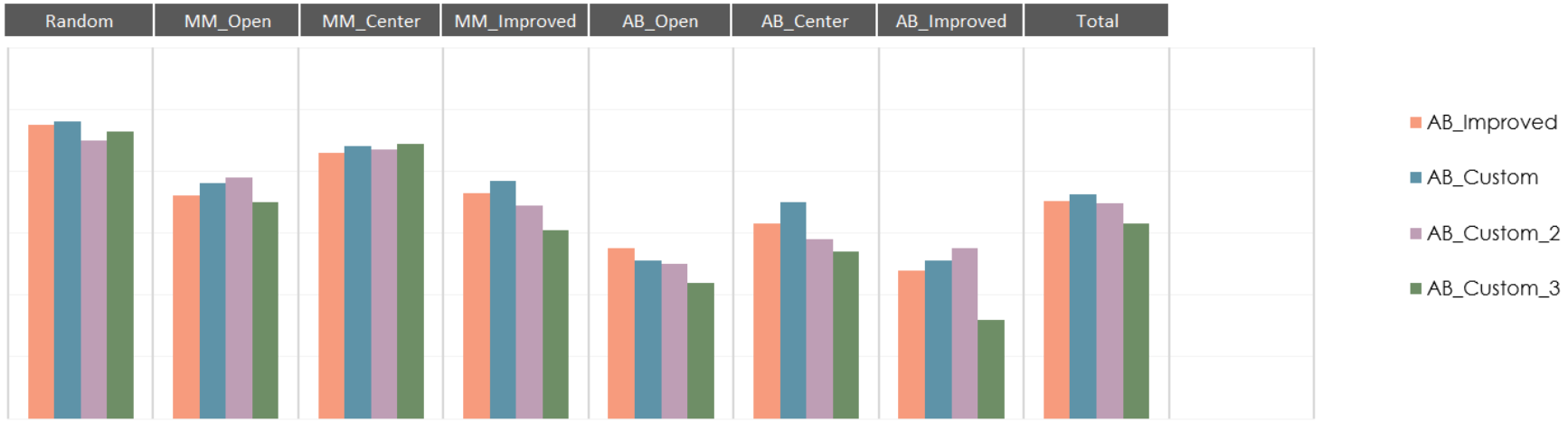
# Tournament Result




| Opp. Player | Random | MM_Open | MM_Cente | MM_Impro | AB_Open | AB_Center | AB_Improv | Total | Average |
|---|---|---|---|---|---|---|---|---|---|
| AB_Improved | 95.00 | 72.00 | 86.00 | 73.00 | 55.00 | 63.00 | 48.00 | 70.29 | |
| AB_Custom | 96.00 | 76.00 | 88.00 | 77.00 | 51.00 | 70.00 | 51.00 | 72.71 | |
| AB_Custom_2 | 90.00 | 78.00 | 87.00 | 69.00 | 50.00 | 58.00 | 55.00 | 69.57 | |
| AB_Custom_3 | 93.00 | 70.00 | 89.00 | 61.00 | 44.00 | 54.00 | 32.00 | 63.29 | |
| **Average** | **93.50** | **74.00** | **87.50** | **70.00** | **50.00** | **61.25** | **46.50** | | |

```
                    **************************
                         Playing Matches
                    **************************

 Match #    Opponent    AB_Improved    AB_Custom    AB_Custom_2  AB_Custom_3
                         Won  | Lost    Won  | Lost    Won  | Lost   Won  | Lost
    1         Random      95  |   5     96  |   4     90  |  10    93  |   7
    2         MM_Open     72  |  28     76  |  24     78  |  22    70  |  30
    3        MM_Center    86  |  14     88  |  12     87  |  13    89  |  11
    4      MM_Improved    73  |  27     77  |  23     69  |  31    61  |  39
    5         AB_Open     55  |  45     51  |  49     50  |  50    44  |  56
    6        AB_Center    63  |  37     70  |  30     58  |  42    54  |  46
    7      AB_Improved    48  |  52     51  |  49     55  |  45    32  |  68
-----------------------------------------------------------------------------
           Win Rate:      70.3%        72.7%        69.6%        63.3%
```

## 1. Custom_score code details

Basic idea: Difference of moves between active and inactive players.

This can be done by using

- ⇨ active_player_moves – inactive_player_moves

Above formula has disadvantage of giving same score between 6 & 5 and 2 & 1

So, I changed formula to percentage of difference then

- ⇨ 5 to 6 percentage increase: 25%, but between 1 to 2: 100%

Worked much better but I got another disadvantage of having same score between 6 & 3 and 2 & 1

Now I go with formula of multiplying percentage with

- ⇨ (100-min(my_moves,opponent_moves)) * percentage

## 2. Custom_score2 code details

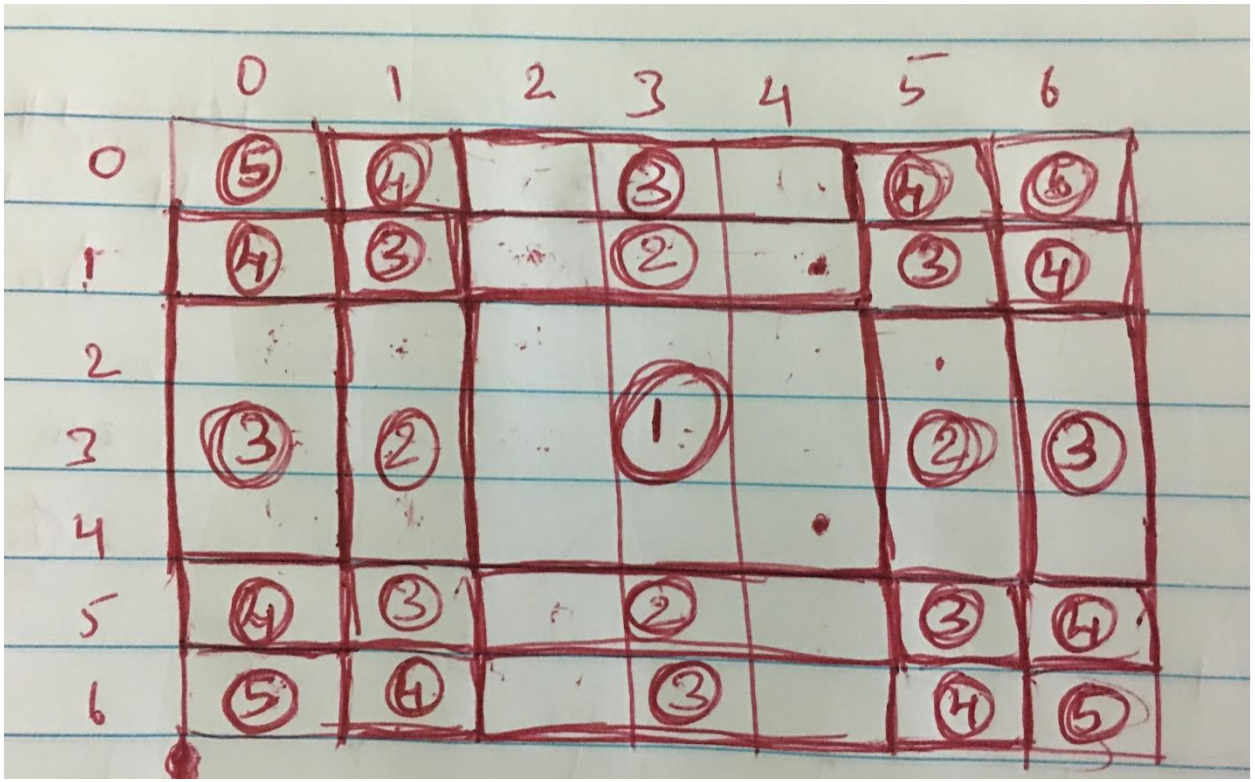
Basic Idea: Random moves. By giving some value to random move for starting moves at least twice the number of height and then based difference of moves between active and inactive players.

Calculate total number of moves each box is having in empty board and use that score by ignoring active and inactive player moves.

NOTE: This works much better inside AB as it doesn't with any move NO need to call a function.

I have placed in this function to accommodate your template.

To explain technically please see below image



In above image numbers inside boxes represent ranks.

| Ranks | Possible Moves |
|---|---|
| 1 | 8 |
| 2 | 6 |
| 3 | 4 |
| 4 | 3 |
| 5 | 2 |

This is what implemented but unless it is first move you have to go from one of possible moves that reduces possible moves by one

## 3. Custom_score3 code details

Basic Idea: Change logic as game progresses and use different logic if both players can't reach each other.

Progress normally means give score based on number of moves available to active player.

If active and inactive players can't reach each other in possible moves then give more score to less number of possibilities so that you will get more moves rather than wasting moves.

NOTE: Apart from all mentioned details each of the above three functions include below logic.

Opponent has only one move and that move is one of active player possible moves then give highest possible score below +inf

Active player has only one move and that move one of opponent possible moves then give lowest possible score above -inf

**My choice:**

Custom_Score => The reason for choosing this one over remaining two are

1. Much better in logic
2. Gives appropriate score based not only possibilities of active player but also on possibilities of opponent player as well.
3. This function gives much higher score when active player is close to win and much lower same is close to lose.

Custom_Score_2 => I believe this should be implemented as a dictionary inside game class.

Ex: {8: [(2,2), (2,3), (2,4) ...,6: [(1,2), (1,3), (1,4)]}

This should work very well if this logic is implemented inside the program as a default instead of using random move.

Custom_Score_3 => This is more about testing process of using different logic at different stages of game.