**Report for Assignment I Submission**

**Test Set:**

1. **--0XX--,--X0X--,0000X00,00XX0X0,0000000,--0XX--,--XX0--**
2. **--000--,--0X0--,00XXX00,000X000,000X000,--000--,--000--**

**Iterative Deepening(dfs\_i.py):**

**Execution Strategy:**

C:\Python34\python.exe C:/Users/srinath/Downloads/ai-2015-02-27/ai/dfs\_i.py

**Enter Board State:**

--0XX--,--X0X--,0000X00,00XX0X0,0000000,--0XX--,--XX0--

**Enter Maximum Iterative Depth:**

12

['(2,0) (0,8) (5,17) (8,22) (17,15) (15,27) (30,32) (32,24) (27,29) (29,17) (18,16)']

**Statistics:**

**level: 12**

**No. of Nodes Expanded: 953082**

**Elapsed Time: 19.440112113952637 secs**

**Memory Size=10.4 KiB**

[ Top 10 ]

<frozen importlib.\_bootstrap>:321: size=10.4 KiB, count=71, average=150 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/dfs\_i.py:129: size=5480 B, count=80, average=68 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/dfs\_i.py:226: size=3264 B, count=6, average=544 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/dfs\_i.py:21: size=2912 B, count=22, average=132 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/dfs\_i.py:159: size=2176 B, count=4, average=544 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/dfs\_i.py:82: size=1704 B, count=2, average=852 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/dfs\_i.py:181: size=1632 B, count=3, average=544 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/dfs\_i.py:246: size=544 B, count=1, average=544 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/dfs\_i.py:240: size=512 B, count=1, average=512 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/dfs\_i.py:192: size=464 B, count=1, average=464 B

**Heuristic II(h2.py):** In this heuristic, we use the fact that the move that makes number of isolated(which does not have any neighbors) pegs less will help us arrive at solution quicker. The code for these is implemented in such a way that, first a move is made corresponding to position (i,j) and number of isolated pegs are counted. Then the move that corresponds to least number of isolated pegs is used to expand our fringe. The code is implemented in **getBestPegH2** function. Every call to this function gives the next best element (one which has least number of isolated pegs) to the caller.

**Execution Strategy:**

C:\Python34\python.exe C:/Users/srinath/Downloads/ai-2015-02-27/ai/h2.py

**Enter Board State:**

--0XX--,--X0X--,0000X00,00XX0X0,0000000,--0XX--,--XX0--

['(2,0) (0,8) (31,23) (5,17) (8,22) (17,15) (15,27) (30,22) (22,24) (29,17) (18,16)']

**Statistics:**

**No. of Nodes Expanded: 10536**

**Elapsed Time: 3.8892228603363037 secs**

**Memory Size=10.4 KiB**

['(2,0) (30,32) (32,24) (5,17) (0,8) (17,29) (29,27) (8,22) (27,15) (15,17) (18,16) ']

[ Top 10 ]

<frozen importlib.\_bootstrap>:321: size=10.4 KiB, count=71, average=150 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h2.py:231: size=5672 B, count=83, average=68 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h2.py:285: size=4224 B, count=8, average=528 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h2.py:255: size=3696 B, count=7, average=528 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h2.py:275: size=3168 B, count=6, average=528 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h2.py:23: size=2912 B, count=22, average=132 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h2.py:145: size=1856 B, count=4, average=464 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h2.py:143: size=1824 B, count=4, average=456 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h2.py:146: size=1704 B, count=20, average=85 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h2.py:81: size=1704 B, count=2, average=852 B

**Heuristic I(h1.py):** In this heuristic, we use the fact that moving pegs in the outer edges first towards the center, brings us closer to the solution. So all the pegs in the sides are chosen first to move and then pegs in the center. The heuristic is implemented in the function **getBestPegH1(board\_state)** which on successive calls will return the pegs in the board state from edges.

**Execution Strategy:**

C:\Python34\python.exe C:/Users/srinath/Downloads/ai-2015-02-27/ai/h1.py

**Enter Board State:**

--0XX--,--X0X--,0000X00,00XX0X0,0000000,--0XX--,--XX0--

['(2,0) (0,8) (31,23) (5,17) (8,22) (17,15) (15,27) (30,22) (22,24) (29,17) (18,16)']

**Statistics:**

**No. of Nodes Expanded: 34**

**Elapsed Time: 2.866495084762573 secs**

**Memory Size=10.4 KiB**

[ Top 10 ]

<frozen importlib.\_bootstrap>:321: size=10.4 KiB, count=71, average=150 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h1.py:137: size=6120 B, count=78, average=78 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h1.py:23: size=2912 B, count=22, average=132 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h1.py:162: size=2592 B, count=4, average=648 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h1.py:163: size=2080 B, count=4, average=520 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h1.py:225: size=1944 B, count=3, average=648 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h1.py:183: size=1944 B, count=3, average=648 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h1.py:79: size=1704 B, count=2, average=852 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h1.py:226: size=1560 B, count=3, average=520 B

C:/Users/srinath/Downloads/ai-2015-02-27/ai/h1.py:184: size=1560 B, count=3, average=520 B

**Statistics:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Algorithm/**  **property** | **Iterative Deepening** | | | **Heuristic I** | | | **Heuristic II** | | |
| **Test 1** | **Test 2** | | **Test 1** | **Test 2** | | **Test 1** | **Test 2** | |
| **Nodes Expanded** | **953082** |  | | **34** |  | | **10536** |  | |
| **Time** | **19 sec** | |  | **3sec** | |  | **4sec** | |  |
| **Memory** | **10.KiB** | | **10.4KiB** | **10.4KiB** | | **10.4KiB** | **10.4KiB** | | **10.4KiB** |

**Conclusions:**

**As we can see that second heuristic performs better than iterative deepening but does not necessarily perform better than first heuristic. This is because the order in which nodes are expanded.**

**As number of isolated pegs although a global measure, it does not give necessarily capture the distance to the final solution accurately as the isolated pegs can increase and decrease. SO Heuristic I is better which captures global measure of distance to the final solution by reducing the number of pegs on the edges by moving them towards the center.**

**Execution:**

1. The folder contains two programs:
   1. dfs\_i.py
   2. h1.py
   3. h2.py
2. Each of the programs are self-contained, in the sense, they can be executed individually.

To execute the code:

Ex: python3 dfs\_i.py

1. The programs are interactive and will prompt for input when required. Both the programs will take input for the board as specified in the assignment.

Ex: python3 dfs\_i.py

Enter Board State:

--0XX--,--X0X--,0000X00,00XX0X0,0000000,--0XX--,--XX0---

1. As the code executes, it will generate solution with path or No path found with statistics below it.