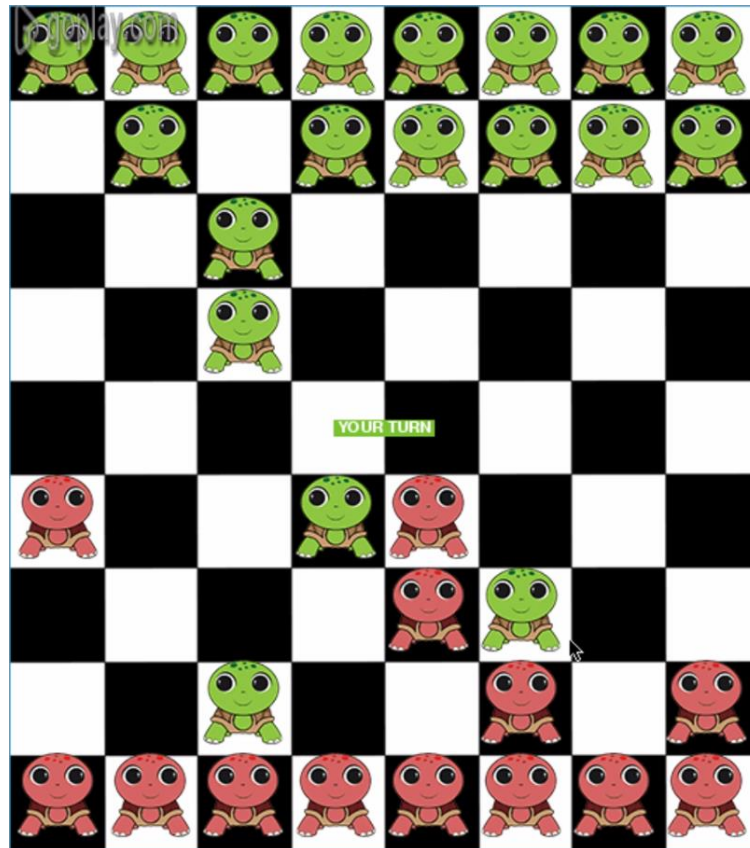
Explanation: In the above image, Row 3 Column 4 (Top-Left is Row0, Column0) turtle (green) is surrounded by a red turtle from right side and so it is in danger of being captured by it in next move. So, for every such configuration a value 10 is subtracted. Finally, this value for both players is calculated and subtracted to obtain final heuristic of the required player. (For red turtle player- Value (red)-Value (green)).

Heuristic 3: For better results we took the linear combination of the above 2 heuristics.

GAME PLAY:

Open this link to view gameplay.

GAME ANALYSIS:

We played 10 games for each mode (random move and precise move)

Here's the statistics for both:

1. CPU vs HUMAN (Random moves):   Game played = 10, CPU won = 8, Game Draw* = 2
   Since human is playing random moves, he's not playing his best move so probability of winning is more for CPU.
2. CPU vs HUMAN (Precise moves):   Game played = 10, CPU won = 4, Game Draw* = 1
   Since human is playing precise moves, he's playing his best move so probability of winning is more for HUMAN.

   * Criteria for game draw is 150 moves.

SHORTCOMING DISCUSSION/ LOSS ANALYSIS:

Various shortcomings are:

1. The ply depth in the above example is 2. Increasing the ply depth results in higher CPU move time.
2. There is no check for **horizon effect** in our heuristic.
3. No **lookup table** for starting and ending moves.
4. Further **quiescence search** might be required for board positions which are quiescent.