

# Custom Score functions implented are:

```
def custom_score(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player.
    This should be the best heuristic function for your project submission.
    Note: this function should be called from within a Player instance as
    `self.score()` -- you should not need to call this function directly.
    Parameters
    -----
    game : `isolation.Board`
        An instance of `isolation.Board` encoding the current state of the
        game (e.g., player locations and blocked cells).
    player : object
        A player instance in the current game (i.e., an object corresponding to
        one of the player objects `game.__player_1__` or `game.__player_2__`.)
    Returns
    -----
    float
        The heuristic value of the current game state to the specified player.
    """
    if game.is_loser(player):
        return float("-inf")
    if game.is_winner(player):
        return float("inf")
    moves_player = len(game.get_legal_moves(player))
    moves_rival = len(game.get_legal_moves(game.get_opponent(player)))
    return float(moves_player/2-2*moves_rival)

def custom_score_2(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player.
    Note: this function should be called from within a Player instance as
    `self.score()` -- you should not need to call this function directly.
    Parameters
    -----
    game : `isolation.Board`
        An instance of `isolation.Board` encoding the current state of the
        game (e.g., player locations and blocked cells).
    player : object
        A player instance in the current game (i.e., an object corresponding to
        one of the player objects `game.__player_1__` or `game.__player_2__`.)
    Returns
    -----
    float
        The heuristic value of the current game state to the specified player.
    """
    if game.is_loser(player):
        return float("-inf")
    if game.is_winner(player):
        return float("inf")
    moves_player = len(game.get_legal_moves(player))
    moves_rival = len(game.get_legal_moves(game.get_opponent(player)))
    return float(-(moves_rival)**2)

def custom_score_3(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player.
    Note: this function should be called from within a Player instance as
    `self.score()` -- you should not need to call this function directly.
    Parameters
    -----
```

```

game : `isolation.Board`
    An instance of `isolation.Board` encoding the current state of the
    game (e.g., player locations and blocked cells).
player : object
    A player instance in the current game (i.e., an object corresponding to
    one of the player objects `game.__player_1__` or `game.__player_2__`.)
Returns
-----
float
    The heuristic value of the current game state to the specified player.
"""
if game.is_loser(player):
    return float("-inf")
if game.is_winner(player):
    return float("inf")
moves_player = len(game.get_legal_moves(player))
moves_rival = len(game.get_legal_moves(game.get_opponent(player)))
return float(moves_player-2*moves_rival)

```

custom score and custom score3 are similar to improved score. Where custom score3 is trying to increase moves of given player and reduce 2 fold moves of rival , but in custom score we are increasing moves of given player by 2 fold and reduce rivals moves by 2 fold.

In custom score2 we are trying to decrease rivals moves exponentially

This script evaluates the performance of the custom\_score evaluation function against a baseline agent using alpha-beta search and iterative deepening (ID) called `AB\_Improved`. The three `AB\_Custom` agents use ID and alpha-beta search with the custom\_score functions defined in game\_agent.py.

*****									
Playing Matches									
*****									
Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	8	2	10	0	10	0	9	1
2	MM_Open	6	4	9	1	6	4	6	4
3	MM_Center	8	2	7	3	9	1	7	3
4	MM_Improved	6	4	5	5	3	7	3	7
5	AB_Open	2	8	6	4	6	4	4	6
6	AB_Center	5	5	6	4	5	5	5	5
7	AB_Improved	6	4	6	4	8	2	5	5
-----									
Win Rate:		58.6%		70.0%		67.1%		55.7%	

The win rate from the tournament results are: AB custom > AB custom2> AB improved > AB custom3

I choose AB custom because:

It has got the highest winning rate of 70% The winning factor for AB custom is dependent on increase of ones moves and decrease of the rivasl moves, by 2 fold helped to get 70% of win rate.