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).

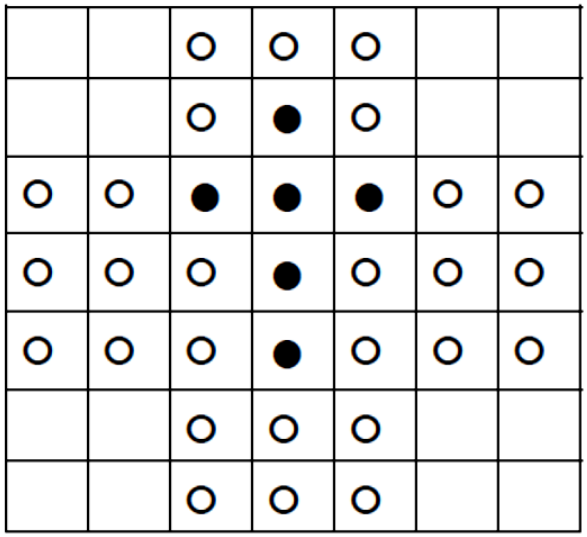

Figure 1 (a)

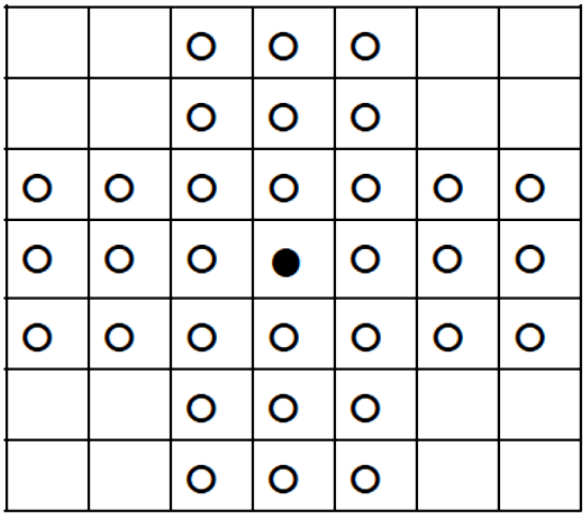

Figure 1(b)

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:

Figure 2

		0	1	2		
		3	4	5		
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
		27	28	29		
		30	31	32		

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: <(9,7),(23,9),(10,8),(7,9),(4,16)>.

You will use two search strategies:

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

For each search strategy – IDS and A* with the two heuristics - your program should generate the number of nodes expanded, memory usage and running time.

Extra points (up to a maximum of 25) will be given if your program uses tricks to prune searches and not revisiting expanded nodes.

Notes:

1. Your program should be written in Python.
2. For more information about peg solitaire see: http://en.wikipedia.org/wiki/Peg_solitaire

Submission:

On or before due date you should email to the TA a zip file containing:

• Source code with good documentation
• A trace of the execution for each search strategy

• Test set
• A report with the required statistics generated

You have until midnight of the due date to email the zip file. You should sign up for a demo with the TA. On the demo date you will be given your source files that you should compile and demo to the TA.

Due Date: February 28