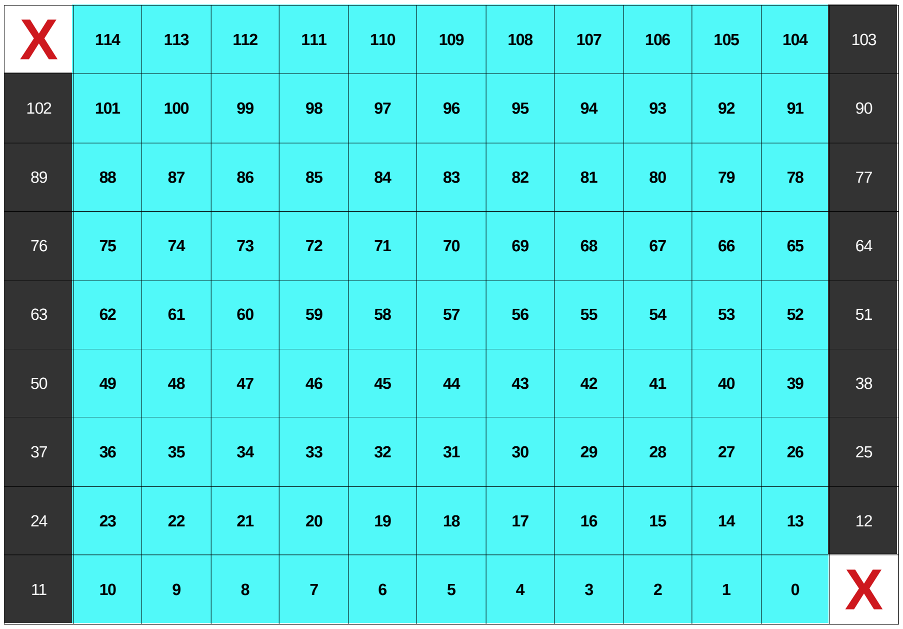
Experiments Report for Project 3
Project: Adversarial Search

Student Name: Ashish Sarode
Nano-Degree name: Artificial Intelligence Nanodegree.

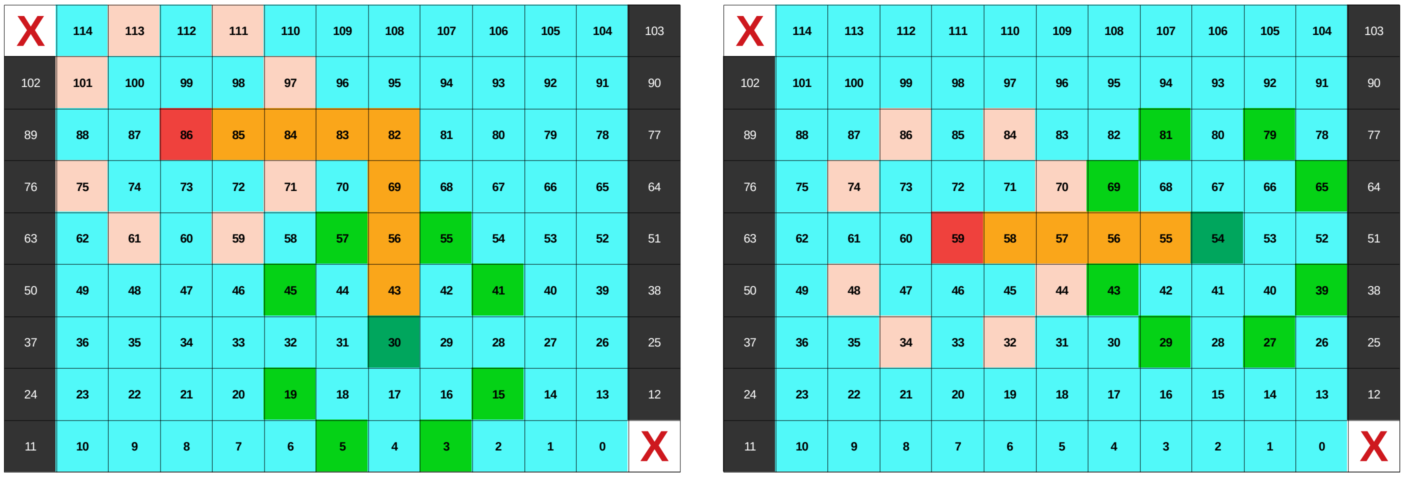
Initial Study:
1. To develop understanding of how good heuristics can be developed for this game, first thing was to get good understanding of the board.

A) Understanding Positions (Figure A) -
Below image shows how indices are mapped on board

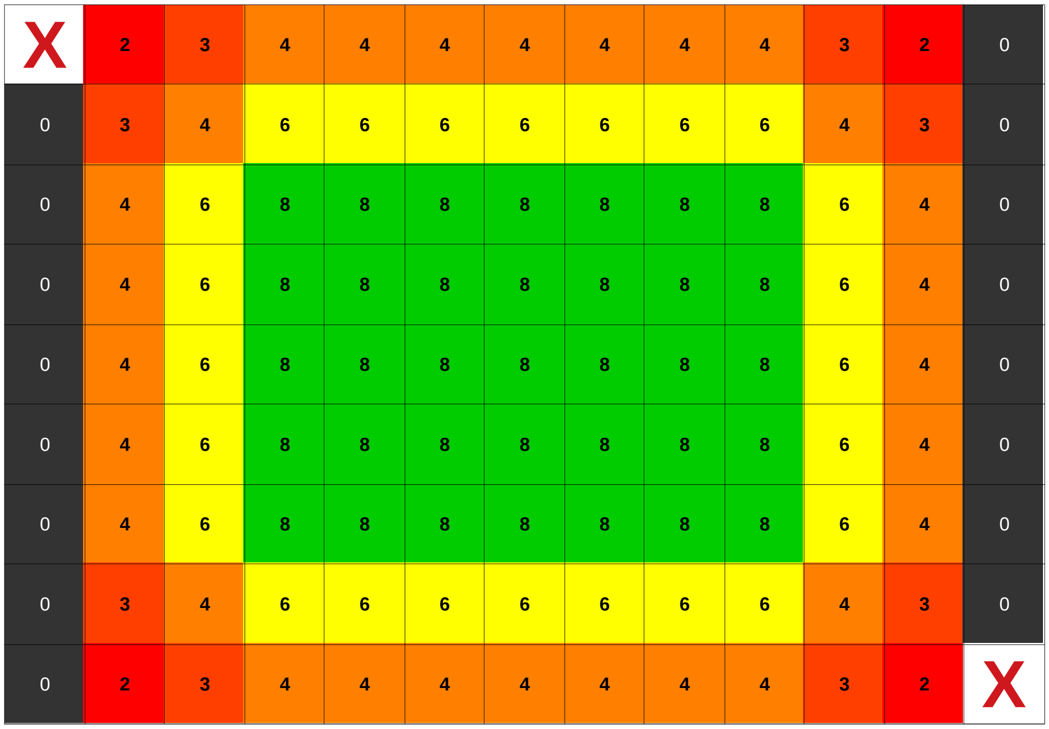


X	114	113	112	111	110	109	108	107	106	105	104	103
102	101	100	99	98	97	96	95	94	93	92	91	90
89	88	87	86	85	84	83	82	81	80	79	78	77
76	75	74	73	72	71	70	69	68	67	66	65	64
63	62	61	60	59	58	57	56	55	54	53	52	51
50	49	48	47	46	45	44	43	42	41	40	39	38
37	36	35	34	33	32	31	30	29	28	27	26	25
24	23	22	21	20	19	18	17	16	15	14	13	12
11	10	9	8	7	6	5	4	3	2	1	0	X

B) Understanding how players attack each other (Figure B)
Player 0 and Player 1 are at positions indicated by Red and Green Cells, figure shows their possible move positions and path in orange to indicate Manhattan distance between them -



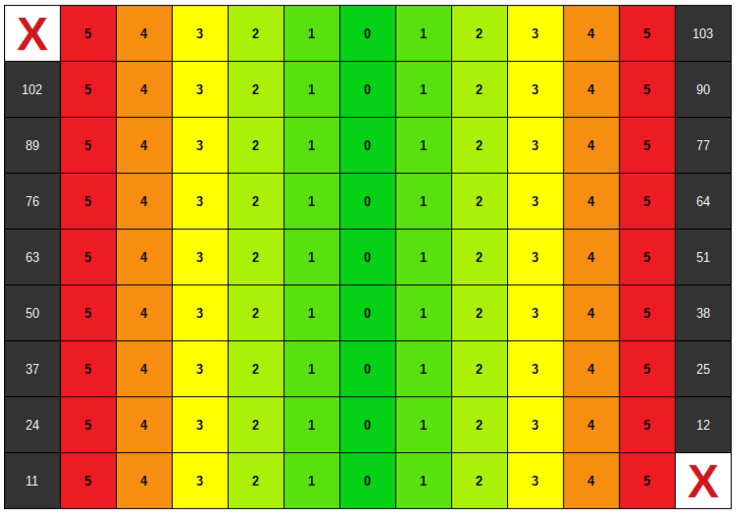
C) Understanding Liberties associated with each cell of the board (Figure C) -
This figure clearly shows that staying in the central area has benefit since it has more degrees of freedom -



X	2	3	4	4	4	4	4	4	4	3	2	0
0	3	4	6	6	6	6	6	6	6	4	3	0
0	4	6	8	8	8	8	8	8	8	6	4	0
0	4	6	8	8	8	8	8	8	8	6	4	0
0	4	6	8	8	8	8	8	8	8	6	4	0
0	4	6	8	8	8	8	8	8	8	6	4	0
0	4	6	8	8	8	8	8	8	8	6	4	0
0	3	4	6	6	6	6	6	6	6	4	3	0
0	2	3	4	4	4	4	4	4	4	3	2	X

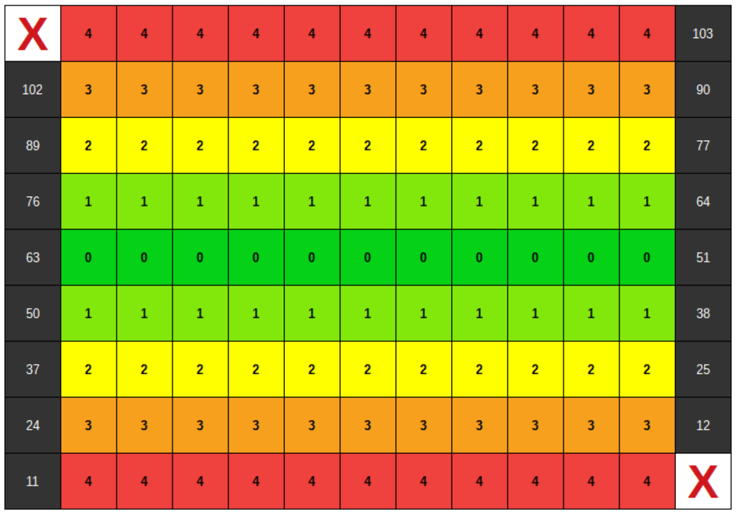
D) Distances from the central cell to different cells (Figure 5)

dx



X	5	4	3	2	1	0	1	2	3	4	5	103
102	5	4	3	2	1	0	1	2	3	4	5	90
89	5	4	3	2	1	0	1	2	3	4	5	77
76	5	4	3	2	1	0	1	2	3	4	5	64
63	5	4	3	2	1	0	1	2	3	4	5	51
50	5	4	3	2	1	0	1	2	3	4	5	38
37	5	4	3	2	1	0	1	2	3	4	5	25
24	5	4	3	2	1	0	1	2	3	4	5	12
11	5	4	3	2	1	0	1	2	3	4	5	X

dy



X	4	4	4	4	4	4	4	4	4	4	4	103
102	3	3	3	3	3	3	3	3	3	3	3	90
89	2	2	2	2	2	2	2	2	2	2	2	77
76	1	1	1	1	1	1	1	1	1	1	1	64
63	0	0	0	0	0	0	0	0	0	0	0	51
50	1	1	1	1	1	1	1	1	1	1	1	38
37	2	2	2	2	2	2	2	2	2	2	2	25
24	3	3	3	3	3	3	3	3	3	3	3	12
11	4	4	4	4	4	4	4	4	4	4	4	X

Custom Heuristic:

A) **Failed** Based on understanding from Figure B, developed 2 heuristics which work like - IntersectionLiberties = Common liberties between our player and opponent player.
Score = OwnLiberties – IntersectionLiberties
This means, we want to avoid positions which intersect with the opponent.
Result: Bad results
This heuristic did not perform well.

Tried other combinations like
Score = OwnLiberties – (Intersection Liberties + OpponentLiberties)/2
This and all other similar trials all gave bad results.

B) **Failed** Tried combinations based on distances based on understanding developed from figure C and D-
E.g. Distance from opponent, distance from center etc.
Did not get much success on it.
Result: Bad results

C) **** Winner** Penalty for based on distance from walls and corners based on understanding developed from figure C and D -

Score = BaseHeuristic – 0.25 for each second level cell from wall
Score = BaseHeuristic – 0.50 for each first level cell from wall
This means for wall corners we have -1.0 subtracted from base heuristic and so on.

Result: Good results

I have used iterative deepening along with the alpha-beta max search. This improves the base heuristic play itself to very good winning rates. Achieving additional improvements are difficult from here.

Without Iterative deepening			
Opponent	**Base Heuristic Win %**	**Custom Heuristic Win %**	**Improvement**
Random Player	92.5	92.5	**0**
Greedy Player	67.5	75	**7.5**
Minimax Player	46.2	55	**8.8**

With Iterative deepening			
Opponent	**Base Heuristic Win %**	**Custom Heuristic Win %**	**Improvement**
Random Player	97.5	97.5	**0**
Greedy Player	90	95	**5**
Minimax Player	73.8	77.5	**3.7**

Depth Effect on Winning %ge (for custom heuristic)			
Opponent	**Without iterative deepening**	**With iterative deepening**	**Improvement**
Random Player	92.5	97.5	**5**
Greedy Player	75	90	**15**
Minimax Player	55	73.8	**18.8**

Analysis of results for custom heuristics

1. For random player it is very difficult to achieve improvement since it is already at 97.5% but at least the results are as good as the base heuristic.
2. For Greedy player there is clear improvement.
3. For minimax player, I tried playing multiple times, the improvement was about 5% consistently.

Analysis of results compared to using iterative deepening

1. Searching more depth seems to give much better

What features of the game does your heuristic incorporate, and why do you think those features matter in evaluating states during search??

As explained previously, my heuristic gives more importance to staying in the area of maximum freedom and away from the minimum freedom.

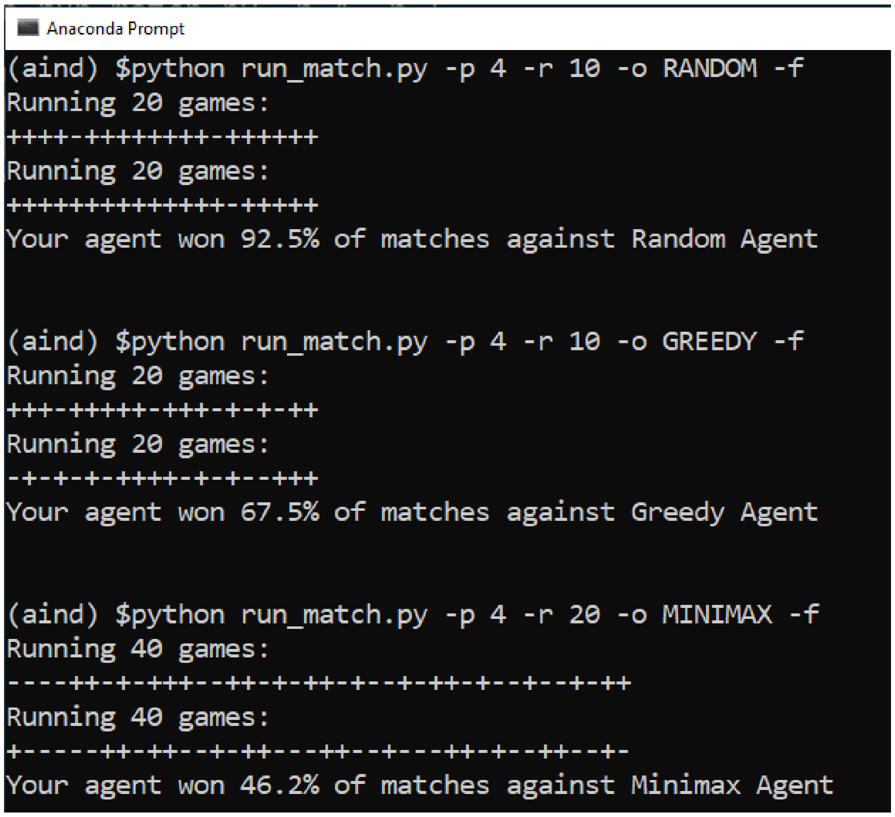
Please refer Figure C and D and explanation about heuristic C.

Analyze the search depth your agent achieves using your custom heuristic. Does search speed matter more or less than accuracy to the performance of your heuristic?

From my experiments, search speed does matter compared to heuristic accuracy. This is evident from the fact that when I tried without iterative deepening, the improvement due to accurate heuristic was much more in terms of percentage, but when iterative deepening is used (more search depth is used) it seems that the benefits of accurate heuristics are diminishing with higher depth of search.

Appendix:

1. Logs of Matches without iterative deepening, base heuristic

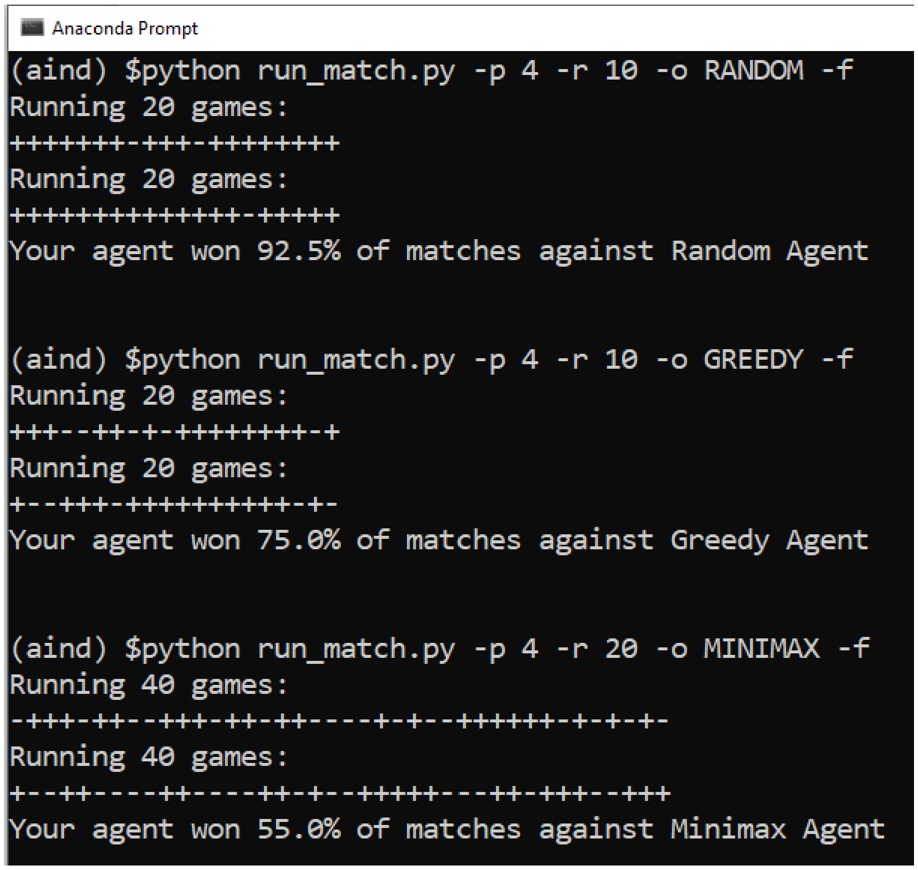


```
(aind) $python run_match.py -p 4 -r 10 -o RANDOM -f
Running 20 games:
++++-++++++++-++++++
Running 20 games:
++++++++++++++-+++++
Your agent won 92.5% of matches against Random Agent


(aind) $python run_match.py -p 4 -r 10 -o GREEDY -f
Running 20 games:
+++-+++++-+++-+-+-++
Running 20 games:
-+-+-+-++++-+-+--+++
Your agent won 67.5% of matches against Greedy Agent


(aind) $python run_match.py -p 4 -r 20 -o MINIMAX -f
Running 40 games:
----++-+-+++--++-+-++-+--+-++-+--+--+-++
Running 40 games:
+-----++-++--+-++---++-+---++-+--++--+-
Your agent won 46.2% of matches against Minimax Agent
```

2. Logs of Matches without iterative deepening, custom heuristic

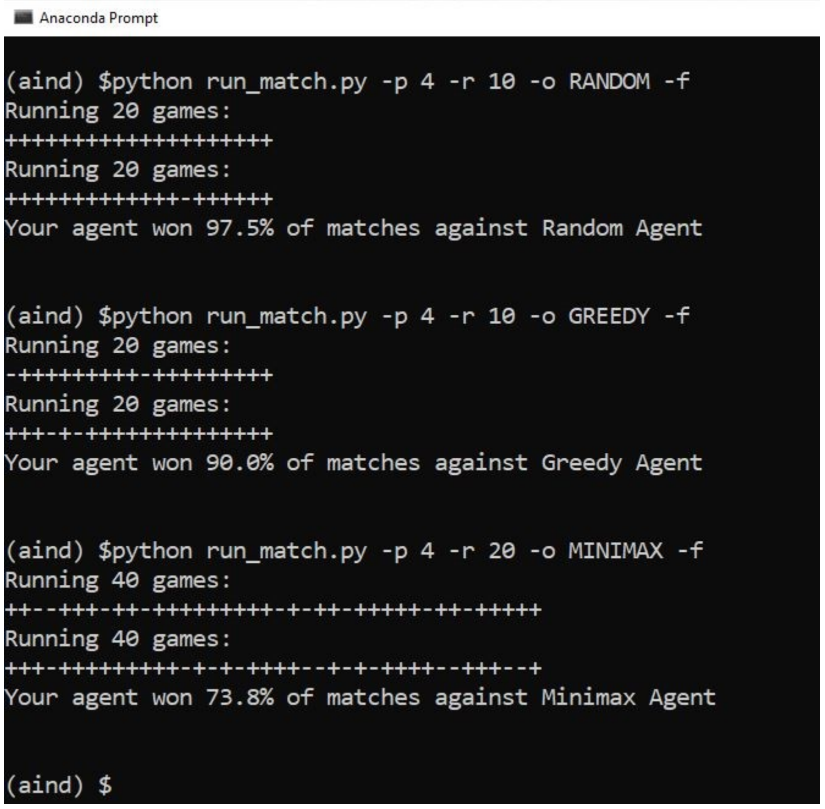


```
(aind) $python run_match.py -p 4 -r 10 -o RANDOM -f
Running 20 games:
+++++++-+++-++++++++
Running 20 games:
++++++++++++++-+++++
Your agent won 92.5% of matches against Random Agent


(aind) $python run_match.py -p 4 -r 10 -o GREEDY -f
Running 20 games:
+++--++-+-++++++++-+
Running 20 games:
+--+++-++++++++++-+-
Your agent won 75.0% of matches against Greedy Agent


(aind) $python run_match.py -p 4 -r 20 -o MINIMAX -f
Running 40 games:
-+++-++--+++-++-++----+-+--++++++-+-+-+-
Running 40 games:
+--++---++----++-+--+++++--++-+++--+++
Your agent won 55.0% of matches against Minimax Agent
```

3. Logs of Matches with iterative deepening, base heuristic



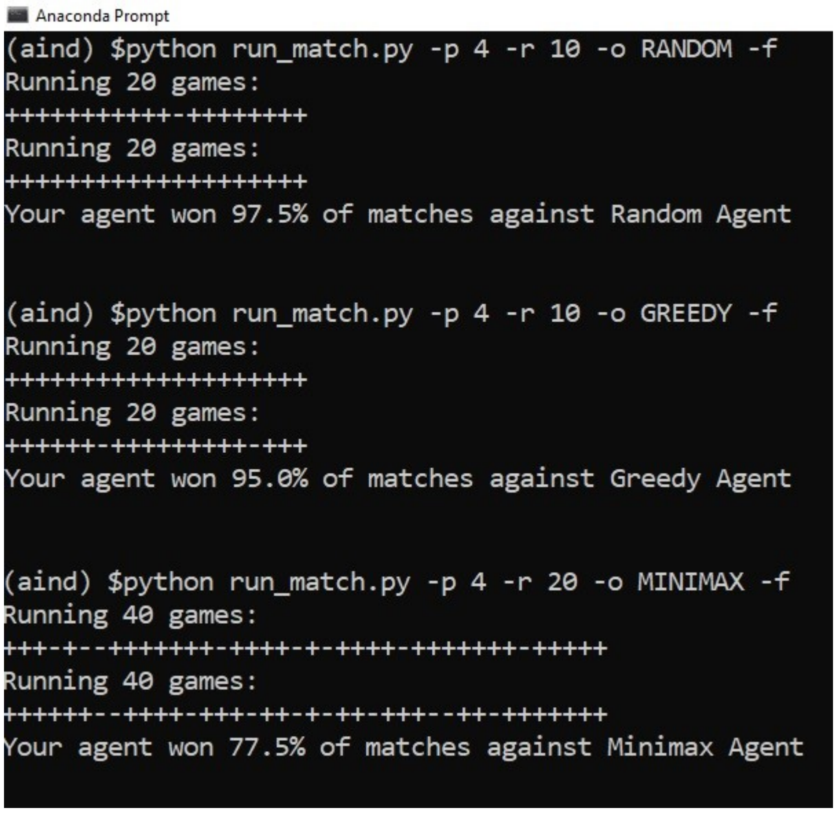
```
(aind) $python run_match.py -p 4 -r 10 -o RANDOM -f
Running 20 games:
++++++++++++++++++++
Running 20 games:
+++++++++++++-++++++
Your agent won 97.5% of matches against Random Agent


(aind) $python run_match.py -p 4 -r 10 -o GREEDY -f
Running 20 games:
-+++++++++-+++++++++
Running 20 games:
+++-+-++++++++++++++
Your agent won 90.0% of matches against Greedy Agent


(aind) $python run_match.py -p 4 -r 20 -o MINIMAX -f
Running 40 games:
++--+++-++-+++++++++-+-++-+++++-++-+++++
Running 40 games:
+++-+++++++++-+-+-++++--+-+-++++--+++--+
Your agent won 73.8% of matches against Minimax Agent


(aind) $
```

4. Logs of Matches with iterative deepening, custom heuristic



```
(aind) $python run_match.py -p 4 -r 10 -o RANDOM -f
Running 20 games:
+++++++++++-++++++++
Running 20 games:
++++++++++++++++++++
Your agent won 97.5% of matches against Random Agent


(aind) $python run_match.py -p 4 -r 10 -o GREEDY -f
Running 20 games:
++++++++++++++++++++
Running 20 games:
++++++-+++++++++-+++
Your agent won 95.0% of matches against Greedy Agent


(aind) $python run_match.py -p 4 -r 20 -o MINIMAX -f
Running 40 games:
+++-+--+++++++-++++-+-++++-+++++++-+++++
Running 40 games:
++++++--++++-+++-++-+-++-+++--++-+++++++
Your agent won 77.5% of matches against Minimax Agent
```