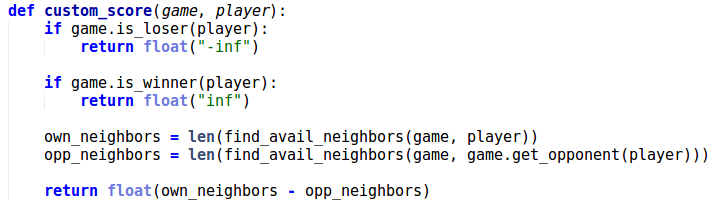
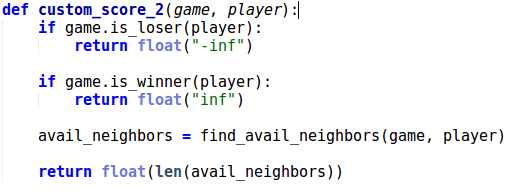
**Heuristic Analysis for Isolation Game**

**May 11, 2017**

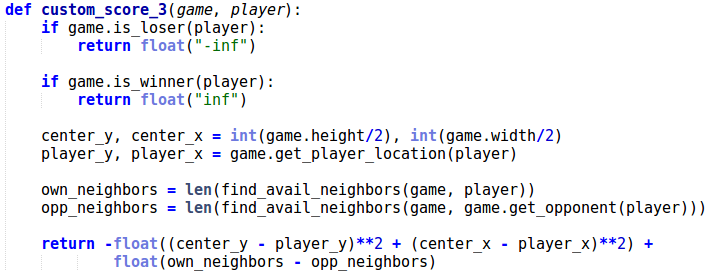
**1. ANALYSIS OF HEURISTIC FUNCTIONS**



Neighbors are defined as locations that are 1 row or 1 column away from the location of the specified player. **custom\_score** calculates the difference between the number of the given player's neighbors and that of their opponent's. The given player is incentivized to move to locations that are adjacent to as many unoccupied spaces as possible while minimizing their opponent's number of unoccupied neighbors. This is because if the given player's neighbors are all occupied, most likely they're trapped and soon they'll lose.



**custom\_score\_2** calculates the number of the given player's neighbors and that of their opponent's. The given player is incentivized to move to locations that are adjacent to as many unoccupied spaces as possible. However, unlike **custom\_score**, this function doesn’t take into account opponent’s neighbors. As a result, it is not as informative as the former. We can see in ***Table 1*** that **custom\_score\_2** performs worse than **custom\_score**.



The function considers both the distance of the player's location from the center and the difference between the number of the given player's neighbors and that of their opponent's. The player is incentivized to move to locations that are both close to the center and that are adjacent to as many unoccupied spaces as possible, while trying to minimize their opponent's number of unoccupied neighbors. However, interestingly, **custom\_score\_3** performs slightly worse than **custom\_score**. This could be because it takes more time to compute **custom\_score\_3** and iterative deepening using this function would be slightly less likely to search as deeply as iterative deepening with **custom\_score** does.

***Table 1: Results from running 20 matches against each opponents***

**Match # Opponent AB\_Improved AB\_Custom AB\_Custom\_2 AB\_Custom\_3**

Won | Lost Won | Lost Won | Lost Won | Lost

1 Random 31 | 9 35 | 5 33 | 7 36 | 4

2 MM\_Open 19 | 21 27 | 13 25 | 15 22 | 18

3 MM\_Center 25 | 15 28 | 12 25 | 15 26 | 14

4 MM\_Improved 21 | 19 20 | 20 19 | 21 22 | 18

5 AB\_Open 21 | 19 19 | 21 20 | 20 20 | 20

6 AB\_Center 22 | 18 20 | 20 17 | 23 20 | 20

7 AB\_Improved 18 | 22 20 | 20 18 | 22 20 | 20

----------------------------------------------------------------------------------------------------------------------

**Win Rate:**  56.1% 60.4% 56.1% 59.3%

**RECOMMENDATIONS:** I recommend using **custom\_score** for several reasons:

* It takes into account information about both the given player and their opponent, whereas **custom\_score\_2** only has information about the given player.
* It is slightly less complicated than **custom\_score\_3**. As the board size grows and/or the allowed search time shrinks, I believe it’s important to have a simpler heuristic function so that we can search more deeply.
* Moreover, as shown in Section 2 below, **get\_move** has already ensured that the given player occupies the center square if vacant. This is equivalent to the minimum value of the squared distance to the center, which is the best case scenario for the first term of **custom\_score\_3** that calculates the squared distance to the center.

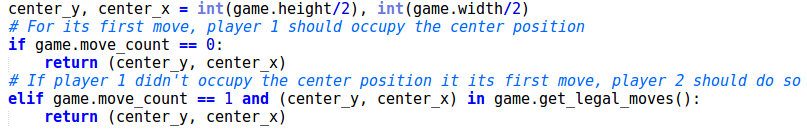
**2. ANALYSIS OF ALPHABETAPLAYER’S GET\_MOVE FUNCTION**

In addition to the required implementation of iterative deepening for **get\_move** in the AlphaBetaPlayer class, I have added a few features to enhance an AlphaBetaPlayer’s performance. Although the features are not in **custom\_score**, they play an important role in the player’s performance. Therefore, it’s worth mentioning them here.

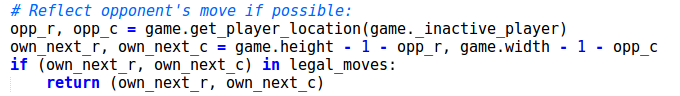
a) If there is only one legal move available to a player, they should choose it, and avoid searching more deeply down the game tree, which is much more time consuming.



b) A player should occupy the center position if available, right from the beginning of the game.

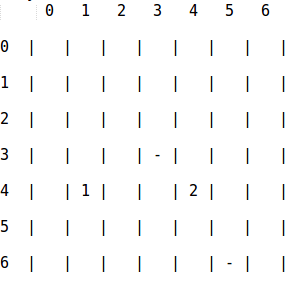


c) As mentioned in the lecture, a player should reflect their opponent's move if possible. See ***Figure 1*** for an example.



***Figure 1 – Example of player 1 reflecting player 2's move***

It’s player’s 1 turn to move. The game state before player 1 moves:



The game state after player 1 moves to (2, 2), reflecting opponent's position at (4, 4):

