Othello

Team members:

Guohao Yan   yanguo
Gurleen Kaur

Roles and responsibilities:

Guohao Yan:
Designed the structure of project, defined classes and relationships.
Also encoded Othello.py which is the representation of Othello game,
And Algorithm.py which contains heuristic algorithms for search.
And writing of the report.

Gurleen Kaur:
Encoded Algorithm.py for minmax algorithms which is used in search engine.

Project type:

Game tree search

Project background:
Othello is a game that satisfy the properties of two-player, zero-sum, discrete, finite, deterministic, and perfect information. It fits perfectly for the course project.
The goal of our project is to determine by results what algorithms are best fit for competitive games like Othello. Our approach for the project is using game tree search, it makes nature sense that in a competitive game, each player wants to win, therefore game tree search is good fit as it minimize opponents profile while maximizing profile of its own.

Methods
After thoroughly analysing the game, we understood that we need two AI that plays against each other with different search algorithms to compare the results.
To achieve the goal, we first need to implement the game itself. (implemented in Othello.py)
Then, we need different heuristic algorithms that determines spot to place moves for AIs, for the algorithms, we had the following ideas:

1. Random move: Next move is picked randomly based on current board, as long as it doesn't violets game rules.
2. Most elimination move: Next move is picked where a single move can eliminate the most pieces of opponent, this is a dummy greedy heuristic algorithm.
3. Minmax move: Next move is picked by doing game tree search such that it will maximize the maximum player chances to win.

After having the algorithms, we would setup boards for AIs to play against each other and obtain the results from output to see if algorithm indeed provide better chances to win.

Evaluation and results
Limits
Conclusion
In this project, I've learned that alpha-beta cuts can improve efficiency of the game tree search by a lot, it limits the number of nodes that were generated and allowed us to explore deeper into the game tree to generate better results.
The part that I'd like to improve in this project is to do a GUI showing how the AIs are playing and the moves they made, because text representation wasn't intuitive to understand and we couldn't print the board every time a move is made, it slowed down the program a lot.

References

We used assignment 1’s implementation of search to create the search space and states to do the heuristic search.