CSE 537- ARTIFICIAL INTELLIGENCE
PROJECT 1 REPORT

Following Files are submitted:

PegSolitaire_IDS.py: Source Code for Peg Solitaire using Iterative Deepening Search technique.
PegSolitaire_A*.py : Source Code for Peg Solitaire using A* - heuristic Average Distance between all pegs.
PegSolitaireA*_IsolatedPegs.py: Source Code for Peg Solitaire using A* - heuristic Number of Isolated pegs.
Test_Sets: Sample boards to test the code.
ExecutionTraces

CODE DESCRIPTION FOR PEG SOLITAIRE:

I. USING ITERATIVE DEEPENING SEARCH:

Execution: To run in IDE , right click on the PegSolitaire_IDS.py file and the select Run

Logic Used:

A stack is used as fringe list. The ***idfs*** will increment the depth to search for the goal at each step and calls the ***dfs*** function. If the goal is reached ***idfs*** will return instead of incrementing the depth limit.
idfs function will push the initial state into the stack before it calls the ***dfs,*** so that each time the search will start from the root.
The ***dfs*** will pop the state on the top of the fringe list and check if it is the goal state. If so, it will return. Else, only if the depth of the tree is not equal to the limit, children for the current state are generated. If the goal is not reached but the depth limit is reached, appropriate value is returned.

Sample Input:

```
--000--
--0X0--
00XXX00
000X000
000X000
--000--
--000--
```

Output:

```
------Iterative Deepening Search------
Enter the initial state:

--000--
--0X0--
00XXX00
000X000
000X000
--000--
--000--
Moves made :
  10 to 8 , 24 to 10 , 11 to 9 , 8 to 10 , 5 to 17
The goal is reached
Number of Nodes Explored: 88
```

Running time: 1.69106078148 seconds

USING A*: HEURISTIC – AVERAGE DISTANCE BETWEEN ALL THE PEGS ON THE BOARD:

Execution: To run in IDE , right click on the PegSolitaire_A*.py file and the select Run.

Logic Used:

Priority queue is used as a fringe list. The ***aStarSearch*** function searches for the goal. The uniformed cost g(n) is estimated to be length of the path from the initial state to the current state, which is the depth of the node where each link has a uniform cost of 1. The heuristic used is the average distance between all the pegs in the current state.

Sample Input:

--000--
--0X0--
00XXX00
000X000
000X000
--000--
--000--

Output:

------A* Algorithm (Heuristic: Average distance)------
Enter the initial state:

--000--
--0X0--
00XXX00
000X000
000X000
--000--
--000--
The goal is reached.
Number of Nodes Explored: 6
Running time: 9.09493684769

USING A*: HEURISTIC – NUMBER OF ISOLATED NODES:

Execution: To run in IDE , right click on the PegSolitaire_IsolatedPegs*.py file and the select Run.

Logic Used:

Priority queue is used as a fringe list. The ***aStarSearch*** function searches for the goal. The uniformed cost g(n) is estimated to be length of the path from the initial state to the current state, which is the depth of the node where each link has a uniform cost of 1. The heuristic used is the number of isolated pegs on the board. A peg is said to be isolated if there no pegs in its immediate positions.

Sample Input:
--000--
--0X0--
00XXX00
000X000
000X000
--000--
--000--

Output:
------A* Algorithm (Heuristic: Isolated Pegs)------

Enter the initial state:
--000--
--0X0--
00XXX00
000X000
000X000
--000--
--000--
The goal is reached
Number of Nodes Explored: 17
Running time: 6.92548799515

OBSERVATIONS:
1. From the above sample outputs of each technique on the same input board, it is clear that A* algorithm performs better as the number of nodes explored is very less (6 and 17) when compared to Iterative Deepening Search (88)

2. The A* algorithm using heuristic “Average Distance Between the Pegs ” explored only 6 nodes but the one using heuristic “ Isolated Pegs” explored 17 nodes. So, the first one is a more informed heuristic.

If there are more number of nodes on the board like below:

--XXX--
--XXX--
00XXX00
00X0X00
0000000
--000--
--000--

Isolate Pegs heuristic explored 8752 before reaching the goal but Average Distance heuristic explored only 120 nodes, which clearly shows that the latter heuristic is far better than the former one.

3. As seen, the execution times for A* is more than IDS because of the heuristic calculation.