Vir Desai

720329873 `

Collaborators: None

HW2

**1. Constraint Satisfaction Problem**

Graduation Dinner

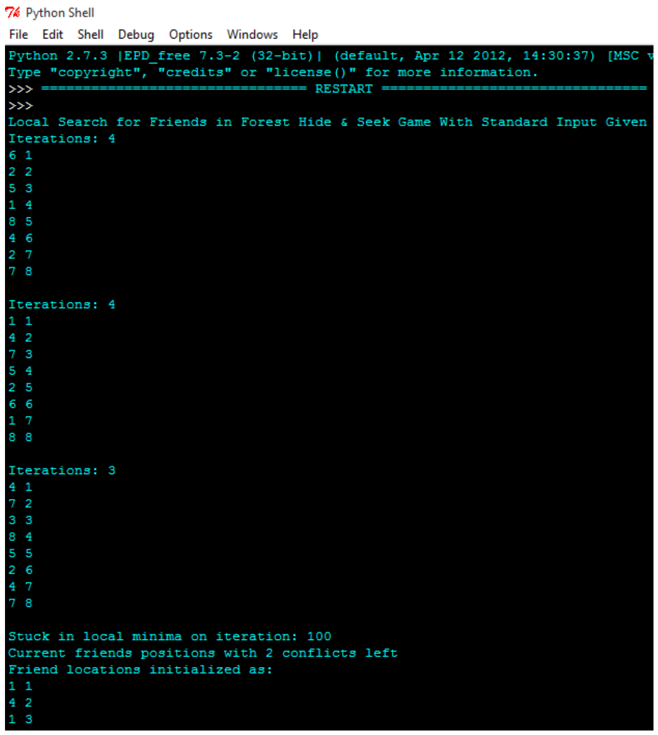
a. A is cousin Jasmine Domain:{1,2,3,4,5}, B is cousin Jason Domain:{1,2,3,4,5}, C is Aunt Trudy Domain:{1,2,3,4,5}, D is Aunt Trudy's husband Randy Domain:{1,2,3,4,5}, E is Aunt Misty Domain:{1,2,3,4,5}. A!=B, A!=C, A!=D, A!=E, B!=C, B!=D, B!=E, C!=D, C!=E, D!=E, |A-B|>1, B<A, |E-B|>1, |E-C|>1, |E-D|>1, |E-C|>=3
b. A: {3,4,5}; B:{1,2,3}; C:{1,2,4,5}; D:{1,2,3,4,5}; E:{1,4,5}
c. A, B, and E would be assigned first using the minimum remaining values heuristic
d. A={4}; B={1,2}; C={1,2}; D={3}; E={5}
e. A=4, B=1, C=2, D=3, E=5 or A=4, B=2, C=1, D=3, E=5

Hide & Seek

For this game using local search, I decided to use the friends as variables instead of every grid location. The domain states of the friends could be and row in the column they were assigned to. That may sound strange but I initialized each friend in 1 column each as per the instructions in a random location/row which row x column value was not already occupied by a tree. So essentially I created my 2D array and went from left to right and checked which rows in each column I could assign friends to and then used a python random number chooser to initialize their locations. This initialization is what leads to some of the success and failure in the algorithm I created. The constraint I already mentioned is that a friend cannot occupy the same row x column position in the array as a tree.

For choosing which friend to move next, I finalized on the choice after trying a few things. An inefficient method I tried was simply checking any of the friends and then if moving them to any other unoccupied row in their column reduced the number of conflicts they had from before the move then I used that and moved to the next iteration. This resulted in a lot of iterations having to be used. I also tried going through and doing basically the same as what I previously mentioned but instead of just seeing if the movement reduced the conflicts by just 1, I checked to see if it reduced the conflict by greater than or equal to half of what it already was at and if it did I chose to move that friend to their new location and did not check other friends until a later iteration. This did not complete the search much of the time as there were many local minima which the algorithm would get stuck on. What I decided to go with was an algorithm which checked every friend per iteration and checked every row they could move to in their respective columns. By check I mean for a friend, I would check the current conflicts they had and then based on that I would go to every open row and see if there were less conflict as a result of the friend being moved there. This gave me a largest reduction of conflicts per friend column based on which row I would move them to. So with knowing the greatest reduction in total conflicts each friend in each column could make, I chose the largest of them and moved them to their new location and then went to the next iteration and checked them all over again.

This obviously method I described checks numerous places and requires a lot of calculations but produces the least number of iterations. From the standard 8x8 input with 15 trees provided, between 60-80% of the time the algorithm succeeded at producing a proper output and usually did so using less than 10 total iterations but otherwise got stuck on local minima. Due to how few iterations my algorithm took I decided to cut off the cycles at 100 iterations because I knew they would be in local minima at that point if it was not solved. The fewer the trees in the forest produced more local minima and my algorithm succeeded less. The bigger the grid with same number (15) of trees required more computing time and increased the iterations and works about at a 50% clip depending on size. More trees in a bigger grid reduces the iterations and increases likelihood of solution being found as randomizing can possibly already put many friends in locations with no conflicts. Usually my local minima occurred when two friends were initialized in consecutive columns (ex 5 and 6) with the same row value (ex both were 4) with few trees in that row. Below are two screenshots of my algorithm running 5 times for this method and working 4 out of 5 with the displays for the final locations of friends and for the time it failed I show what the initial friend locations were and what they finalized at in the local minima before I cut off the search.

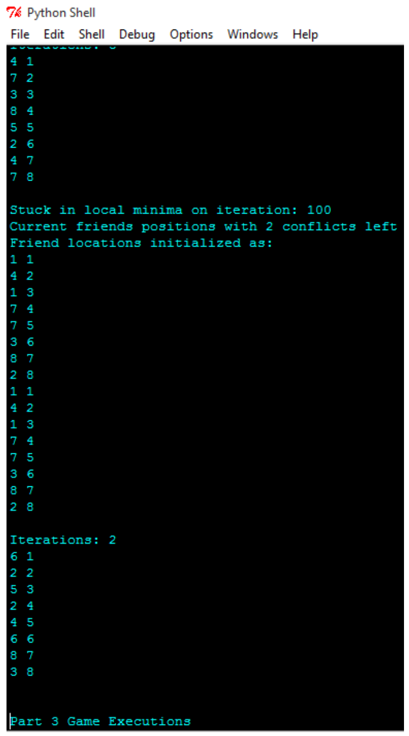


```
Python 2.7.3 |EPD_free 7.3-2 (32-bit)| (default, Apr 12 2012, 14:30:37) [MSC v
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Local Search for Friends in Forest Hide & Seek Game With Standard Input Given
Iterations: 4
6 1
2 2
5 3
1 4
8 5
4 6
2 7
7 8

Iterations: 4
1 1
4 2
7 3
5 4
2 5
6 6
1 7
8 8

Iterations: 3
4 1
7 2
3 3
8 4
5 5
2 6
4 7
7 8

Stuck in local minima on iteration: 100
Current friends positions with 2 conflicts left
Friend locations initialized as:
1 1
4 2
1 3
```



```
4 1
7 2
3 3
8 4
5 5
2 6
4 7
7 8

Stuck in local minima on iteration: 100
Current friends positions with 2 conflicts left
Friend locations initialized as:
1 1
4 2
1 3
7 4
7 5
3 6
8 7
2 8
1 1
4 2
1 3
7 4
7 5
3 6
8 7
2 8

Iterations: 2
6 1
2 2
5 3
2 4
4 5
6 6
8 7
3 8


Part 3 Game Executions
```

**2. Minimax**

<u>Candy Game</u>

For this section I used python as well. I was able to execute my searches and get solutions for all of the game boards using all of the variations of algorithm combinations for depths of 2 and 3 is using minimax and 2, 3, 4, and 5 for alpha-beta. I was able to get all of the AlmondJoy.txt file solved using my minimax with a depth limit of 4 but the execution time was awful as it took 4-6 million explored nodes to execute for each minimax so I aborted that run. My algorithm produces file outputs which are on the

CS drive which have the detailed outputs required for each run I executed. Overall the alpha-beta performs much quicker than does the minimax for the obvious reason of eliminating choices it knows will not be chosen instead of exploring them. The evaluation function I used was a simple. For each iteration and check I ran the minimax and alpha-beta (whichever algorithm I was using) to increment the score of one of the players and decrement the score of the other based on the board choice picked. Decrement would only happen if a node was changed from one color to the other. These values would be added to the current players score if they were the maximizing player choice in the minimax tree depth and decremented from the others. If the current player is the minimizing player than the score is added to the opponent's score and taken away from them. These would be the new scores to run the next version of the minimax with a decreased depth of 1 until the depth was 0 and the two scores were subtracted from each other and returned as the value. This went back up the tree of recursion for how ever deep it was and returned the value. If the current player is a max player it took the max of a low value and the value returned and used that as the new large otherwise if it was min it did the same but took the min value vs a large number and made the lowest the min. When the depth for this search was reached it used the best value it found as the optimal next move. This was checked for every position on the board to find the best choices.

Alpha-beta was similar but in addition to the score changing based on whether the current player was a maximizing or minimizing player, the algorithm first broke up earlier and if the player was maximizing it checked for anything higher than the alpha value it had and if it was minimizing it checked for values lower than the beta value. If either branches of this had a beta value less than or equal to alpha it broke out of the function and returned the alpha (if maximizing) or beta (if minimizing) value from it. Also instead of recursively calling minimax for all the grid locations like the previous algorithm, this recursively calls itself for all the grid values but only to a depth assigned. I have updating functions and such which basically check the current cell and the neighbors of it and if any of the neighbors are the same color as the current cell and there are other neighbor cells which are the other color then it changed them to its own color and added that value to that player and subtracted it from the other. Below are some of the combinations of algorithms run on the different game boards with different depth limits.

```
On File: AlmondJoy.txt with depth limit: 3 on algorithm: Alphabeta for
Blue/Player 1 and depth limit: 3 on algorithm: Alphabeta for
Green/Player2

Blue: drop A1, Green: drop B1 then
Blue: drop C1, Green: drop D1 then
Blue: drop E1, Green: drop F1 then
Blue: drop A2, Green: drop B2 then
Blue: drop C2, Green: drop D2 then
Blue: drop E2, Green: drop F2 then
Blue: drop A3, Green: drop B3 then
Blue: drop C3, Green: drop D3 then
Blue: drop E3, Green: drop F3 then
Blue: drop A4, Green: drop B4 then
Blue: drop C4, Green: drop D4 then
Blue: drop E4, Green: drop F4 then
Blue: drop A5, Green: drop B5 then
Blue: drop C5, Green: drop D5 then
Blue: drop E5, Green: drop F5 then
Blue: drop A6, Green: drop B6 then
Blue: drop C6, Green: drop D6 then
Blue: drop E6, Green: drop F6

B     G     B     G     B     G
G     B     G     B     G     G
B     G     B     G     G     G
G     B     G     G     G     G
B     G     G     G     G     G
G     G     G     G     G     G

Player 1 Total Score: 9
Player 2 Total Score: 27
Total Nodes Expanded by Player 1: 8058
Total Nodes Expanded by Player 2: 7415
Avg Nodes Expanded by Player 1 per move: 447.666666667
Avg Nodes Expanded by Player 2 per move: 411.944444444
Avg Time Spent by Player 1 per move: 0.017833325598
Avg Time Spent by Player 2 per move: 0.016666677263
```

```
On File: AlmondJoy.txt with depth limit: 4 on algorithm: Alphabeta for
Blue/Player 1 and depth limit: 4 on algorithm: Alphabeta for
Green/Player2

Blue: drop A1, Green: drop B1 then
Blue: drop C1, Green: drop D1 then
Blue: drop E1, Green: drop F1 then
Blue: drop A2, Green: drop B2 then
Blue: drop C2, Green: drop D2 then
Blue: drop E2, Green: drop F2 then
Blue: drop A3, Green: drop B3 then
Blue: drop C3, Green: drop D3 then
Blue: drop E3, Green: drop F3 then
Blue: drop A4, Green: drop B4 then
Blue: drop C4, Green: drop D4 then
Blue: drop E4, Green: drop F4 then
Blue: drop A5, Green: drop B5 then
Blue: drop C5, Green: drop D5 then
Blue: drop E5, Green: drop F5 then
Blue: drop A6, Green: drop B6 then
Blue: drop C6, Green: drop D6 then
Blue: drop E6, Green: drop F6



B     G     B     G     B     G
G     B     G     B     G     G
B     G     B     G     G     G
G     B     G     G     G     G
B     G     G     G     G     G
G     G     G     G     G     G

Player 1 Total Score: 9
Player 2 Total Score: 27
Total Nodes Expanded by Player 1: 14865
Total Nodes Expanded by Player 2: 13632
Avg Nodes Expanded by Player 1 per move: 825.833333333
Avg Nodes Expanded by Player 2 per move: 757.333333333
Avg Time Spent by Player 1 per move: 0.0574999782774
Avg Time Spent by Player 2 per move: 0.0497222476535
```

```
On File: AlmondJoy.txt with depth limit: 5 on algorithm: Alphabeta for
Blue/Player 1 and depth limit: 5 on algorithm: Alphabeta for
Green/Player2

Blue: drop A1, Green: drop B1 then
Blue: drop C1, Green: drop D1 then
Blue: drop E1, Green: drop F1 then
Blue: drop A2, Green: drop B2 then
Blue: drop C2, Green: drop D2 then
Blue: drop E2, Green: drop F2 then
Blue: drop A3, Green: drop B3 then
Blue: drop C3, Green: drop D3 then
Blue: drop E3, Green: drop F3 then
Blue: drop A4, Green: drop B4 then
Blue: drop C4, Green: drop D4 then
Blue: drop E4, Green: drop F4 then
Blue: drop A5, Green: drop B5 then
Blue: drop C5, Green: drop D5 then
Blue: drop E5, Green: drop F5 then
Blue: drop A6, Green: drop B6 then
Blue: drop C6, Green: drop D6 then
Blue: drop E6, Green: drop F6



B     G     B     G     B     G
G     B     G     B     G     G
B     G     B     G     G     G
G     B     G     G     G     G
B     G     G     G     G     G
G     G     G     G     G     G

Player 1 Total Score: 9
Player 2 Total Score: 27
Total Nodes Expanded by Player 1: 193152
Total Nodes Expanded by Player 2: 172129
Avg Nodes Expanded by Player 1 per move: 10730.6666667
Avg Nodes Expanded by Player 2 per move: 9562.72222222
Avg Time Spent by Player 1 per move: 0.488555550575
Avg Time Spent by Player 2 per move: 0.452388895883
```

```
On File: AlmondJoy.txt with depth limit: 5 on algorithm: Alphabeta for
Blue/Player 1 and depth limit: 3 on algorithm: Minimax for
Green/Player2

Blue: drop A1, Green: drop F6 then
Blue: drop B1, Green: drop E6 then
Blue: drop C1, Green: drop D6 then
Blue: drop D1, Green: drop C6 then
Blue: drop E1, Green: drop B6 then
Blue: drop F1, Green: drop A6 then
Blue: drop A2, Green: drop F5 then
Blue: drop B2, Green: drop E5 then
Blue: drop C2, Green: drop D5 then
Blue: drop D2, Green: drop C5 then
Blue: drop E2, Green: drop B5 then
Blue: drop F2, Green: drop A5 then
Blue: drop A3, Green: drop F4 then
Blue: drop B3, Green: drop E4 then
Blue: drop C3, Green: drop D4 then
Blue: drop D3, Green: drop C4 then
Blue: drop E3, Green: drop B4 then
Blue: drop F3, Green: drop A4



B     B     B     B     B     B
B     B     B     B     B     B
G     G     G     B     B     B
G     G     G     G     B     B
G     G     G     G     G     G
G     G     G     G     G     G

Player 1 Total Score: 17
Player 2 Total Score: 19
Total Nodes Expanded by Player 1: 193152
Total Nodes Expanded by Player 2: 186979
Avg Nodes Expanded by Player 1 per move: 10730.6666667
Avg Nodes Expanded by Player 2 per move: 10387.7222222
Avg Time Spent by Player 1 per move: 0.449277745353
Avg Time Spent by Player 2 per move: 0.396611134211
```

```
On File: AlmondJoy.txt with depth limit: 5 on algorithm: Alphabeta for
Blue/Player 1 and depth limit: 4 on algorithm: Minimax for
Green/Player2

Blue: drop A1, Green: drop F6 then
Blue: drop B1, Green: drop E6 then
Blue: drop C1, Green: drop D6 then
Blue: drop D1, Green: drop C6 then
Blue: drop E1, Green: drop B6 then
Blue: drop F1, Green: drop A6 then
Blue: drop A2, Green: drop F5 then
Blue: drop B2, Green: drop E5 then
Blue: drop C2, Green: drop D5 then
Blue: drop D2, Green: drop C5 then
Blue: drop E2, Green: drop B5 then
Blue: drop F2, Green: drop A5 then
Blue: drop A3, Green: drop F4 then
Blue: drop B3, Green: drop E4 then
Blue: drop C3, Green: drop D4 then
Blue: drop D3, Green: drop C4 then
Blue: drop E3, Green: drop B4 then
Blue: drop F3, Green: drop A4



B     B     B     B     B     B
B     B     B     B     B     B
G     G     G     B     B     B
G     G     G     G     B     B
G     G     G     G     G     G
G     G     G     G     G     G

Player 1 Total Score: 17
Player 2 Total Score: 19
Total Nodes Expanded by Player 1: 193159
Total Nodes Expanded by Player 2: 4856844
Avg Nodes Expanded by Player 1 per move: 10731.0555556
Avg Nodes Expanded by Player 2 per move: 269824.666667
Avg Time Spent by Player 1 per move: 0.493277801408
Avg Time Spent by Player 2 per move: 11.1934999757
```

```
On File: AlmondJoy.txt with depth limit: 4 on algorithm: Minimax for
Blue/Player 1 and depth limit: 4 on algorithm: Minimax for
Green/Player2

Blue: drop F6, Green: drop E6 then
Blue: drop D6, Green: drop C6 then
Blue: drop B6, Green: drop A6 then
Blue: drop F5, Green: drop E5 then
Blue: drop D5, Green: drop C5 then
Blue: drop B5, Green: drop A5 then
Blue: drop F4, Green: drop E4 then
Blue: drop D4, Green: drop C4 then
Blue: drop B4, Green: drop A4 then
Blue: drop F3, Green: drop E3 then
Blue: drop D3, Green: drop C3 then
Blue: drop B3, Green: drop A3 then
Blue: drop F2, Green: drop E2 then
Blue: drop D2, Green: drop C2 then
Blue: drop B2, Green: drop A2 then
Blue: drop F1, Green: drop E1 then
Blue: drop D1, Green: drop C1 then
Blue: drop B1, Green: drop A1


G     G     G     G     G     G
G     G     G     G     G     B
G     G     G     G     B     G
G     G     G     B     G     B
G     G     B     G     B     G
G     B     G     B     G     B

Player 1 Total Score: 9
Player 2 Total Score: 27
Total Nodes Expanded by Player 1: 5604705
Total Nodes Expanded by Player 2: 4856832
Avg Nodes Expanded by Player 1 per move: 311372.5
Avg Nodes Expanded by Player 2 per move: 269824.0
Avg Time Spent by Player 1 per move: 11.5891111294
Avg Time Spent by Player 2 per move: 10.1569444338
```

```
On File: AlmondJoy.txt with depth limit: 3 on algorithm: Minimax for
Blue/Player 1 and depth limit: 5 on algorithm: Alphabeta for
Green/Player2

Blue: drop F6, Green: drop A1 then
Blue: drop E6, Green: drop B1 then
Blue: drop D6, Green: drop C1 then
Blue: drop C6, Green: drop D1 then
Blue: drop B6, Green: drop E1 then
Blue: drop A6, Green: drop F1 then
Blue: drop F5, Green: drop A2 then
Blue: drop E5, Green: drop B2 then
Blue: drop D5, Green: drop C2 then
Blue: drop C5, Green: drop D2 then
Blue: drop B5, Green: drop E2 then
Blue: drop A5, Green: drop F2 then
Blue: drop F4, Green: drop A3 then
Blue: drop E4, Green: drop B3 then
Blue: drop D4, Green: drop C3 then
Blue: drop C4, Green: drop D3 then
Blue: drop B4, Green: drop E3 then
Blue: drop A4, Green: drop F3


G     G     G     G     G     G
G     G     G     G     G     G
B     B     G     G     G     G
B     B     B     G     G     G
B     B     B     B     B     B
B     B     B     B     B     B

Player 1 Total Score: 17
Player 2 Total Score: 19
Total Nodes Expanded by Player 1: 209304
Total Nodes Expanded by Player 2: 172119
Avg Nodes Expanded by Player 1 per move: 11628.0
Avg Nodes Expanded by Player 2 per move: 9562.16666667
Avg Time Spent by Player 1 per move: 0.442166672813
Avg Time Spent by Player 2 per move: 0.40566666921
```

So I just showed a good amount of variations of the outputs for just AlmondJoy.txt and as shown, the execution time per move and number of explored nodes for the minimax algorithm with a depth of 4 is above 10 seconds each and over 4 nodes. I'll continue showing just the execution of minimax:3 vs alphabeta:5 and alphabeta:5 vs minimax:3 for the rest of the game boards. This is the choice because as it will show, the total nodes expanded and average execution times are the closest of any of the combinations at this level. Generally, player 2 has a higher score than player 1 no matter the algorithm selected. Only a few times is that different. The full outputs of all of the combinations of algorithms, game boards, and reasonable depth-limits will be on the CS server and output by my code there as well.

```
On File: Ayds.txt with depth limit: 3 on algorithm: Minimax for
Blue/Player 1 and depth limit: 5 on algorithm: Alphabeta for
Green/Player2

Blue: drop F6, Green: drop A1 then
Blue: drop E6, Green: drop B1 then
Blue: drop D6, Green: drop C1 then
Blue: drop C6, Green: drop D1 then
Blue: drop B6, Green: drop E1 then
Blue: drop A6, Green: drop F1 then
Blue: drop F5, Green: drop A2 then
Blue: drop E5, Green: drop B2 then
Blue: drop D5, Green: drop C2 then
Blue: drop C5, Green: drop D2 then
Blue: drop B5, Green: drop E2 then
Blue: drop A5, Green: drop F2 then
Blue: drop F4, Green: drop A3 then
Blue: drop E4, Green: drop B3 then
Blue: drop D4, Green: drop C3 then
Blue: drop C4, Green: drop D3 then
Blue: drop B4, Green: drop E3 then
Blue: drop A4, Green: drop F3


G    G    G    G    G    G
G    G    G    G    G    G
B    B    G    G    G    G
B    B    B    G    G    G
B    B    B    B    B    B
B    B    B    B    B    B

Player 1 Total Score: 801
Player 2 Total Score: 999
Total Nodes Expanded by Player 1: 209304
Total Nodes Expanded by Player 2: 172119
Avg Nodes Expanded by Player 1 per move: 11628.0
Avg Nodes Expanded by Player 2 per move: 9562.16666667
Avg Time Spent by Player 1 per move: 0.443944454193
Avg Time Spent by Player 2 per move: 0.40711110168
```

```
On File: Ayds.txt with depth limit: 5 on algorithm: Alphabeta for
Blue/Player 1 and depth limit: 3 on algorithm: Minimax for
Green/Player2

Blue: drop A1, Green: drop F6 then
Blue: drop B1, Green: drop E6 then
Blue: drop C1, Green: drop D6 then
Blue: drop D1, Green: drop C6 then
Blue: drop E1, Green: drop B6 then
Blue: drop F1, Green: drop A6 then
Blue: drop A2, Green: drop F5 then
Blue: drop B2, Green: drop E5 then
Blue: drop C2, Green: drop D5 then
Blue: drop D2, Green: drop C5 then
Blue: drop E2, Green: drop B5 then
Blue: drop F2, Green: drop A5 then
Blue: drop A3, Green: drop F4 then
Blue: drop B3, Green: drop E4 then
Blue: drop C3, Green: drop D4 then
Blue: drop D3, Green: drop C4 then
Blue: drop E3, Green: drop B4 then
Blue: drop F3, Green: drop A4



B     B     B     B     B     B
B     B     B     B     B     B
G     G     G     B     B     B
G     G     G     G     B     B
G     G     G     G     G     G
G     G     G     G     G     G

Player 1 Total Score: 801
Player 2 Total Score: 999
Total Nodes Expanded by Player 1: 193152
Total Nodes Expanded by Player 2: 186979
Avg Nodes Expanded by Player 1 per move: 10730.6666667
Avg Nodes Expanded by Player 2 per move: 10387.7222222
Avg Time Spent by Player 1 per move: 0.448055545489
Avg Time Spent by Player 2 per move: 0.399611115456
```

```
On File: Bit-O-Honey.txt with depth limit: 3 on algorithm: Minimax for
Blue/Player 1 and depth limit: 5 on algorithm: Alphabeta for
Green/Player2

Blue: drop F6, Green: drop A1 then
Blue: drop E6, Green: drop B1 then
Blue: drop D6, Green: drop C1 then
Blue: drop C6, Green: drop D1 then
Blue: drop B6, Green: drop E1 then
Blue: drop A6, Green: drop F1 then
Blue: drop F5, Green: drop A2 then
Blue: drop E5, Green: drop B2 then
Blue: drop D5, Green: drop C2 then
Blue: drop C5, Green: drop D2 then
Blue: drop B5, Green: drop E2 then
Blue: drop A5, Green: drop F2 then
Blue: drop F4, Green: drop A3 then
Blue: drop E4, Green: drop B3 then
Blue: drop D4, Green: drop C3 then
Blue: drop C4, Green: drop D3 then
Blue: drop B4, Green: drop E3 then
Blue: drop A4, Green: drop F3



G     G     G     G     G     G
G     G     G     G     G     G
B     B     G     G     G     G
B     B     B     G     G     G
B     B     B     B     B     B
B     B     B     B     B     B

Player 1 Total Score: 320
Player 2 Total Score: 58
Total Nodes Expanded by Player 1: 209304
Total Nodes Expanded by Player 2: 172119
Avg Nodes Expanded by Player 1 per move: 11628.0
Avg Nodes Expanded by Player 2 per move: 9562.16666667
Avg Time Spent by Player 1 per move: 0.443666643567
Avg Time Spent by Player 2 per move: 0.40600001812
```

```
On File: Bit-O-Honey.txt with depth limit: 5 on algorithm: Alphabeta
for Blue/Player 1 and depth limit: 3 on algorithm: Minimax for
Green/Player2

Blue: drop A1, Green: drop F6 then
Blue: drop B1, Green: drop E6 then
Blue: drop C1, Green: drop D6 then
Blue: drop D1, Green: drop C6 then
Blue: drop E1, Green: drop B6 then
Blue: drop F1, Green: drop A6 then
Blue: drop A2, Green: drop F5 then
Blue: drop B2, Green: drop E5 then
Blue: drop C2, Green: drop D5 then
Blue: drop D2, Green: drop C5 then
Blue: drop E2, Green: drop B5 then
Blue: drop F2, Green: drop A5 then
Blue: drop A3, Green: drop F4 then
Blue: drop B3, Green: drop E4 then
Blue: drop C3, Green: drop D4 then
Blue: drop D3, Green: drop C4 then
Blue: drop E3, Green: drop B4 then
Blue: drop F3, Green: drop A4



B     B     B     B     B     B
B     B     B     B     B     B
G     G     G     B     B     B
G     G     G     G     B     B
G     G     G     G     G     G
G     G     G     G     G     G

Player 1 Total Score: 46
Player 2 Total Score: 332
Total Nodes Expanded by Player 1: 193152
Total Nodes Expanded by Player 2: 186979
Avg Nodes Expanded by Player 1 per move: 10730.6666667
Avg Nodes Expanded by Player 2 per move: 10387.7222222
Avg Time Spent by Player 1 per move: 0.450722230805
Avg Time Spent by Player 2 per move: 0.395944436391
```

```
On File: Mounds.txt with depth limit: 5 on algorithm: Alphabeta for
Blue/Player 1 and depth limit: 3 on algorithm: Minimax for
Green/Player2

Blue: drop A1, Green: drop F6 then
Blue: drop B1, Green: drop E6 then
Blue: drop C1, Green: drop D6 then
Blue: drop D1, Green: drop C6 then
Blue: drop E1, Green: drop B6 then
Blue: drop F1, Green: drop A6 then
Blue: drop A2, Green: drop F5 then
Blue: drop B2, Green: drop E5 then
Blue: drop C2, Green: drop D5 then
Blue: drop D2, Green: drop C5 then
Blue: drop E2, Green: drop B5 then
Blue: drop F2, Green: drop A5 then
Blue: drop A3, Green: drop F4 then
Blue: drop B3, Green: drop E4 then
Blue: drop C3, Green: drop D4 then
Blue: drop D3, Green: drop C4 then
Blue: drop E3, Green: drop B4 then
Blue: drop F3, Green: drop A4



B     B     B     B     B     B
B     B     B     B     B     B
G     G     G     B     B     B
G     G     G     G     B     B
G     G     G     G     G     G
G     G     G     G     G     G

Player 1 Total Score: 34
Player 2 Total Score: 38
Total Nodes Expanded by Player 1: 193152
Total Nodes Expanded by Player 2: 186979
Avg Nodes Expanded by Player 1 per move: 10730.6666667
Avg Nodes Expanded by Player 2 per move: 10387.7222222
Avg Time Spent by Player 1 per move: 0.448388907644
Avg Time Spent by Player 2 per move: 0.398944430881
```

```
On File: Mounds.txt with depth limit: 3 on algorithm: Minimax for
Blue/Player 1 and depth limit: 5 on algorithm: Alphabeta for
Green/Player2

Blue: drop F6, Green: drop A1 then
Blue: drop E6, Green: drop B1 then
Blue: drop D6, Green: drop C1 then
Blue: drop C6, Green: drop D1 then
Blue: drop B6, Green: drop E1 then
Blue: drop A6, Green: drop F1 then
Blue: drop F5, Green: drop A2 then
Blue: drop E5, Green: drop B2 then
Blue: drop D5, Green: drop C2 then
Blue: drop C5, Green: drop D2 then
Blue: drop B5, Green: drop E2 then
Blue: drop A5, Green: drop F2 then
Blue: drop F4, Green: drop A3 then
Blue: drop E4, Green: drop B3 then
Blue: drop D4, Green: drop C3 then
Blue: drop C4, Green: drop D3 then
Blue: drop B4, Green: drop E3 then
Blue: drop A4, Green: drop F3



G     G     G     G     G     G
G     G     G     G     G     G
B     B     G     G     G     G
B     B     B     G     G     G
B     B     B     B     B     B
B     B     B     B     B     B

Player 1 Total Score: 34
Player 2 Total Score: 38
Total Nodes Expanded by Player 1: 209304
Total Nodes Expanded by Player 2: 172119
Avg Nodes Expanded by Player 1 per move: 11628.0
Avg Nodes Expanded by Player 2 per move: 9562.16666667
Avg Time Spent by Player 1 per move: 0.440111080805
Avg Time Spent by Player 2 per move: 0.407055589888
```

```
On File: ReesesPieces.txt with depth limit: 5 on algorithm: Alphabeta
for Blue/Player 1 and depth limit: 3 on algorithm: Minimax for
Green/Player2

Blue: drop A1, Green: drop F6 then
Blue: drop B1, Green: drop E6 then
Blue: drop C1, Green: drop D6 then
Blue: drop D1, Green: drop C6 then
Blue: drop E1, Green: drop B6 then
Blue: drop F1, Green: drop A6 then
Blue: drop A2, Green: drop F5 then
Blue: drop B2, Green: drop E5 then
Blue: drop C2, Green: drop D5 then
Blue: drop D2, Green: drop C5 then
Blue: drop E2, Green: drop B5 then
Blue: drop F2, Green: drop A5 then
Blue: drop A3, Green: drop F4 then
Blue: drop B3, Green: drop E4 then
Blue: drop C3, Green: drop D4 then
Blue: drop D3, Green: drop C4 then
Blue: drop E3, Green: drop B4 then
Blue: drop F3, Green: drop A4


B     B     B     B     B     B
B     B     B     B     B     B
G     G     G     B     B     B
G     G     G     G     B     B
G     G     G     G     G     G
G     G     G     G     G     G

Player 1 Total Score: 556
Player 2 Total Score: 1097
Total Nodes Expanded by Player 1: 193152
Total Nodes Expanded by Player 2: 186979
Avg Nodes Expanded by Player 1 per move: 10730.6666667
Avg Nodes Expanded by Player 2 per move: 10387.7222222
Avg Time Spent by Player 1 per move: 0.448777768347
Avg Time Spent by Player 2 per move: 0.398055566682
```

```
On File: ReesesPieces.txt with depth limit: 3 on algorithm: Minimax
for Blue/Player 1 and depth limit: 5 on algorithm: Alphabeta for
Green/Player2

Blue: drop F6, Green: drop A1 then
Blue: drop E6, Green: drop B1 then
Blue: drop D6, Green: drop C1 then
Blue: drop C6, Green: drop D1 then
Blue: drop B6, Green: drop E1 then
Blue: drop A6, Green: drop F1 then
Blue: drop F5, Green: drop A2 then
Blue: drop E5, Green: drop B2 then
Blue: drop D5, Green: drop C2 then
Blue: drop C5, Green: drop D2 then
Blue: drop B5, Green: drop E2 then
Blue: drop A5, Green: drop F2 then
Blue: drop F4, Green: drop A3 then
Blue: drop E4, Green: drop B3 then
Blue: drop D4, Green: drop C3 then
Blue: drop C4, Green: drop D3 then
Blue: drop B4, Green: drop E3 then
Blue: drop A4, Green: drop F3



G    G    G    G    G    G
G    G    G    G    G    G
B    B    G    G    G    G
B    B    B    G    G    G
B    B    B    B    B    B
B    B    B    B    B    B

Player 1 Total Score: 1055
Player 2 Total Score: 598
Total Nodes Expanded by Player 1: 209304
Total Nodes Expanded by Player 2: 172119
Avg Nodes Expanded by Player 1 per move: 11628.0
Avg Nodes Expanded by Player 2 per move: 9562.16666667
Avg Time Spent by Player 1 per move: 0.441388871935
Avg Time Spent by Player 2 per move: 0.406722240978
```