UNIVERSIDAD
CATÓLICA
BOLIVIANA
COCHABAMBA

Departamento de Ingeniería y Ciencias Exactas
Carrera de Ingeniería en Sistemas

Assignment 3, Smart Othello

Sistemas Inteligentes

Rojas Marquez Cristhian Victor
Sanchez San Miguel Jose Manuel

Cochabamba- Bolivia
20 de Octubre de 2023

Description of the solution.

Since the point of this assignment is the implementation of the min max algorithm, with the alpha beta pruning and the alpha beta pruning with heuristics, we took a repository containing the logic of the game already. The link for the repository will be at the end of the document.

The game in this repository was coded in python, so we also made the algorithm in python, so there was no need for APIs.

At first we had to be sure that the algorithm was working right, we started coding the algorithm in a tic tac toe game, we started with min max, since it's the easiest one to implement; also in a tic tac toe game there aren't that many states to expand, so it was a good starting point. Once we made sure it was working correctly, we added the alpha beta pruning, and the heuristic, which was random so it wasn't very smart, but it worked.

The next step was combining our min max with the game, lucky for us, the game was already using a min max, we only had to switch theirs with ours. After a little debugging it worked perfectly.

We were still using the random heuristic which was very dumb, we needed to come up with something better. We played a lot of othello after this assignment was assigned, so we understood the basics of this game and we realized the most important positions in the game are the corners, since they are the only inmutable positions. So we definitely had to make sure the heuristic was taking this into account. The borders come in a close second because they are hard to change.

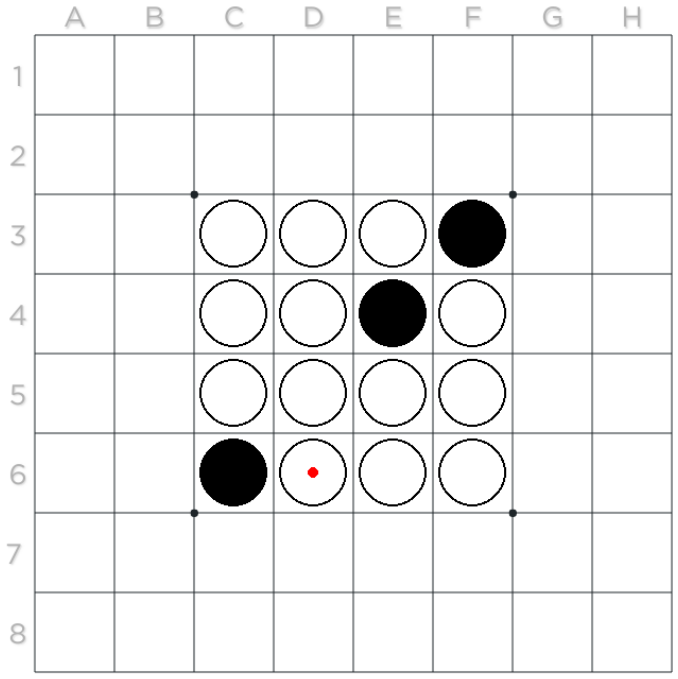
Our basic heuristic is just counting who is winning at the end of the turn, since the white is represented as "-1" and the blacks are represented as "1". All we have to do is add all the boards and we get our answer.

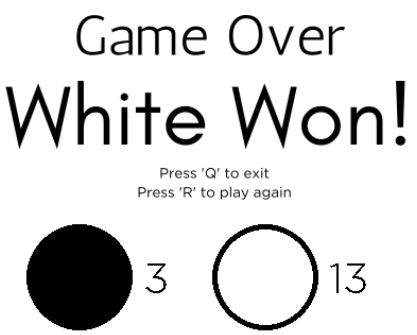
Our second and more intelligent heuristic is taking in consideration the corners and the borders, it adds the value of the borders and the corners, where each one has a different weight; the borders weigh 2 and the corners 4.

Experiments.

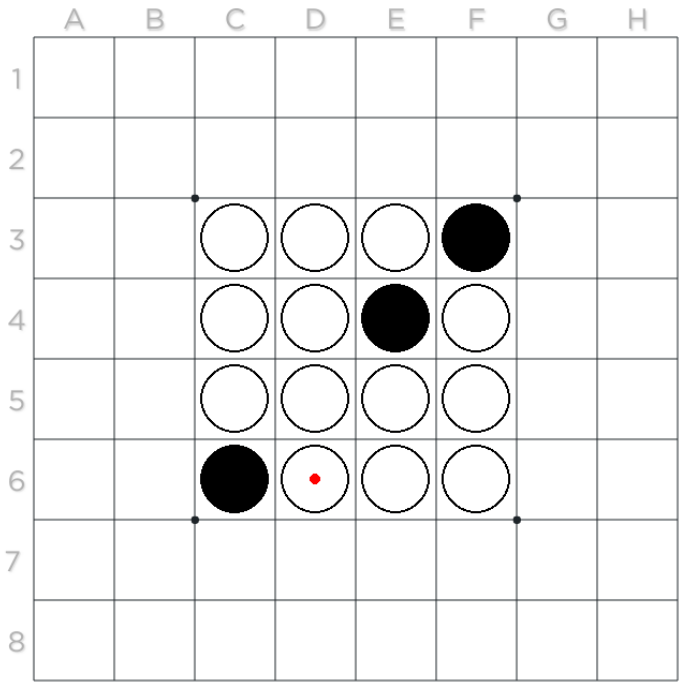
We have to see the difference between min max, min max alpha beta pruning and the heuristics. Before we started all the experiments, we made a fascinating discovery(it was probably discovered before), when 2 AIs "fight", if nothing changes in the algorithm(no randoms), the outcome will always be the same. We called this fate(it probably has another name, but we will keep it since it sounds cool). At first we thought that we should run a lot of games between different heuristics and different algorithms, but after the discovery of fate we realized that only a few experiments needed to be run.

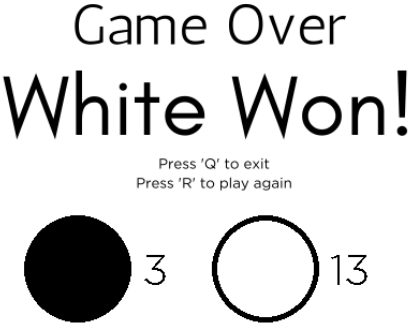
First we needed to watch the min max algorithm working, for this experiment will be using the min max that was already implemented in the repository. We also need to take in consideration that many of the other algorithms can only think as the white player, so we can't test both as a white and black, only as the black player. The code for this experiment is in the 4x4_board_min_max branch in the repository.





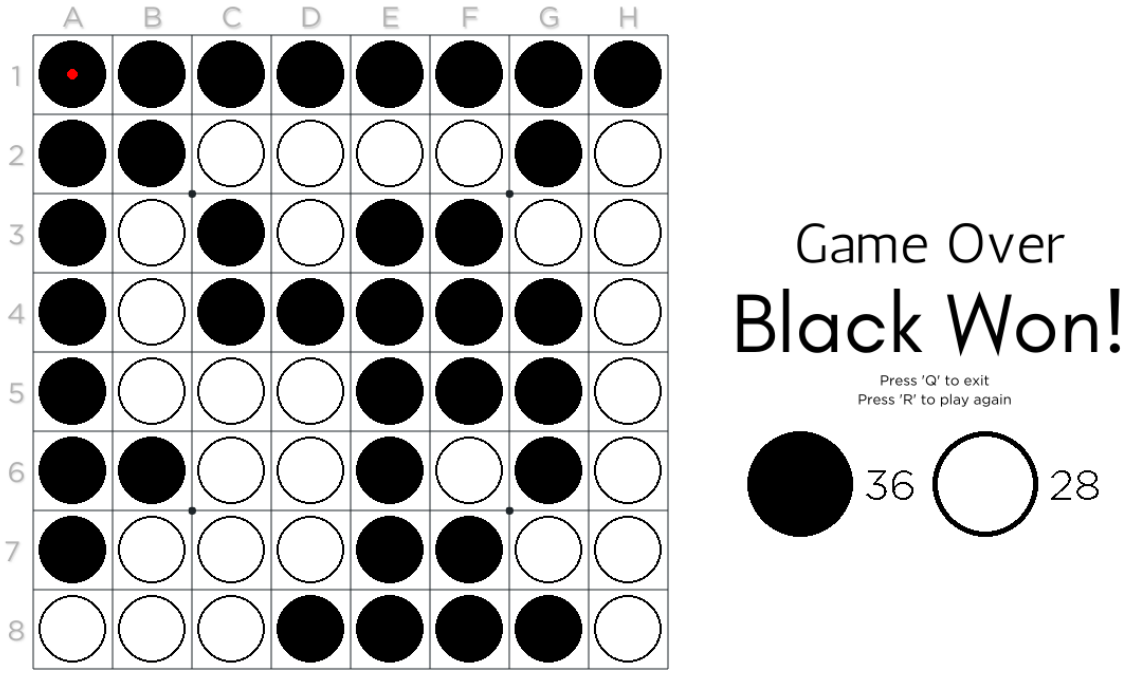
This is the fate of the board, surprisingly the min max(which expands all of the possible moves) gets destroyed by a min max with heuristic. This is likely to happen since the black player is expecting the best move from the white player, but the white player has a limited depth so we get a “so smart is dumb” scenario. We can try the same experiment with the min max alpha beta, which is in the branch 4x4_board_alpha_beta.



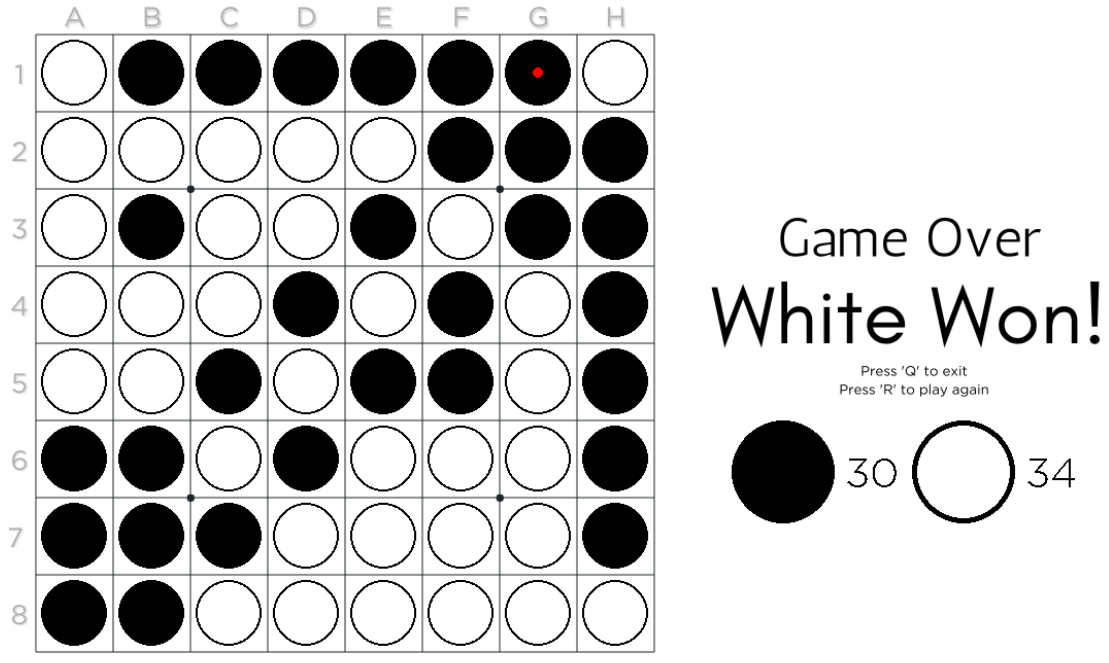


As we can see the fate of the game is the same, since it took the same choices, but the time it took to end the game was way less than the time that normal min max did, which is expected from the

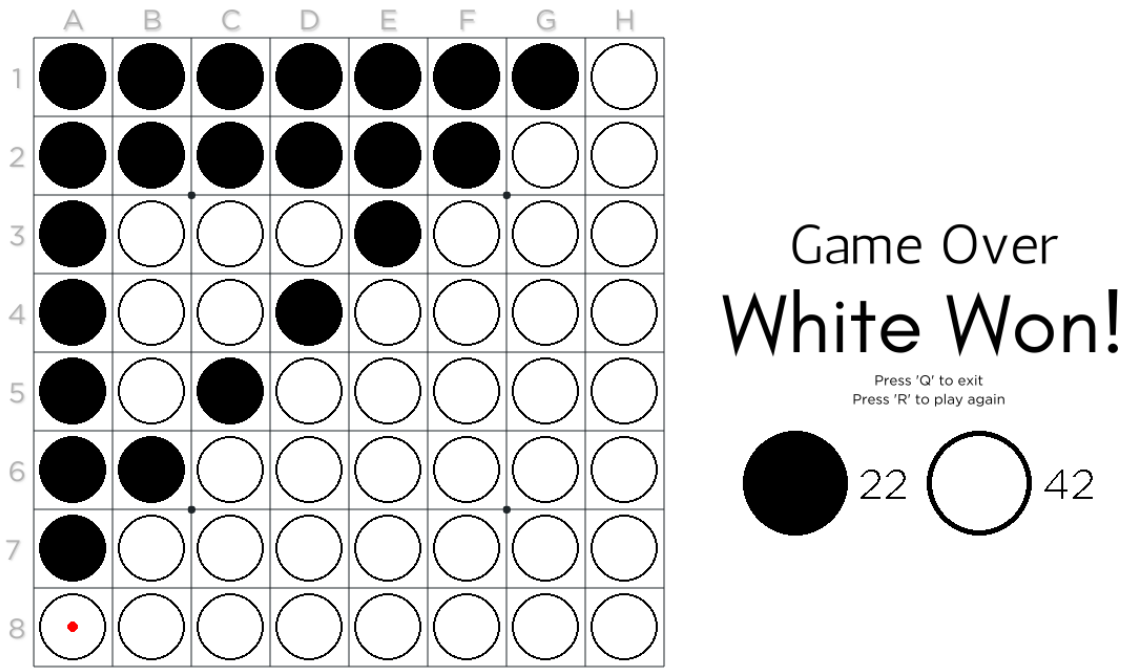
pruning algorithm since it cuts the branches that don't affect the outcome.(The normal min max almost killed Jose's computer, Cristian's worked fine)
Now we can test the fate of the min max with our best heuristic with a full board which is in the branch autoplay.



As we can observe, our algorithm won(we would leave a video showing how it won, but the fate will be the same, so you can watch the game live with our code). We also tried our best heuristic with the best AI we found on the internet, https://playpager.com/othello-reversi/ in the hard mode.



From the experiments we've run, the AI in that game has a random element since it doesn't obey fate. Now we can try the IA in the game with our basic heuristic.



Our basic heuristic is destroyed by the in game min max.
The in-game heuristic is a little more complicated, and we tried to avoid looking at it, since it felt like cheating. Our heuristic works well against AI, but it falls short against human players.

Conclusions

- The corners and borders heuristic works way better than the first heuristic.
- As long as there isn't a random in the algorithm, the fate of the game will always be the same.
- Min max and min max alpha beta pruning get the same result most of the time.
- Since min max and min max alpha beta expects the player to play his best moves, it can be easily outperformed.
- Min max alpha beta heuristic is the best one, and the only realistic one because in a full board the amount of states it has to expand is about: 10^{60}. It is heavily dependent on the heuristic, ours is simple but it works well enough.

Appendix

Original repository: https://github.com/TERNION-1121/Othello-Reversi-Game

Installation guide

The code requires python 3.10 +

Open a terminal in the main folder and use the command ‘pip install -r requirements.txt’.
Go to the folder scr and run the command python main.py.
To see the code for the experiments, switch the branch before downloading the code.