Assignment 1 – Due September 24, 2015
100 points

This assignment covers the topic of AI search. It will reinforce your understanding of this topic. In the assignment you will write a program to find a solution to the game of Peg Solitaire. A description of this board game follows:

The board consists of peg holders. A peg holder can either be empty or hold exactly one peg. In Figure 1 below "○" and "●" correspond to an empty and a filled peg holder respectively. The other blank spaces in the board that are neither empty or filled are unusable - these are the 2 x 2 squares on the four corners of the board. The objective of the game is to remove all except one peg from the board. The rule for removing a peg is this: If X, Y and Z are three consecutive peg holders with X and Y holding a peg each and Z empty then the peg in X can jump over Y and fill Z. In the process Y is removed from the board. The peg holders X and Y become empty and Z now holds a peg. Note that only horizontal and vertical moves are allowed.

There are several variants of the game with differing board sizes and shapes. For this assignment we will use the 7 x 7 board as shown in Figure 1 below with the 2 x 2 squares on the four corners unused, i.e. they are not peg holders and hence unusable by our definition. In our game the objective is to remove all the pegs in the board except one and this peg should be placed in the center peg holder - see Figure 1(b).

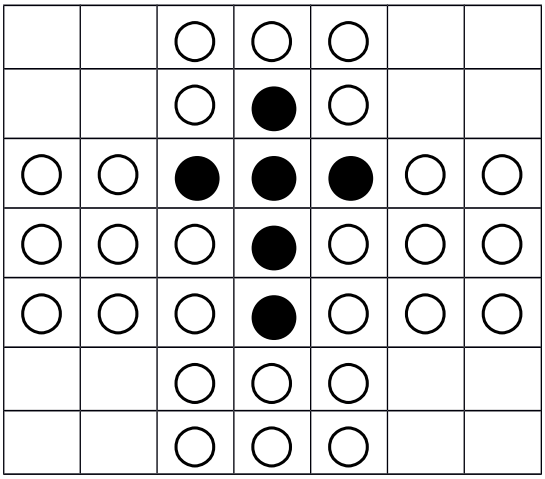

Figure 1 (a)

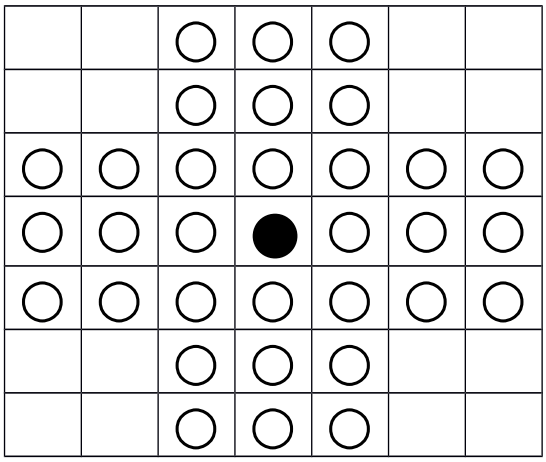

Figure 1(b)

The board's configuration can be represented by a string characters, one string per row. The configuration in Figure 1(a) above will be represented as:

< - - 0 0 0 - -, - - 0 X 0 - - , 0 0 X X X 0 0, 0 0 0 X 0 0 0, 0 0 0 X 0 0 0, - - 0 0 0 - -, - - 0 0 0 - ->

0 and X denote an empty and filled peg holder respectively and – denotes an unusable peg holder.

You can number the peg holders as shown in Figure 2 below:

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)
(1,0)	.	.	.	.	.	.
(2,0)	.	.	.	.	.	.
(3,0)	.	.	.	.	.	.
(4,0)	.	.	.	.	.	.
(5,0)	.	.	.	.	.	.
(6,0)	.	.	.	.	.	(6,6)

Figure 2

You are required to implement AI search algorithms to solve the peg solitaire game as specified above. Your program will take a given initial configuration of the board and will output a sequence of valid moves that will result in a solution, if one exists. A valid move is one that results in removing a peg. A move will be a pair (X,Y) where the peg in peg holder numbered X is moved to the peg holder numbered Y. For the initial board configuration in Figure 1(a) the following sequence of valid moves will result in the configuration in Figure 1(b) which is also the solution to the game with Figure 1(a) as the initial configuration:

1. Iterative Deepening Search (uninformed)
2. A* with two kinds of heuristics, one more informed than the other

Coding Details:

config.py: This is where all the details of the students goes into, Open and Read the file to know how to fill the details.

game.txt: This contains the initial setup of the game.

PegSolitaire.py: Command line Parsing and appropriate function calling is done here, You should NOT change anything in this file.

PegSolitaireUtils.py: This is where real assignment starts, You have to implement four functions here. Open and read the file for details.

ReadGame.py: you don't have to change anything in here. It just reads data from game.txt and make initial game setup.

Search.py: This file contains different searching algorithms which you have to implement.

Running Code:

code can be run by first going to the directory Homework1 and using command **python pegSolitaire.py –-input ./game.txt –-flag 0**

you can change name of the file ./game.txt with whatever configuration you are using.

Flag is used to run the specific searching algorithms:

0 will run all of them , 1 will run only iterative deepening, 2 will run A star with heuristic one and 3 will run A star with your second heuristic.

Example: for the figure 1(a) like setup you would see output like:

Itr Deepening Search:

Execution Time: 12.000

Nodes Expanded: 1234

Trace: [(2,3), (2,5), (4,3), (2,3), (2,2), (2,4), (2,5), (2, 3), (1,3), (3, 3)]

Astar One Search:

Execution Time: 8.110

Nodes Expanded: 123

Trace: [(2,3), (2,5), (4,3), (2,3), (2,2), (2,4), (2,5), (2, 3), (1,3), (3, 3)]

Astar Two Search:

Execution Time: 9.800

Nodes Expanded: 0120

Trace: [(2,3), (2,5), (4,3), (2,3), (2,2), (2,4), (2,5), (2, 3), (1,3), (3, 3)]

NOTES:

1. You MUST save the trace in a list inside pegSolitaireObject.trace for each search strategy to get grades.

2. You MUST call getNextState(self,oldPos, direction) function for expanding the search tree to get grades, Otherwise there is no way to see how many nodes you are expanding.

3. You must describe your heuristics used using comments on code.

4. config.py is very important file you should update to get grades, if you fill it incorrect then there is no other way to recognize your program's identity and you'll be penalize accordingly.

5. You have to use python 2.7, you can use future import library though.

6. Changes should be made only in specified locations in the file, restricting the natural flow of execution of the program results in penalty. It'll automatically displays time your program used, nodes expanded and trace of the answer.

7. One setup of problem can have many answers, we need only the first trace you get. DONT save all the traces in pegSolitaireObject.trace just one will do.

8. For more information of pegSolitaire you can use wiki.

9. Points Division(100 Max):

    a) (5+5): Comments on code and filling config.py correctly

    b) (20): Correct execution on given game setting

    c) (50): Correct execution on (NOT given) tricky game settings.

    d) (20): Demo/class quiz for assignment 1. You'll be informed.

10. Extra Points: (25 Max):

    a) if your program uses tricks to prune searches and not revisiting expanded nodes.

Submission:
1. On or before midnight of the due date you should submit through BLACKBOARD (NO e-mail), zip file of the modified Homework1 folder.
2. You can work in pairs of two, EITHER one of the two can send the assignment it doesn't matter who sends it as long as you have updated config.py file right.