Ziyan Feng February 27, 2017

Heuristic Analysis

For this project of building an Isolation game, three heuristic functions were implemented and tested.

Heuristic Function 0 - calculates **number of my moves left**. This is an intuitively straight forward evaluation function, where the higher the number of my moves left, the better the chance that I will win.

```
def heuristic0(game, player):

    my_moves = len(game.get_legal_moves(player))
    score = game.utility(player) + float(my_moves)

    return score
```

Heuristic Function 1 - calculates (**number of my moves left - number of the opponent's moves left**). This evaluation function tries to encourage the advantage of me against my opponent, i.e. the more moves left for me than for my opponent, the better the chance I will win.

```
def heuristic1(game, player):

    my_moves = len(game.get_legal_moves(player))
    opponent_moves = len(game.get_legal_moves(game.get_opponent(player)))

    score = game.utility(player) + float(my_moves - opponent_moves)

    return score
```

Heuristic Function 3 - calculates (**number of my moves left - 2 * number of the opponent's moves left**). This evaluation further awards the advantages of me against my opponent with a stricter manner than Heuristic1, by giving the opponent moves a weight of 2.

```
def heuristic2(game, player):

    my_moves = len(game.get_legal_moves(player))
    opponent_moves = len(game.get_legal_moves(game.get_opponent(player)))

    score = game.utility(player) + float(my_moves - 2 * opponent_moves)

    return score
```

	Heuristic 0	Heuristic 1	Heuristic 2
ID_Improved	**66.43%**	64.29%	62.86%
Student	**66.43%**	**68.57%**	**70.00%**

As shown in the above table, Heuristic 0 draws the game with ID_Improved while Heuristic 1 and Heuristic 2 both beat the ID_Improved by marginal advantage.