CSE4082 - Artificial Intelligence
Muhammed KILIÇ - 150115048
10/12/2018

ENGLISH PEG SOLITAIRE

Read README.md first to run.

In this project we were expected to get a solution path for english peg solitaire by applying 5 different search algorithms with limited time and if necessary limited memory. The game starts with the following board below called INITIAL and reaches to goal state with the following board below called SOLUTION which are located under *item.py* file.

```
INITIAL = np.array([[2, 2, 1, 1, 1, 2, 2],
                    [2, 2, 1, 1, 1, 2, 2],
                    [1, 1, 1, 1, 1, 1, 1],
                    [1, 1, 1, 0, 1, 1, 1],
                    [1, 1, 1, 1, 1, 1, 1],
                    [2, 2, 1, 1, 1, 2, 2],
                    [2, 2, 1, 1, 1, 2, 2]],dtype=np.int)

SOLUTION = np.array([[2, 2, 0, 0, 0, 2, 2],
                     [2, 2, 0, 0, 0, 2, 2],
                     [0, 0, 0, 0, 0, 0, 0],
                     [0, 0, 0, 1, 0, 0, 0],
                     [0, 0, 0, 0, 0, 0, 0],
                     [2, 2, 0, 0, 0, 2, 2],
                     [2, 2, 0, 0, 0, 2, 2]])
```

Searching Algorithms(*search.py*)

a. Breadth First Search:

Current board state is chosen from head of frontier list and new states created from current states are sorted in a sublist according to their *move_value*s and appended to the frontier list. I had to put a memory limit on this because my computer didn't work well after it reached huge memory usage, I couldn't save even move pointer so I put keyboard interrupt option which turns current sub optimal solution and it also stops when memory limit is reached. So I couldn't make it run 1 hour but I put memory limit that I will need the program not to interrupt my computer usage and it runned almost 210 seconds. It had visited 1,347,089 nodes which is not a lot but in the frontier list it had 1,202,132 nodes which makes use of memory. I assume that if I run it 1 complete hour it will just get to 8th depth level and the rest will freeze my computer. The sub optimal board state can be seen below and the rest of outputs are located under *output_files/bfs.txt*.

```
Bread First Search
Time Limit is  3600  seconds.
MEMORY OUT
Sub optimal solution found.
Total Run Time:  210.7193741798401
Frontier Length:  1202132
Node Visited:  1347089
7
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 1, 1, 2, 2],
       [1, 1, 0, 0, 0, 1, 1],
       [1, 1, 1, 1, 0, 0, 1],
       [0, 0, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
6
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 1, 1, 2, 2],
       [1, 1, 0, 0, 0, 1, 1],
       [1, 1, 1, 1, 0, 0, 1],
       [1, 1, 0, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
5
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 0, 0, 1, 1],
       [1, 1, 0, 1, 0, 0, 1],
       [1, 1, 0, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
4
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 0, 0, 0, 1, 1],
       [1, 1, 1, 1, 0, 0, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
```

```
3
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 0, 2, 2],
       [1, 1, 0, 0, 1, 1, 1],
       [1, 1, 1, 1, 1, 0, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
2
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 0, 2, 2],
       [1, 1, 1, 1, 0, 1, 1],
       [1, 1, 1, 1, 1, 0, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
1
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 0, 0, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
0
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 0, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
```

b. Depth First Search:

Depth First Search was easy to implement didn't consume so much memory. New states are appended to frontier list and current node selected was last element of frontier list. It ended up searching after 833 seconds which is almost 14 minutes by visiting 7,667,889 nodes and in the frontier list just 118 nodes left, memory is free which is good. The game board for each move state can be seen in *output_files/dfs.txt*.

```
Depth First Search
Time Limit is 3600  seconds.
Optimum solution found.
Total Run Time:  833.2322733402252
Frontier Length:  118
Node Visited:  7667889
31
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2]])
30
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0],
       [2, 2, 0, 1, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2]])
29
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 1, 0],
       [2, 2, 0, 1, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2]])
28
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0],
       [2, 2, 0, 1, 1, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
27
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
```

```
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0],
       [2, 2, 0, 1, 1, 2, 2],
       [2, 2, 1, 1, 0, 2, 2]])
26
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 1, 1, 0],
       [2, 2, 0, 1, 0, 2, 2],
       [2, 2, 1, 1, 0, 2, 2]])
25
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 1, 0],
       [2, 2, 0, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
24
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 1, 0, 0],
       [2, 2, 0, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
23
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0, 1, 1],
       [2, 2, 0, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
22
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2],
       [0, 0, 0, 0, 1, 0, 0],
```

```
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 1, 1],
       [2, 2, 0, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
21
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 1, 1, 1],
       [2, 2, 0, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
20
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 1, 0, 0, 0],
       [0, 0, 0, 1, 1, 1, 1],
       [2, 2, 0, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
19
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 1, 2, 2],
       [0, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 1, 1, 1, 1],
       [2, 2, 0, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
18
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 1, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 1, 0, 0, 0],
       [0, 0, 1, 1, 1, 1, 1],
       [2, 2, 0, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
17
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 1, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 1, 0, 0, 0],
```

```
       [1, 1, 0, 1, 1, 1, 1],
       [2, 2, 0, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
16
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 1, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
15
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 1, 0, 0],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
14
array([[2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
13
array([[2, 2, 0, 0, 1, 2, 2],
       [2, 2, 1, 0, 1, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
12
array([[2, 2, 0, 0, 1, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
```

```
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
11
array([[2, 2, 0, 0, 1, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 1, 0, 1, 0, 0],
       [0, 0, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
10
array([[2, 2, 0, 0, 1, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 1, 0, 1, 0, 0],
       [1, 1, 0, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
9
array([[2, 2, 1, 0, 1, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [0, 0, 0, 0, 1, 0, 0],
       [1, 1, 0, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
8
array([[2, 2, 1, 0, 1, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 1, 0, 1, 0, 0],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
7
array([[2, 2, 1, 0, 1, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 1, 0, 0, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
```

```
       [2, 2, 1, 1, 1, 2, 2]])
6
array([[2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2],
       [0, 0, 1, 0, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
5
array([[2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2],
       [1, 1, 0, 0, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
4
array([[2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2],
       [1, 0, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
3
array([[2, 2, 0, 1, 1, 2, 2],
       [2, 2, 0, 0, 1, 2, 2],
       [1, 0, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
2
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 0, 1, 2, 2],
       [1, 0, 0, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
```

```
1
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 0, 1, 2, 2],
       [1, 1, 1, 0, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
0
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 0, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
```

c. Iterative Deepening Search:

Iterative Deepening Search will find the solution at the deepest level which means it has to open all nodes of tree before it founds solution. It is almost impossible. So it has reached 8th level in 1 hour. We can see when time ended it had already visited 5,486,310 nodes and 4,917,156 nodes in frontier list.

```
Iterative Deepening Search
Time Limit is  3600  seconds.
IDS For Depth: 0
Time Limit is  3599.9999854564667  seconds.
Depth  0  completed.
Total Run Time:  0.0003364086151123047
Frontier Length:  3
Node Visited:  5
IDS For Depth:  1
Time Limit is  3599.999626159668  seconds.
Depth  1  completed.
Total Run Time:  0.0011720657348632812
Frontier Length:  11
Node Visited:  17
IDS For Depth:  2
Time Limit is  3599.998440027237  seconds.
```

```
Depth  2  completed.
Total Run Time:  0.004576206207275391
Frontier Length:  59
Node Visited:  77
IDS For Depth:  3
Time Limit is  3599.993837118149  seconds.
Depth  3  completed.
Total Run Time:  0.026069164276123047
Frontier Length:  399
Node Visited:  477
IDS For Depth:  4
Time Limit is  3599.967647790909  seconds.
Depth  4  completed.
Total Run Time:  0.17095518112182617
Frontier Length:  2959
Node Visited:  3437
IDS For Depth: 5
Time Limit is  3599.7960329055786  seconds.
Depth  5  completed.
Total Run Time:  1.387758731842041
Frontier Length:  24599
Node Visited:  28037
IDS For Depth:  6
Time Limit is  3598.399619102478  seconds.
Depth  6  completed.
Total Run Time:  13.205155611038208
Frontier Length:  221071
Node Visited:  249109
IDS For Depth:  7
Time Limit is  3585.117059469223  seconds.
Depth  7  completed.
Total Run Time:  520.2798058986664
Frontier Length:  2076743
Node Visited:  2325853
IDS For Depth: 8
Time Limit is  3064.1773521900177  seconds.
TIME IS OUT
Sub optimal solution found.
Sub optimal solution located at depth: 8
Total Run Time:  3235.0432419776917
Frontier Length:  4917156
```

```
Node Visited:  5486310
TIME IS OUT FOR IDS
Sub optimal solution found.
Total Run Time:  3999.27095079422
8
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 0, 1, 2, 2],
       [1, 1, 0, 1, 0, 1, 1],
       [1, 0, 0, 0, 1, 1, 1],
       [1, 1, 1, 0, 0, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
7
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 0, 1, 2, 2],
       [1, 1, 0, 0, 0, 1, 1],
       [1, 0, 0, 1, 1, 1, 1],
       [1, 1, 1, 1, 0, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
6
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 0, 1, 2, 2],
       [1, 1, 0, 0, 0, 1, 1],
       [1, 1, 1, 0, 1, 1, 1],
       [1, 1, 1, 1, 0, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
5
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 0, 1, 2, 2],
       [1, 1, 0, 1, 1, 0, 1],
       [1, 1, 1, 0, 1, 1, 1],
       [1, 1, 1, 1, 0, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
4
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 0, 1, 2, 2],
       [1, 1, 1, 1, 1, 0, 1],
       [1, 1, 0, 0, 1, 1, 1],
```

```
       [1, 1, 1, 1, 0, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
3
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 0, 1, 2, 2],
       [1, 1, 1, 1, 1, 0, 1],
       [1, 1, 1, 1, 0, 1, 1],
       [1, 1, 1, 1, 0, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
2
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 0, 1, 2, 2],
       [1, 1, 1, 1, 0, 0, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
1
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 0, 1, 2, 2],
       [1, 1, 1, 0, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
0
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 0, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
```

d. Depth First Search with Random Selection:

DFS with random selection is almost same with DFS just with a little difference which is I shuffled sublist after creating states from possible moves. As I expected it didn't give me an optimal solution even it had 7 pegs left after 1 hour is ended. I had visited 3,0013,209 nodes and 121 nodes left in frontier when time is up. File can be seen from *output_files/dfs-rand.txt*.

```
Depth First Search with Random Selection
Time Limit is  3600  seconds.
TIME IS OUT
Sub optimal solution found.
Total Run Time:  3600.5181605815887
Frontier Length:  121
Node Visited:  30013209
25
array([[2, 2, 0, 1, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [1, 1, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
24
array([[2, 2, 0, 1, 1, 2, 2],
       [2, 2, 0, 0, 1, 2, 2],
       [1, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
23
array([[2, 2, 0, 1, 1, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [1, 1, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 1, 0, 0],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
22
array([[2, 2, 0, 1, 1, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
```

```
       [0, 1, 0, 0, 1, 0, 0],
       [1, 0, 0, 0, 1, 0, 0],
       [1, 0, 0, 0, 1, 0, 0],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
21
array([[2, 2, 0, 1, 1, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 1, 1, 1, 0, 0],
       [1, 0, 0, 0, 1, 0, 0],
       [1, 0, 0, 0, 1, 0, 0],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
20
array([[2, 2, 0, 1, 1, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 1, 1, 1, 0, 0],
       [1, 0, 0, 0, 1, 0, 0],
       [1, 0, 0, 0, 0, 1, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
19
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [0, 0, 0, 1, 1, 0, 0],
       [1, 0, 0, 0, 1, 0, 0],
       [1, 0, 0, 0, 0, 1, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
18
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 1, 1, 1, 0, 0],
       [1, 0, 1, 0, 1, 0, 0],
       [1, 0, 0, 0, 0, 1, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
17
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [1, 1, 0, 1, 1, 0, 0],
```

```
       [1, 0, 1, 0, 1, 0, 0],
       [1, 0, 0, 0, 0, 1, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
16
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 0, 0, 0, 1, 0, 0],
       [1, 0, 0, 0, 0, 1, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
15
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 0, 0, 0, 1, 0, 0],
       [1, 0, 0, 1, 1, 0, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
14
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 0, 0, 0, 0, 1, 1],
       [1, 0, 0, 1, 1, 0, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
13
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 0, 0, 0, 0, 1, 1],
       [1, 0, 0, 1, 1, 0, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
12
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 0, 0, 0, 0, 1, 1],
```

```
       [1, 1, 1, 0, 1, 0, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
11
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 0, 0, 0, 0, 1, 1],
       [1, 1, 1, 0, 1, 0, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 1, 0, 2, 2]])
10
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 1, 1, 2, 2],
       [1, 1, 1, 1, 0, 1, 1],
       [1, 0, 0, 0, 0, 1, 1],
       [1, 1, 1, 0, 1, 0, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 1, 0, 2, 2]])
9
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 1, 1, 2, 2],
       [1, 1, 1, 1, 0, 1, 1],
       [1, 0, 0, 1, 1, 0, 1],
       [1, 1, 1, 0, 1, 0, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 1, 0, 2, 2]])
8
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 1, 1, 2, 2],
       [1, 1, 1, 1, 0, 1, 1],
       [1, 1, 1, 0, 1, 0, 1],
       [1, 1, 1, 0, 1, 0, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 1, 0, 2, 2]])
7
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 1, 1, 2, 2],
       [1, 1, 1, 0, 0, 1, 1],
       [1, 1, 1, 1, 1, 0, 1],
       [1, 1, 1, 1, 1, 0, 1],
```

```
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 1, 0, 2, 2]])
6
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 0, 1, 1, 2, 2],
       [1, 1, 0, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 0, 1],
       [1, 1, 1, 1, 1, 0, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 1, 0, 2, 2]])
5
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 0, 1, 1, 0, 1],
       [1, 1, 1, 1, 1, 0, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 1, 0, 2, 2]])
4
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 0, 1, 1, 0, 1],
       [1, 1, 1, 1, 0, 0, 1],
       [2, 2, 1, 0, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
3
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 0, 1, 1, 0, 1],
       [1, 1, 1, 0, 1, 1, 1],
       [2, 2, 1, 0, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
2
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 0, 0, 1, 0, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
```

```
       [2, 2, 1, 1, 1, 2, 2]])
1
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 0, 0, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
0
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 0, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
```

e. Depth First Search with a Node Selection Heuristic:

I had my own heuristic which calculates distance of each peg to the center, [3,3] in array notation, and return ratio of total distance between pegs and center to the twice of number of pegs on board. Which means a lot of division and multiplication operations will be evaluated. After writing and running my heuristic I noticed this will make running slower and mostly for each level most of the boards, especially if they are symmetrical, will have same *move_value* which will make searching and selecting correct node harder but if it finds a solution the solution would be memorable.

```
Depth First Search with Special Selection
Time Limit is 1 Hour
TIME IS OUT
Sub optimal solution found.
Total Run Time:  3600.440537214279
Frontier Length:  118
Node Visited:  15589425
29
array([[2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0],
```

```
       [0, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
28
array([[2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 0, 2, 2],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 1, 1, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
27
array([[2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [0, 0, 1, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
26
array([[2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [1, 1, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
25
array([[2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [1, 0, 1, 1, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
24
array([[2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [0, 0, 1, 1, 0, 0, 0],
       [1, 1, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0],
```

```
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
23
array([[2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [1, 1, 0, 1, 0, 0, 0],
       [1, 1, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
22
array([[2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [1, 1, 0, 0, 1, 1, 0],
       [1, 1, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
21
array([[2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2],
       [1, 1, 1, 1, 0, 1, 0],
       [1, 1, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
20
array([[2, 2, 1, 1, 0, 2, 2],
       [2, 2, 1, 1, 0, 2, 2],
       [1, 1, 1, 0, 0, 1, 0],
       [1, 1, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
19
array([[2, 2, 1, 1, 0, 2, 2],
       [2, 2, 1, 1, 0, 2, 2],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 1, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
```

```
       [2, 2, 1, 0, 0, 2, 2]])
18
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 0, 0, 0],
       [1, 1, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
17
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 0, 2, 2],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 1, 0, 0, 1, 0, 0],
       [1, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
16
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 0, 2, 2],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 1, 1, 1, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
15
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 0, 2, 2],
       [1, 1, 1, 1, 0, 0, 0],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 0, 0, 0, 1, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
14
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 1, 1, 1, 0, 0, 0],
       [1, 0, 0, 0, 1, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 0, 2, 2]])
```

```
13
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 1, 1, 1, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0],
       [2, 2, 0, 0, 1, 2, 2],
       [2, 2, 1, 0, 1, 2, 2]])
12
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 0, 0, 0, 1, 0, 0],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 1, 2, 2]])
11
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 1, 1, 1, 1, 0, 0],
       [1, 0, 0, 0, 0, 1, 1],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 1, 2, 2]])
10
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 0],
       [1, 1, 1, 1, 1, 1, 0],
       [1, 0, 0, 0, 0, 0, 1],
       [2, 2, 0, 0, 0, 2, 2],
       [2, 2, 1, 0, 1, 2, 2]])
9
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 0],
       [1, 1, 1, 1, 1, 1, 0],
       [1, 0, 1, 0, 0, 0, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
8
```

```
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 0, 1, 0, 0, 0, 0],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
7
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 0, 0, 1, 1, 0, 0],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
6
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 0, 1, 0, 0],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
5
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 0, 0, 1, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
4
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 0, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 0, 0, 1, 2, 2]])
3
array([[2, 2, 1, 1, 1, 2, 2],
```

```
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 0, 1],
       [2, 2, 1, 0, 0, 2, 2],
       [2, 2, 1, 1, 0, 2, 2]])
2
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 0, 0, 1],
       [2, 2, 1, 0, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
1
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 0, 1, 1, 1],
       [2, 2, 1, 0, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
0
array([[2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2],
       [1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 0, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1],
       [2, 2, 1, 1, 1, 2, 2],
       [2, 2, 1, 1, 1, 2, 2]])
```

Functions

search.py

Breadth First Search:

```
def bfs(cur_node,point_table,time_limit):
```

Depth First Search:

```
def dfs(cur_node,point_table,time_limit):
```

Iterative Deepening Search:

```
def ids(cur_node,point_table,time_limit):
```

Depth First Search with Random Selection:

```
def dfs_rand(cur_node,point_table,time_limit):
```

Depth First Search with Special Selection:

```
def dfs_spec(cur_node,time_limit):
```

mynode.py

MyNode Class

```
class MyNode:
   def __init__(self, board, parent,depth_level,peg_number):
       self.parent = parent
       self.board = board
       self.depth_level = depth_level
       self.peg_number = peg_number
```

board.py

Except Board class it had make_moves function with given peg coordinate and direction and return a new board:

```
def make_move(board_array_param,x,y,direction)
```

heuristic.py

Manhattan Distance implemented here:

```
def man_dist(board_array):
```

item.py

all small calculation functions and variables are located here.