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 -

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

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

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 Advanced Heuristic (Failed Attempt):

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) **Partially Failed** 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.
Final formula is:
score = OwnLiberties – OpponentLiberties – DistancePenalty

Result: Good results

The results were good but not consistent, hence I decided to work on advanced options like Monte Carlo Tree Search.

Advanced Search Technique (Monte Carlo Tree Search/ MCTS)

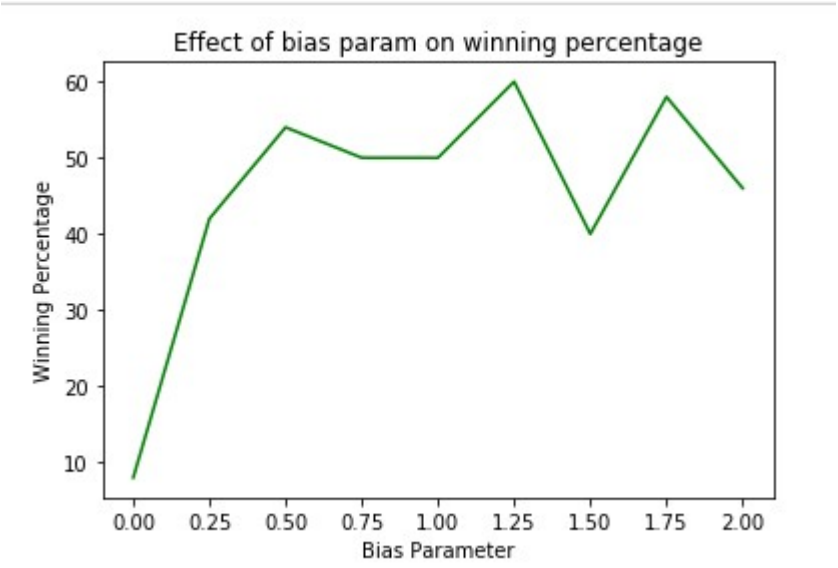
Based on various references mentioned in the references section, MCTS seems to be promising search technique.

The results were not great in the beginning, so there were two option available

- Use data.pickle along with MCTS
- Finetune bias parameter (balance between exploration and exploitation)

Fine tuning bias parameter looked worth trying first (based on 50 games).

Sr. No.	bias_param_const	Winning Percentage against MiniMax agent
1	0	8.0
2	0.25	42.0
3	0.50	54.0
4	0.75	50.0
5	1.00	50.0
6	**1.25**	**60.0**
7	1.50	40.0
8	1.75	58.0
9	2.00	46.0

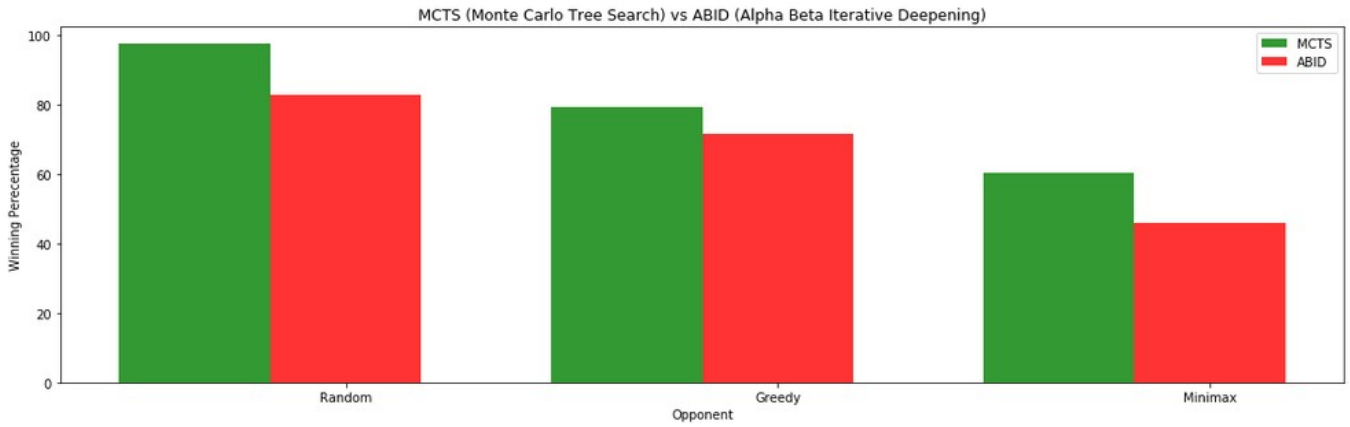


From above results, 1.25 bais param const seems to be giving best results.

Performance of MCTS in comparison with alpha beta iterative deepening based search

Below tables shows games result for 200 games played with different players.

Opponent	MCTS	Alpha-beta with iterative deepening
Random	97.5	83.0
Greedy	79.5	71.5
Minimax	60.5	46.0



Note:

1. Clearly MCTS has performed far better compared to Alpha Beta with iterative deepening.
2. fair_matches flag was off for the simulation, reason: MCTS decides its start position based on simulation result, use of fair flag is not logical with MCTS.
3. Although I have used constant bias factor, I think that further improvements can be achieved if we fine-tune bias parameter based on game state (more exploitation in start and more expansions towards end)
4. Although I have not used opening book, MCTS algorithm can benefit a lot from Opening book. But this will require long time for game plays and huge sized data.pickle file.

Choose a baseline search algorithm for comparison (for example, alpha-beta search with iterative deepening, etc.). How much performance difference does your agent show compared to the baseline?

I have chosen alpha-beta search with iterative deepening for comparison. As it can be seen from the comparison table, MCTS always shows performance 10-15% higher than alpha-beta iterative deepening.

Why do you think the technique you chose was more (or less) effective than the baseline?

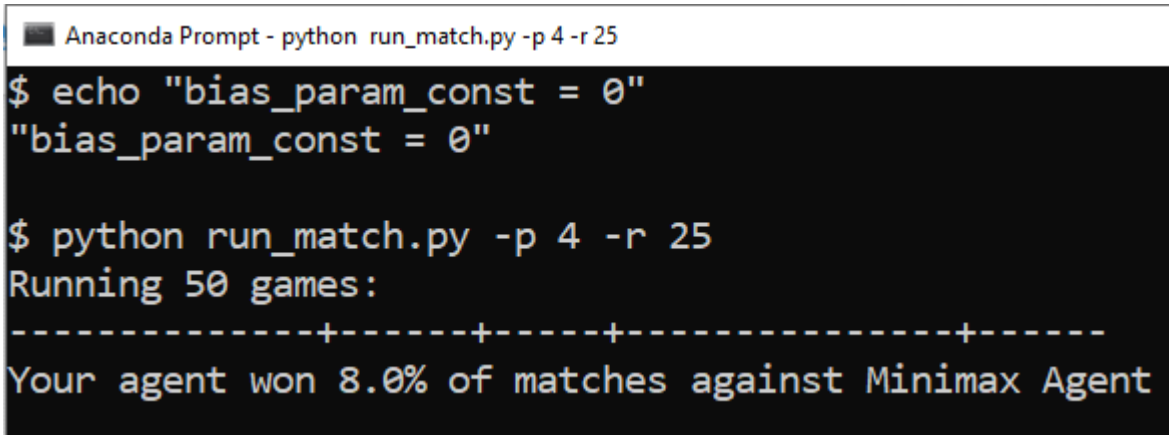
MCTS uses knowledge of game simulation till terminal node, where as alpha beta can search only till depth of 3-4 in given time constraints. Another observation is that MCTS favors better playing nodes compared to the bad ones.

References:

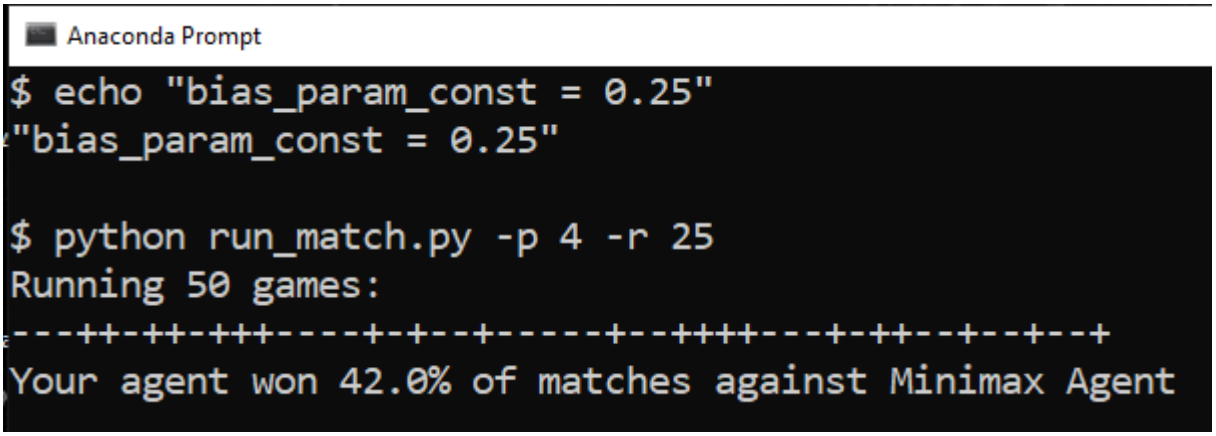
1. http://mcts.ai/about/
2. https://towardsdatascience.com/monte-carlo-tree-search-158a917a8baa
3. https://int8.io/monte-carlo-tree-search-beginners-guide/

Appendix:

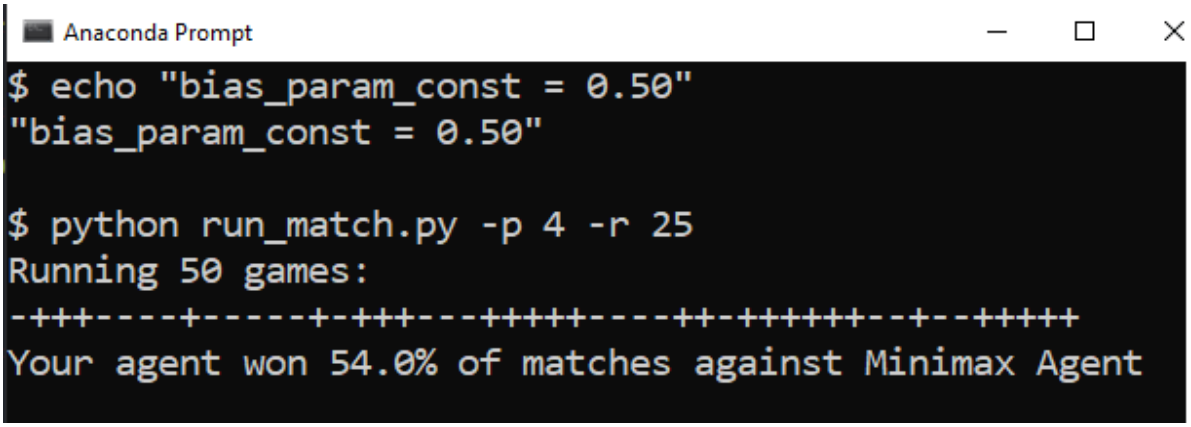
1. bias_param_const = 0



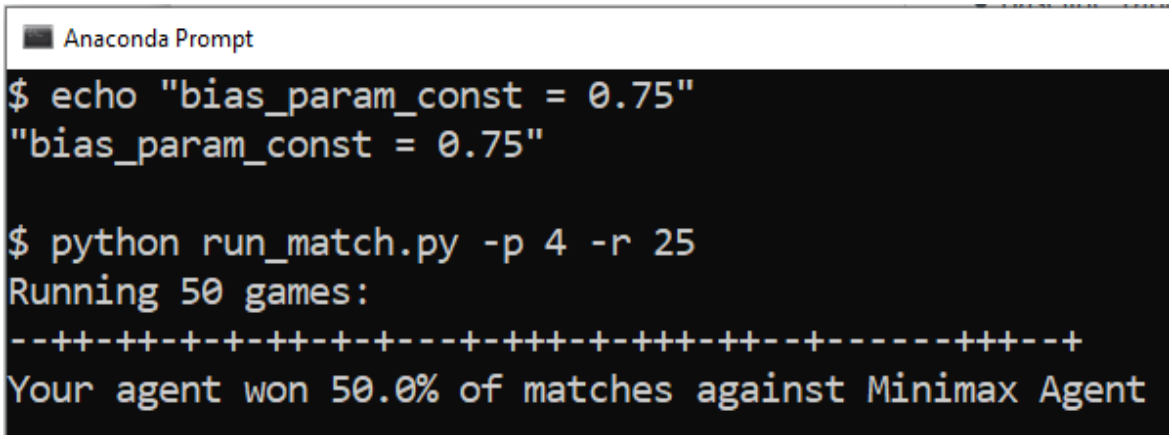
2. bias_param_const = 0.25



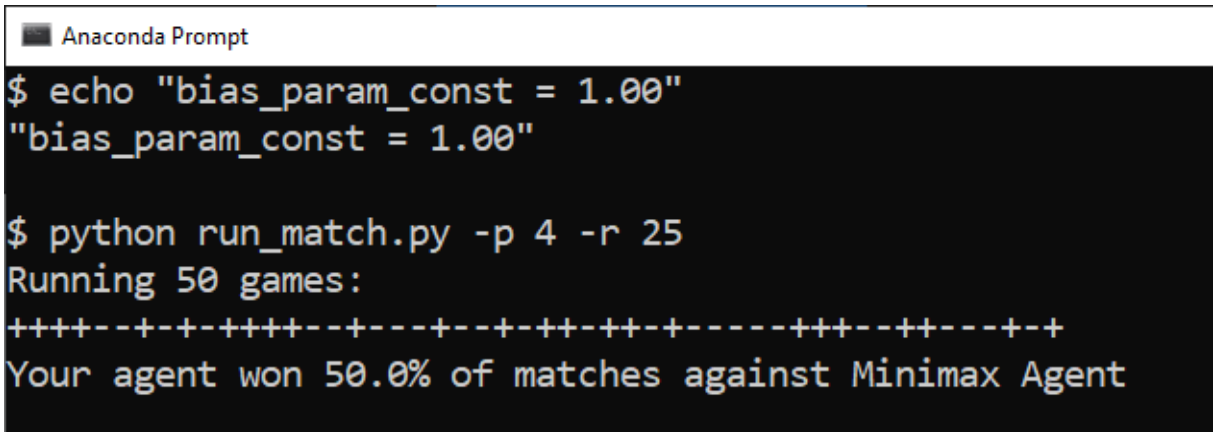
3. bias_param_const = 0.50



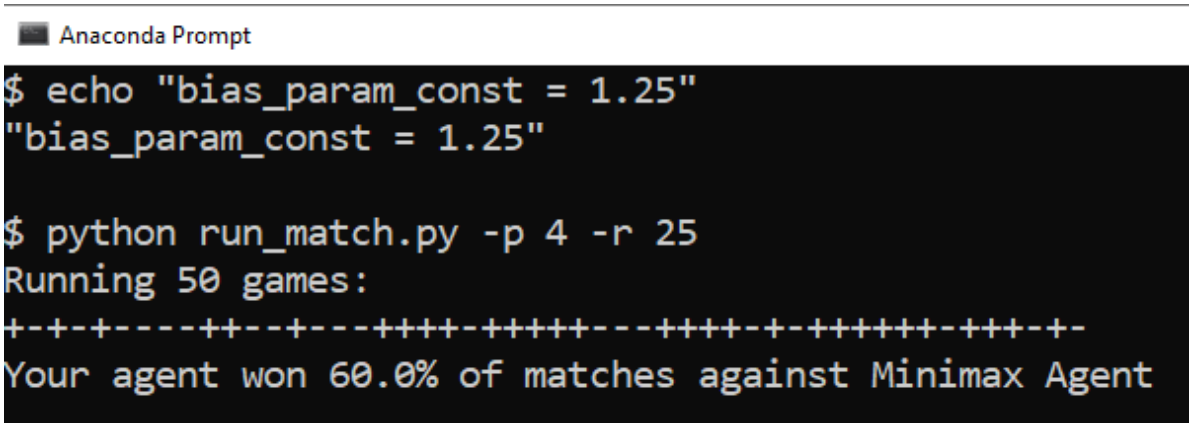
4. bias_param_const = 0.75



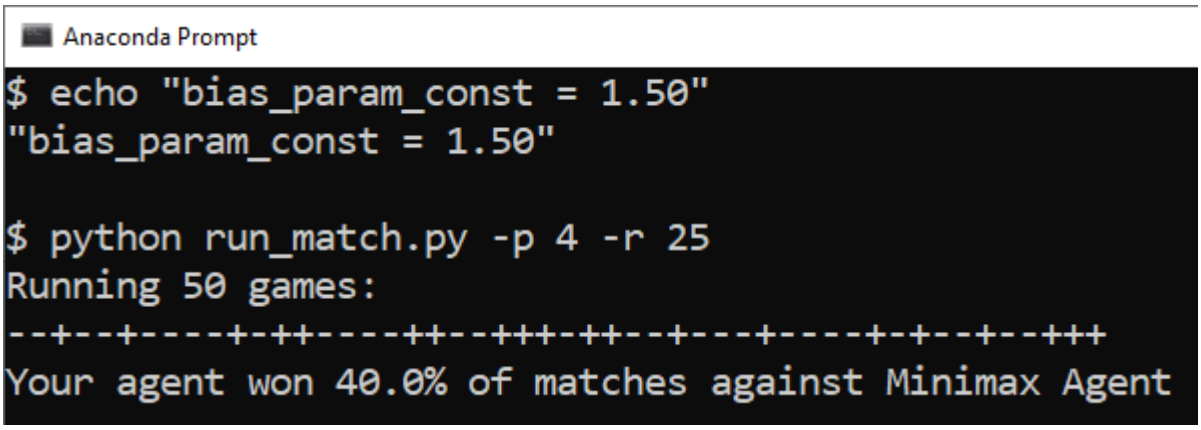
5. bias_param_const = 1.0



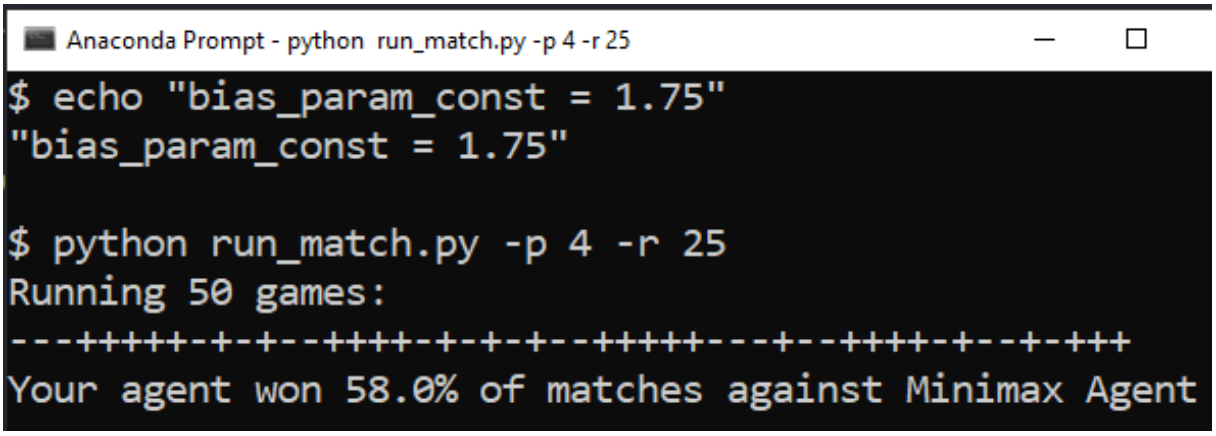
6. bias_param_const = 1.25



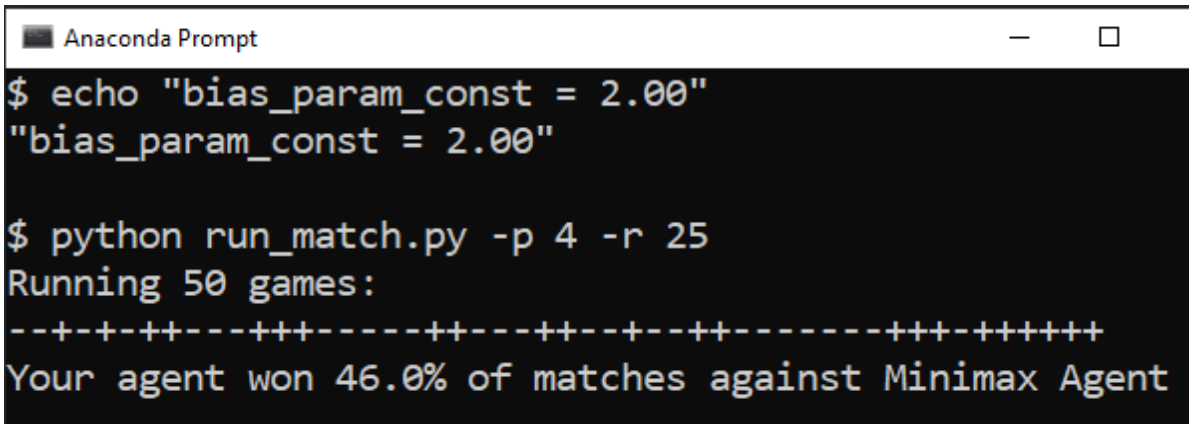
7. bias_param_const = 1.50



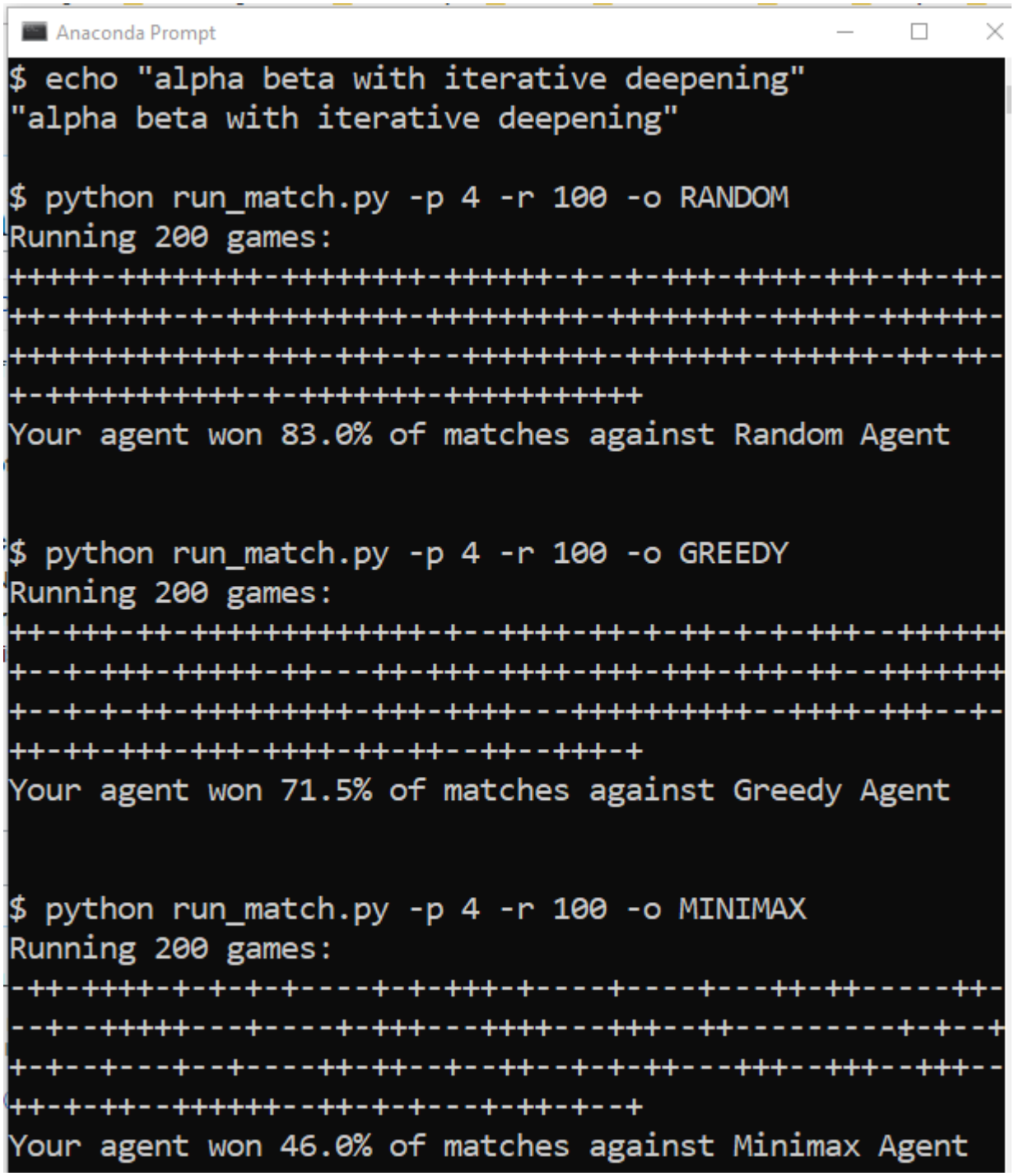
8. bias_param_const = 1.75



9. bias_param_const = 2.00



Log for Alpha Beta Search with Iterative Deepening



Log for Monte Carlo Tree Search with bias_param_const = 1.25