Othello

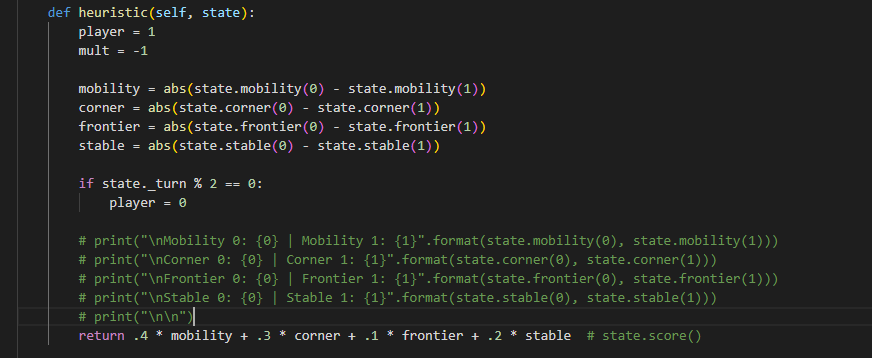
The goal of the Othello project was to create an agent capable of winning in the game of Othello against preset agents and other student agents. Some of the techniques to be considered during the several iterations of my *solution agent* included:

* Alpha-beta pruning
* Move ordering
* Game heuristics
* Best move memory

With the above in mind, I will proceed to elaborate the several iterations my *agent* went through.

**Version 1: Improve Alpha-Beta heuristics**

From the onset, my approach was to improve the heuristic function that calculated the *leaf* values of the *alpha-beta* tree. I wanted to maximize mobility, corners and stability while reducing frontiers. To do so, I came up with a simple formula showcased in the picture below.



To test my *agent*, I played it against the *alpha-beta* agent and below are the results after 5 games

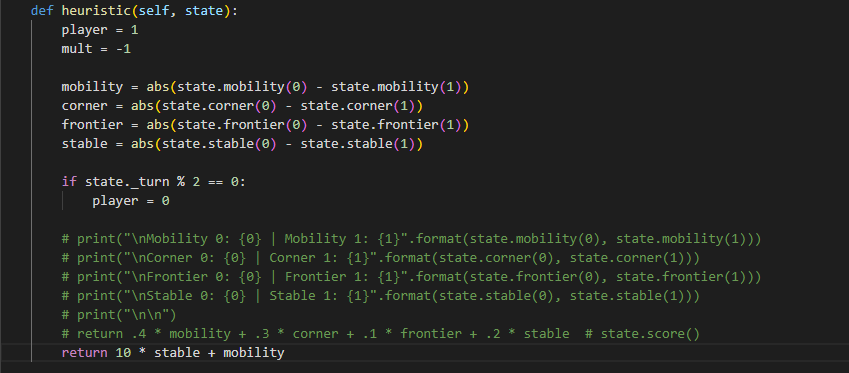
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Agent** | **1** | **2** | **3** | **4** | **5** |
| pokorie | L | W | W | L | W |
| Alpha-Beta | W | L | L | W | L |

My *agent* won 3/5 games, a 60% winning rate. I uploaded this version to the contest and my results against the preset agents were as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Elo Rating** | **Elo Rating vs Reference** | **Grade** | **Overall Record** | **Record vs Reference** | **Record vs Student** |
| Princewill Okorie | 1437.60 | 1550.38 | 65.53 | 45-53-2 (46.00%) | 26-22 (54.17%) | 19-31-2 (38.46%) |

**Version 2: Only consider stability and mobility**

Still on improving the heuristics, I decided to only consider stability and mobility as shown in the picture below.



In addition, I wrote a simple test suite to run my *agent* against the *alpha-beta* agent in a 10-series game challenge without visual. Each agent had 2 seconds to decide a move. The results were as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Agent** | **Wins** | **Losses** | **Sequence** |
| pokorie (B) | 6 | 4 | 'B', 'W', 'B', 'W', 'W', 'B', 'W', 'B', 'B', 'B' |
| Alpha-Beta (W) | 4 | 6 | 'B', 'W', 'B', 'W', 'W', 'B', 'W', 'B', 'B', 'B' |

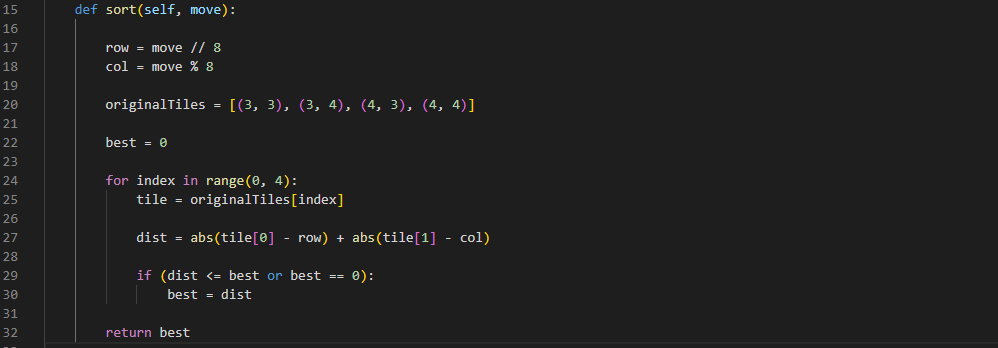
Still a 60% winning rate. I uploaded this version to the contest and my results against the preset agents were as follows:

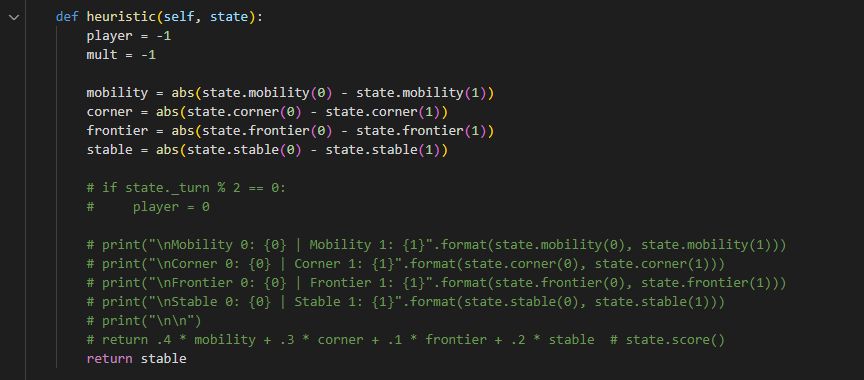
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Elo Rating** | **Elo Rating vs Reference** | **Grade** | **Overall Record** | **Record vs Reference** | **Record vs Student** |
| Princewill Okorie | 1527.90 | 1628.86 | 71.02 | 65-54-1 (54.58%) | 29-19 (60.42%) | 36-35-1 (50.69%) |

My *Elo* rating increased across the board. I did better against the reference agents and other students.

**Version 3: Add move ordering and only consider stability**

My next attempt was to introduce sorting to the list of *moves* explored before my *agent* ran out of time. The sort logic is pictured below:





The idea was to explore moves closer to the epicenter of the game first. I also considered just using stability as my heuristics. After testing this new version against the *alpha-beta* agent, the test results came back as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Agent** | **Wins** | **Losses** | **Sequence** |
| pokorie (B) | 8 | 1 | 'W', 'B', 'B', 'B', 'D', 'B', 'B', 'B', 'B', 'B' |
| Alpha-Beta (W) | 1 | 8 | 'W', 'B', 'B', 'B', 'D', 'B', 'B', 'B', 'B', 'B' |

A 20% increase in winning chances. Several tests revealed this increase in efficiency because of the addition of the move ordering. I uploaded this version to the contest and my results against the preset agents were as follows:

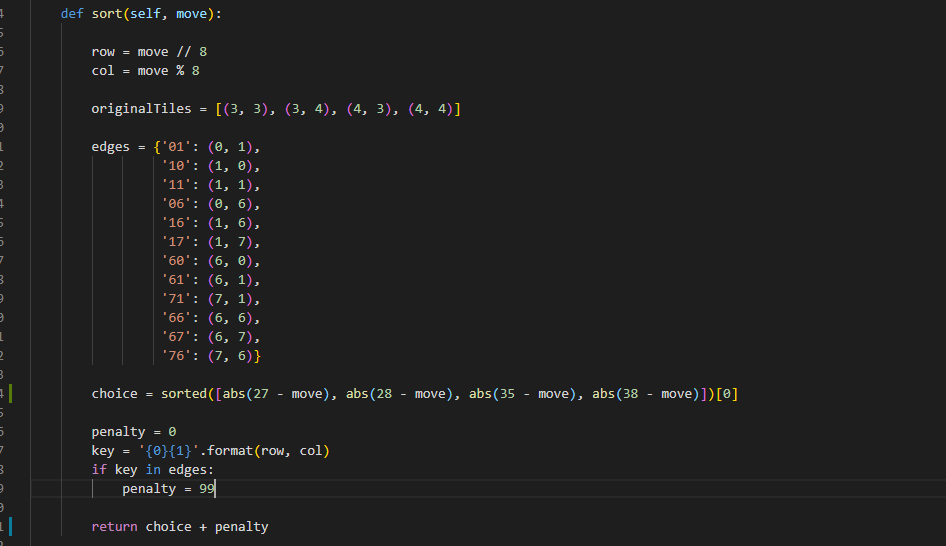
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Elo Rating** | **Elo Rating vs Reference** | **Grade** | **Overall Record** | **Record vs Reference** | **Record vs Student** |
| Princewill Okorie | 1727.81 | 1716.81 | 77.95 | 33-28-1 (54.03%) | 17-10-1 (62.50%) | 16-18 (47.06%) |

My *Elo* rating increased again across the board. I did better against the reference agents, but worse against other students.

**Version 4 and so on:**

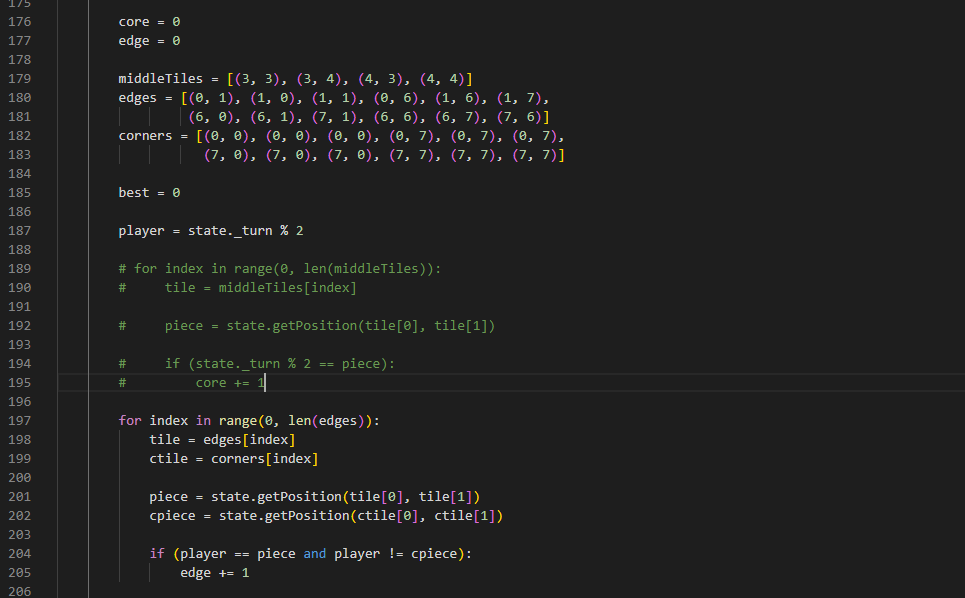
Unfortunately, I could not improve my agent further. I tried several techniques as depicted in my source code and eventually ran out of time. Some of the things I tried included the following:

1. **A different sorting algorithm: Add penalty for moves close to the corners**



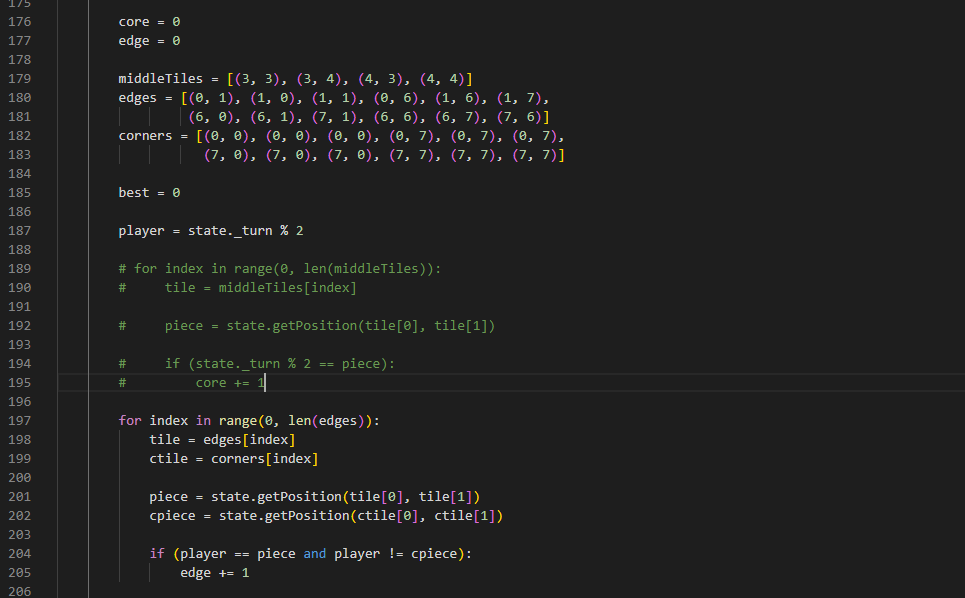
The idea here was to explore last moves that meant placing my *agent’s* disk in locations adjacent to the corners

1. **A different heuristic: Add penalty if piece is close to the corners**



The idea here was to introduce a heavy penalty, *edge*, to the heuristic value if a move caused my *agent’s* disc to be adjacent to the corners.

1. **A different heuristic: Reward moves that maintains disk superiority in the epicenter**



The idea here was to reward moves that meant maintaining my *agent’s* superiority, *corner,* in the 4 discs in the epicenter of the board.