CS 440 Assignment 1 Report

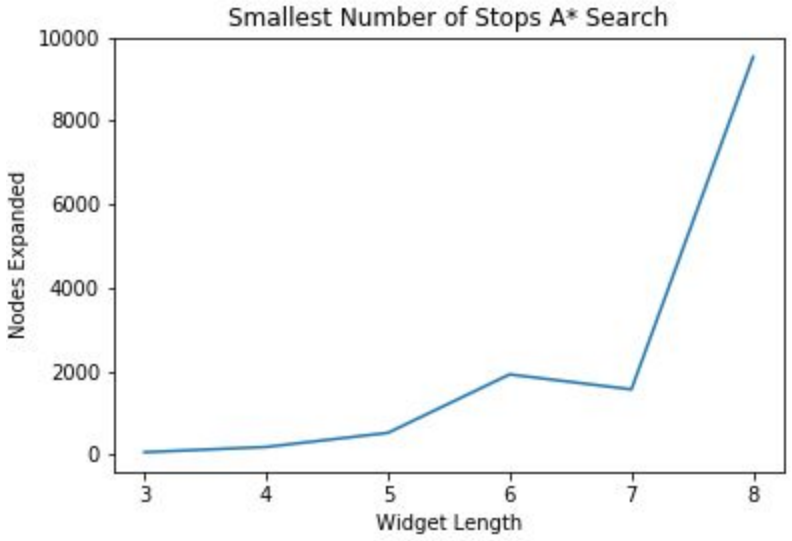
Edbert Linardi, YoungBin Jo, Xiao Tang (3 credit)

Part 1.1

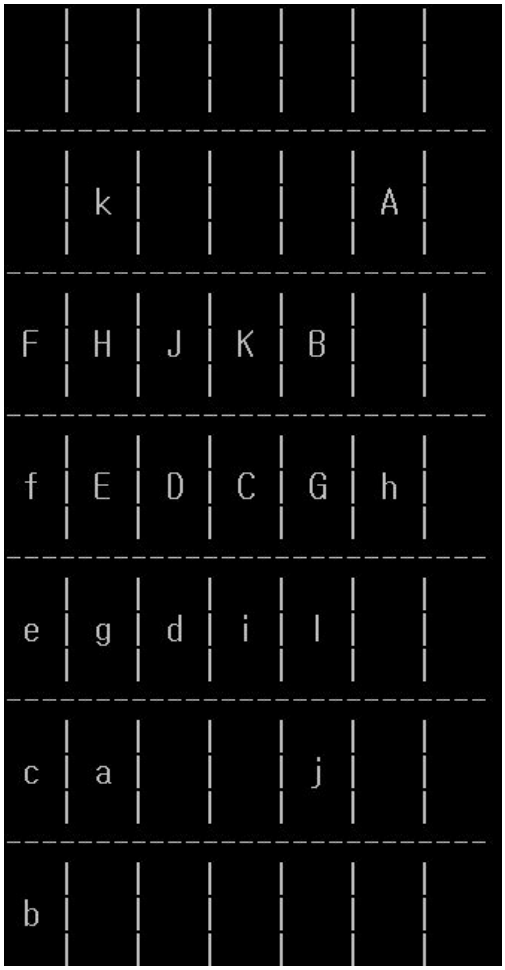
Path with the smallest possible number of stops: DBAEDCBACED
Nodes Expanded: 429
Heuristic used: maximum of length of parts of remaining widgets. This heuristic is admissible, since it will not ever overestimate the number of stops, because the truck will always need to stop at the remaining factories that make each part of the longest remaining widget.

Path with the shortest possible distance: BEDAEDCBCEAD
Nodes Expanded: 678
Distance: 7783
Heuristic used: maximum of total shortest distance between factories that make each part lefts of the remaining widgets. This heuristic is admissible, since it will not overestimate the distance that the truck needs to stop, because the truck always needs to stop at each factories that make each part of the widget with largest distance.
The result of our code here can be maximized. We have tried adding a node with lower cost to the frontier list, even if its widgets left is already in the history. However, there's a tiny bug that prevents us for doing so, even the heuristic and everything is logically correct.

Part 1.3.1



Part 2.1
Small letter = red stone (first player)
Capital letter = blue stone (second player)



Started in (1,1) for first player and (5,5) for second player. It followed 4 rules described in document. Second player is the winner since 5 blocks are in a row from (0,4) to (4,4)

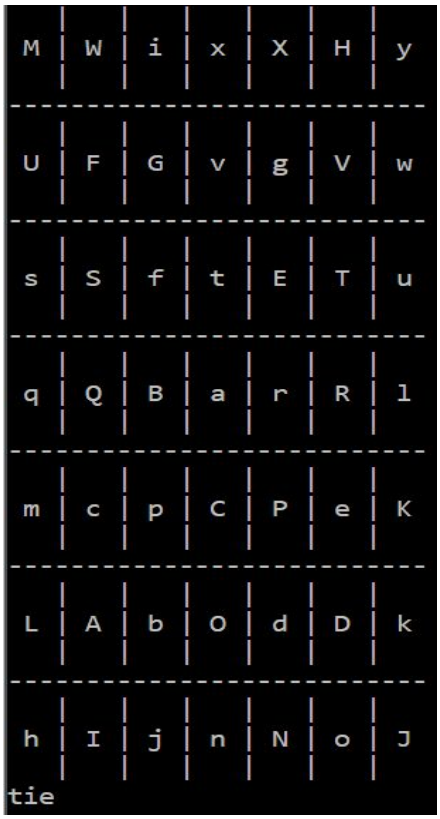
Part 2.2

We generate all possible board positions the two players can produce in 3 turns, evaluate each board position, and assign the value of a move using minimax search. The evaluation function is implemented as scanning through all 6-blocks on the board, and transforming them into a string consists of '0', '1', and '2', where '0' means a open spot, '1' means the player's stone, and '2' means the opponent's stone. If a block matches a pattern, the value of this pattern will be added to result. We decided to scan 6-blocks instead of 5-blocks because some valuable patterns can't be recognized with 5-blocks. However, there are 4 diagonal 5-blocks that can have valuable patterns. We decided to address this by scanning and pattern matching them separately. Pattern-value assignments:

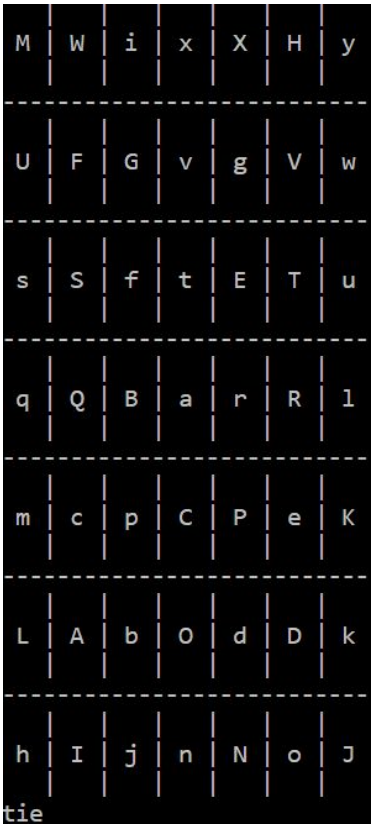
1. 5-in-a-row (11110, 11112, etc.) = 100000 points
   This is a win for the player, so it has a very high value.
2. Opponent 5-in-a-row (22220, 12222, etc.) = -1000000 points
   Because we are scanning the board that is 3 steps ahead of the game, and we didn't check if any of the player has already won the game, it is possible that the opponent can win the game in step 2. We should prevent that from happening, even though we can have 5-in-a-row in step 3. Thus, opponent 5-in-a-row has a very negative value
3. Opponent 4-in-a-row but blocked on one side (122220, 222200, etc) = -100000 points
   With this pattern the opponent can win in the next move, so we need to block. If we don't block the board value will go down significantly.
4. 4-in-a-row open (011110) = 10000 points
   If we have this pattern, the opponent can't stop us from winning the game unless they win in the next move, so this pattern has a high value.
5. Opponent 3-in-a-row open (022200, etc) = -5000 points
   We need to block in this situation unless we already have 4-in-a-row.
6. 4-in-a-row but blocked on one side (211110, etc) = 1000 points
   Not a "guaranteed win", but has some value.
7. 3-in-a-row open (001110, etc) = 1000 points
   Not a "guaranteed win", but has some value.
8. 3-in-a-row blocked (211100, etc) = 100 points
   Has less value than 3-in-a-row open
9. Has win condition, has 2 player stones (200110, etc.) = 10 points
10. Has win condition, has 1 player stones (200010, etc.) = 1 points

(Alpha-Beta implemetation)
The only difference between the minimax and alpha beta is we have the alpha beta value, which is -inf and inf initially. For each child in the min node, if the value is <= alpha, it breaks from the loop. For each child in the max node , if the value is >= beta, it breaks from the loop. Otherwise, if the value is bigger than alpha, it updates the alpha value

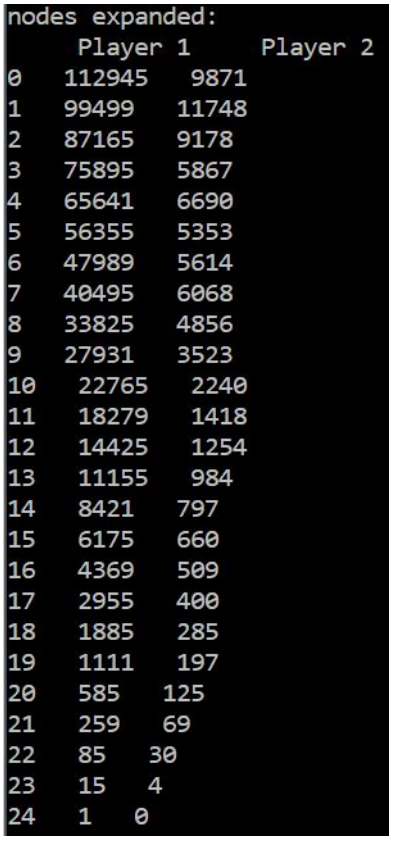
Minimax vs alpha-beta



Alpha-Beta vs Minimax

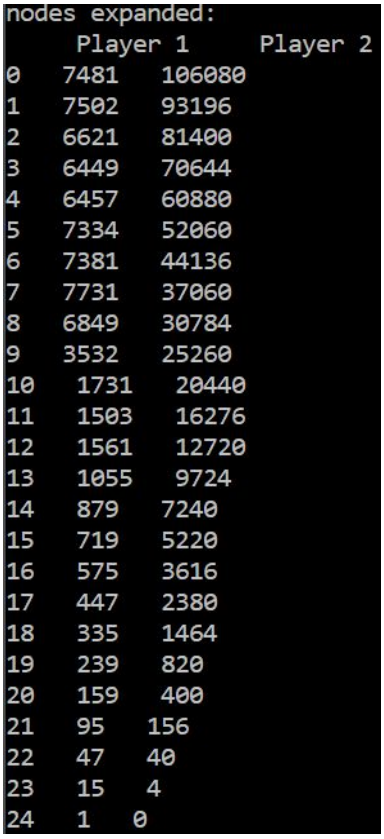


Minimax vs alpha-beta



nodes expanded:		
	Player 1	Player 2
0	112945	9871
1	99499	11748
2	87165	9178
3	75895	5867
4	65641	6690
5	56355	5353
6	47989	5614
7	40495	6068
8	33825	4856
9	27931	3523
10	22765	2240
11	18279	1418
12	14425	1254
13	11155	984
14	8421	797
15	6175	660
16	4369	509
17	2955	400
18	1885	285
19	1111	197
20	585	125
21	259	69
22	85	30
23	15	4
24	1	0

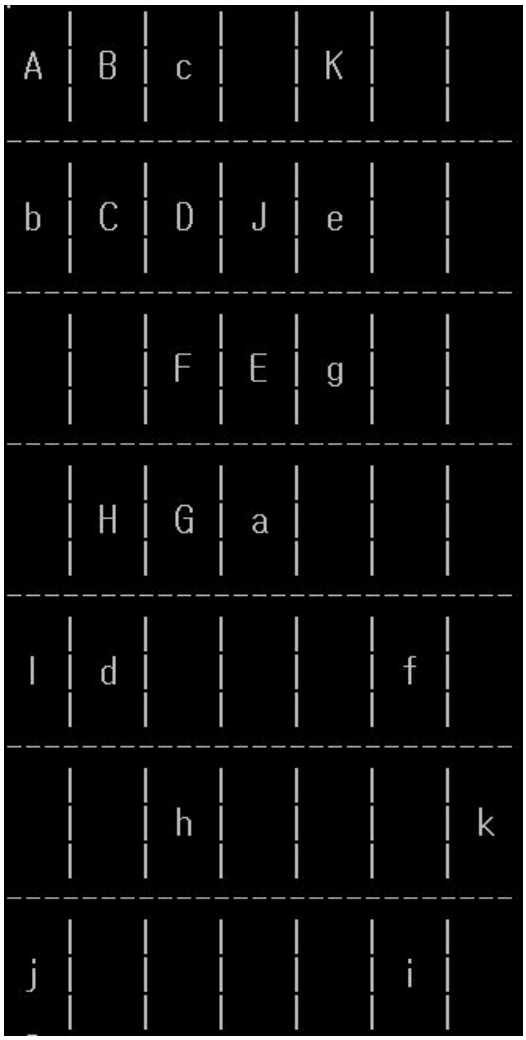
Alpha-Beta vs Minimax



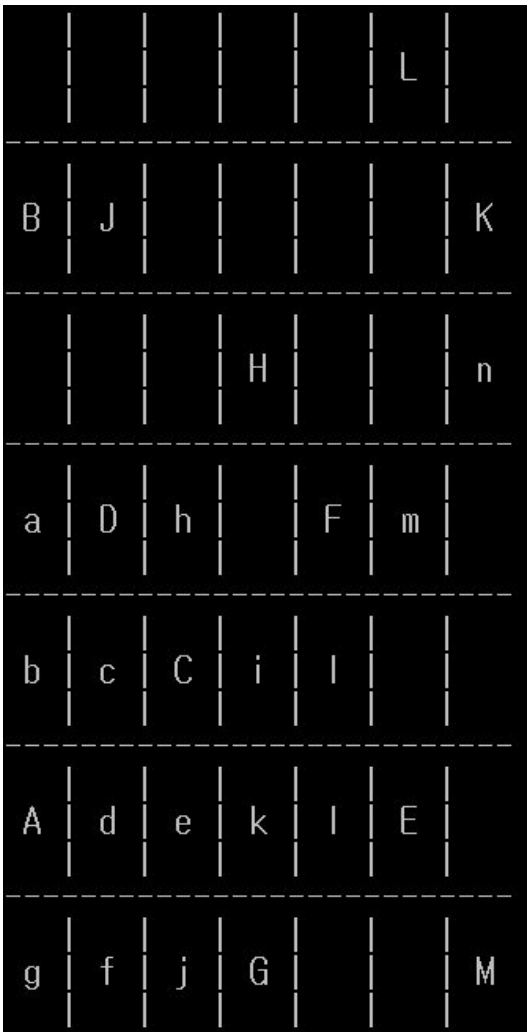
nodes expanded:		
	Player 1	Player 2
0	7481	106080
1	7502	93196
2	6621	81400
3	6449	70644
4	6457	60880
5	7334	52060
6	7381	44136
7	7731	37060
8	6849	30784
9	3532	25260
10	1731	20440
11	1503	16276
12	1561	12720
13	1055	9724
14	879	7240
15	719	5220
16	575	3616
17	447	2380
18	335	1464
19	239	820
20	159	400
21	95	156
22	47	40
23	15	4
24	1	0

As can be seen from this table, Alpha-Beta agent effectively reduced the nodes expanded, while still giving the same results as minimax agent. When the board has a lot of open positions, Alpha-Beta agent reduced the nodes expanded by over 90%.

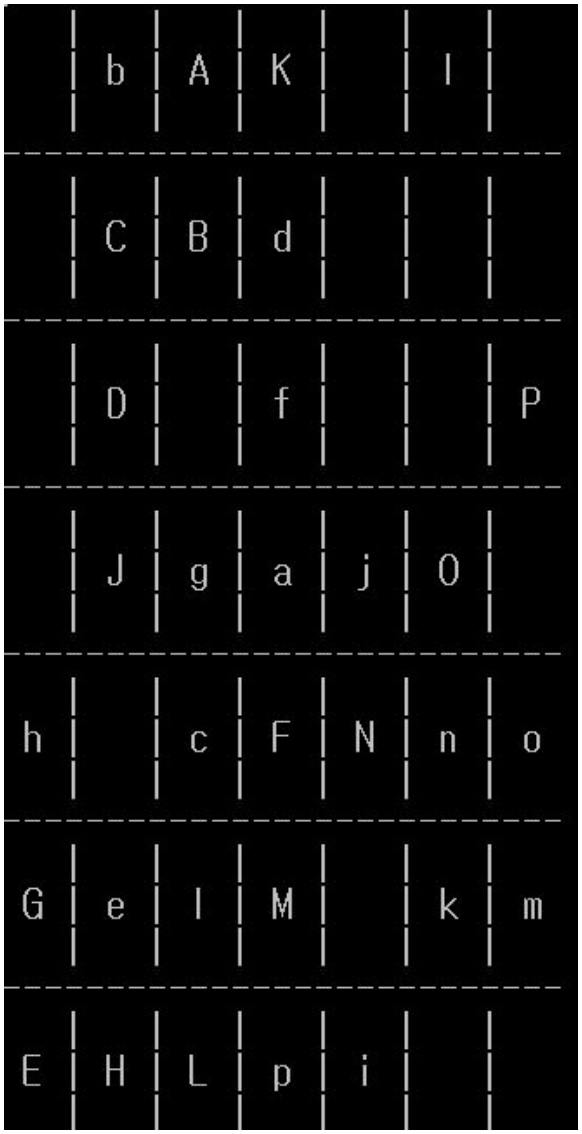
Alpha-Beta vs Reflex



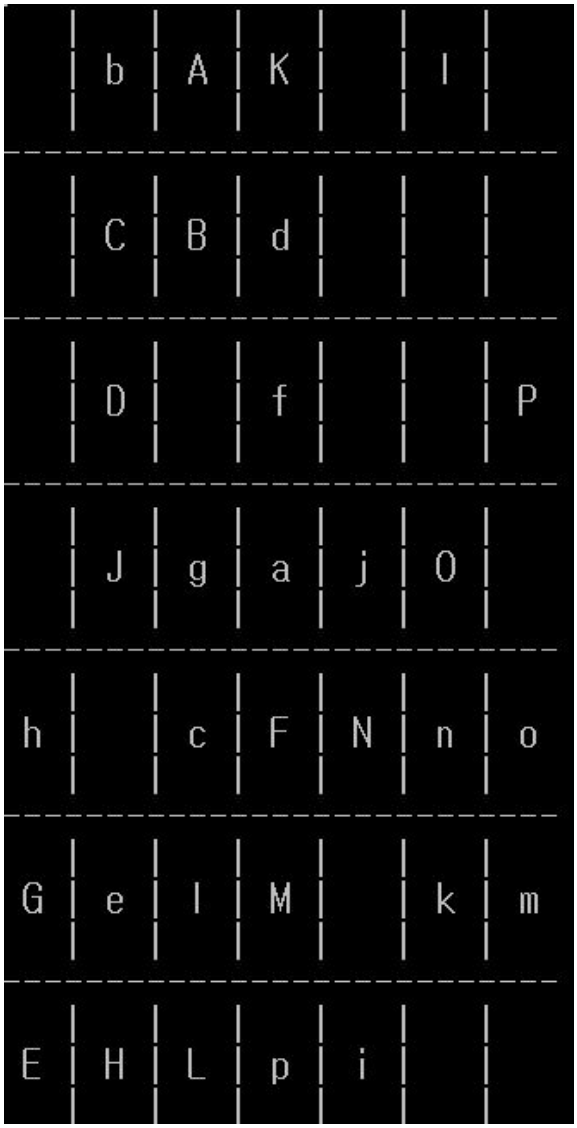
Reflex vs Alpha-Beta



Reflex Vs Minimax



Minimax vs Reflex



Added 2.4.1 video (Human vs Reflex agent) in zip file