Nick Carey

AI-HW2 Written Portion

1) Total Nodes Explored:

Max Depth	4x4 Minimax	4x4 AlphaBeta	6x6 Minimax	6x6 AlphaBeta	8x8 Minimax	8x8 AlphaBeta
1	7	7	33	33	124	124
2	23.5	19.5	709.5	376	5560	2211.5
3	58	43	655.5	290.5	4118.5	1304.5
4	149.5	90	40557	7693	1527329.5	86975.5
5	367	159	18026	2436.5	280791	21216.5

2)

4x4 board, max depth 8:
note: time is in seconds

Turn	Minimax Nodes	AB Nodes	Minimax Time	AlphaBeta Time
0 (black)	580	191	.0513	.0266
1 (white)	140	66	.0061	.0028
2 (black)	186	50	.0062	.0061
3 (white)	396	168	.0243	.0028
4 (black)	147	38	.0015	.0013
5 (white)	253	119	.0066	.0017

6x6 board, max depth 7:
NOTE: Since there are 27 turns, I am only reporting turns in intervals of 5. Full transcripts of the output and turn data are in the output folder

Turn	Minimax Nodes	AB Nodes	Minimax Time	AlphaBeta Time
0 (black)	5060	962	.1224	.0757
5 (white)	137707	5859	.4385	.0369
10 (black)	107106	6078	.2921	.0173
15 (white)	314648	13207	.5330	.0323
20 (black)	12441	780	.0298	.0018
25 (white)	65	33	.0003	.0003

8x8 board, max depth 6:
NOTE: Since there are 54 turns, I am only reporting intervals of 9. Full transcripts of the output and turn data are located in the output folder.

Turn	Minimax Nodes	AB Nodes	Minimax Time	AlphaBeta Time
0 (black)	8232	1744	.1616	.0961
9 (white)	5606638	167667	27.40	.9920
18 (black)	8040649	110838	33.44	.6259
27 (white)	6605416	403763	22.24	1.823
36 (black)	253386	11480	.5680	.0358
45 (white)	6004	511	.0101	.0018
54 (black)	2	2	.0001	.0001

3) The data here tells us that the AlphaBeta Algorithm is superior to the standard Minimax Algorithm by about an order of magnitude. However, both algorithms scale exponentially in regard to board size. Near the beginning and the end of the game, AlphaBeta does do slightly better than Minimax. However, mid-game is when AlphaBeta really puts the hurt on Minimax. This is because the search tree is the largest mid game. In the beginning, there are a limited number of possible moves to explore. At the end of the game, many branches end prematurely in terminal game-states. Mid-game, however, there are many possible moves to take and many search tree branches to explore.

4) My board evaluation heuristic consists of scoring a board based on how many moves are available to either you or your opponent. Since the game ends when you have no moves available, the heuristic scores boards higher that give you many choices in available moves. Conversely, the heuristic scores boards lower that give your opponent more available moves.

5) It took a lot of thought for me to come up with a decent board evaluation heuristic. Speed was a very important consideration. I did not want a heuristic that investigated possible future-states of a board; that was a job for the minimax or search tree algorithm. Instead, I opted to score a board based only on its current state and not any possible future states. At first I considered and coded up a heuristic that scored a board based on how many chips you have and how many chips your opponent has. The rationale behind this was that if an opponent had less chips, he had less options in terms of movement. Conversely, the more chips I have, the more move options I would have. However, this heuristic did not work out as well as I hoped. Not only did it involve a full scan of the board at every call to the scoring function, but number of chips does not strongly correlate with winning. A player could have many more chips than his opponent, but all his chips could be isolated with no movement options.

I found my next heuristic to be quite acceptable. I decided to evaluate a board based on the amount of movement options available to myself and my opponent. Calculating the number of available moves was achieved by a single method call to the Board state. A board state that gave a player many movement options is favorable because no matter what your opponent does, there is a good chance that some of those movement options will still be available to you next turn. At the same time, denying your opponent movement options limits the amount of choices he has to find a winning strategy.

6) Time Bound will be 3 seconds. Average over 4 runs, 2 on each side

Board Size	Minimax depth	AB max depth	Minimax Wins	AB wins
4x4	6	6	2	2
6x6	10	13	0	4
8x8	8	11	0	4

7) We see that AlphaBeta was able to get to a deeper search tree depth. AlphaBeta was also able to win most of the games. This is because if AlphaBeta is going to be able to go to a deeper search depth than Minimax, then AlphaBeta is going to discover the end-game states first. AlphaBeta will then be able to direct the game to a win state for itself before Minimax can even spot the win-state and avoid it.

AlphaBeta is able to go to a deeper search depth because it can prune branches off of the search tree, allowing to evaluate levels of the search tree faster than Minimax. Evaluating the search tree faster means that AlphaBeta has more time to explore further depths of the search tree.

8) Board Size is 8x8

Time Bound or Depth Bound	Naive M.O. nodes	Heuristic M.O. nodes
3 seconds	10,625,327	2,284,485
5 seconds	18,254,867	2,803,117
Max Depth of 4	50,522	12,159
Max Depth of 6	740,522	126,817

9) Here we see that the Heuristic Move Ordering agent expanded many less nodes than the Naive Move Ordering. This is because ordered moves facilitate Alpha-Beta pruning, and allow the algorithm to skip evaluation of many nodes. With naive move ordering, there is a higher chance that the last move evaluated will be the highest scored move, resulting in many more nodes of the search tree being examined.

10) Of 6 games on an 8x8 board with a time bound of 4 seconds, Naive Move Ordering agent won 1 games and Heuristic Move Ordering agent won 5 games.

The heuristic move ordering agent won more games for the same reason that AlphaBeta won more games versus Minimax; the heuristic move ordering agent was able to prune more branches off of the search tree and examine a deeper level of the search tree in the time allotted. Going even slightly deeper in the search tree gives a better chance of finding a win state first.

11) Weakly Dominant Strategies:
    Player A:
    Player B: p
12) Strongly Dominant Strategies:
    Player A: x
    Player B:

13) There are none. If A chooses x, B can choose either p or r.

14) Weakly Dominant Strategies:
    None
15) Strongly Dominant Strategies:
    None
16) A chooses x, B chooses p