CS230 EMA (Graph Applications in AI) Assignment

Assignment Due LDoC 04/24

1 Gradescope Assignment

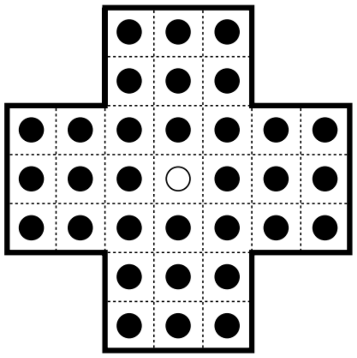

Figure 1: The classic/standard Peg Solitaire board.

Peg Solitaire is a single-player board game with many variations but a rather simple game mechanism. The game starts with a board (see Figure 1 for the standard variation) with n holes and $n-1$ pegs (so only 1 of the n holes is empty). A *valid* move is to jump a peg either vertically or horizontally over an adjacent peg into a hole two positions away. In doing this move, we remove the peg being jumped over. In other words, each valid move removes one peg from the game. The goal is to keep making valid moves until there is exactly one peg left *at the originally empty position.*[1]

You can familiarize yourself with the game mechanics here; that website has the standard version (called *English*) and another version of the board (called *French*). Many other versions exist.

Consider a general version of the game with n holes and $n-1$ pegs (and a known board configuration). Throughout the assignment, you can use asymptotic notations/combinatorial symbols without simplification.

1. (State Space Graph Modeling.) We will now model the game as a state space graph.
   (a) What would the states be? In other words, what information do we need to specify to capture a state?
   (b) How many states are there in total? *Note that we are considering a general version of the graph, not just Fig. 1.*
   (c) How many possible valid moves can exist for a given state? In other words, what is the maximum degree of the state space graph? Try obtain an upper bound. *Hint: you may need to make your own assumptions about what the board can look like. It is supposed to be an estimation, not a precise count.*
2. (Search Algorithms.)
   (a) If the game is solvable (note that a general board configuration may not be solvable), which level is the goal state located in the search tree? Briefly justify your answer.
   (b) Suppose we apply **BFS Tree Search** on this state space graph. Is the search algorithm guaranteed to terminate? If it does, how many vertices *at most* would we have explored before either finding a solution or declaring failure?
   (c) Suppose we apply **DFS Tree Search** on this state space graph. Is the search algorithm guaranteed to terminate? If it does, how many vertices *at most* would we have explored before either finding a solution or declaring failure?
   (d) Suppose we apply **any Graph Search** on this state space graph. Is the search algorithm guaranteed to terminate? If it does, how many vertices *at most* would we have explored before either finding a solution or declaring failure?
3. (Pruning.) For large values of n, the search complexity of this problem is likely prohibitive. What would be a useful *pruning strategy* that we can leverage so that we can elimiate (i.e., not expand on) *clear dead-ends*, i.e., states that can never lead to a solution? Briefly describe (and justify) your strategy.

[1]Some variants of the game only requires there is exactly one peg left, but it can be at any position.