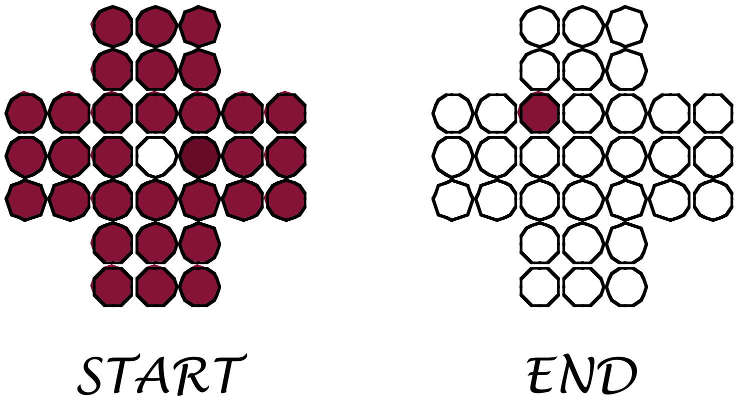
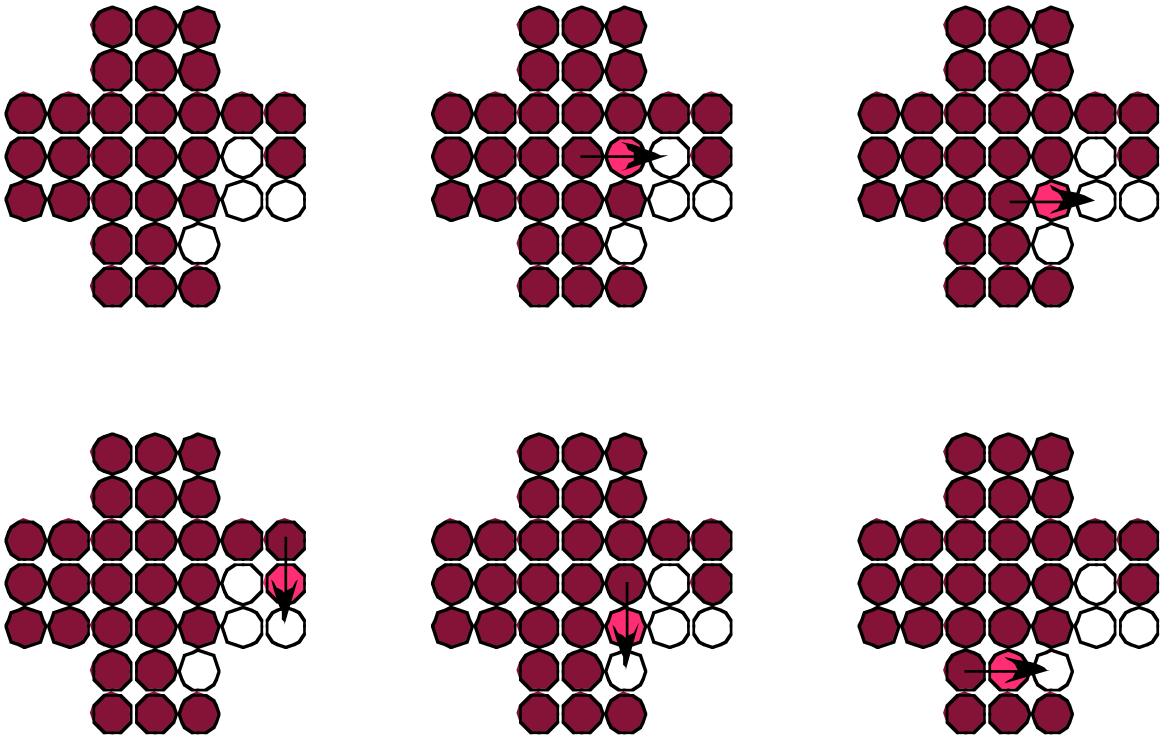
Computer Science 308-250B Homework #4
Due Monday March 22, 2004, 13:30

•**1)** A "**HI-Q**" board is an array of 33 holes, organized in 7 rows, 4 of which contain 3 holes each and the middle 3 rows contain 7 holes each. Originally, 32 out of 33 holes are filled with red pegs, except the very middle hole which is empty. See below:

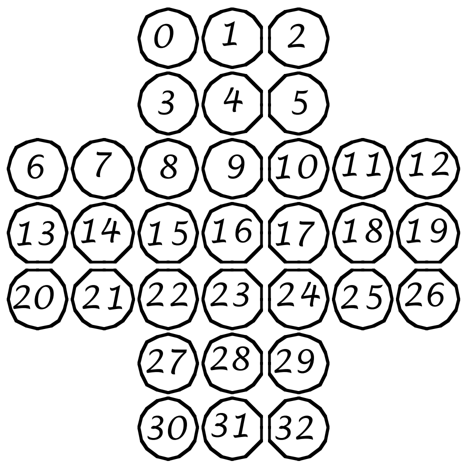


The rules of this puzzle is that at any point in time a peg may be removed by jumping over it from a neighboring peg to a neighboring hole, either horizontally or vertically. The goal of the game is to end with a single red peg on the board.



A weight 29 configuration
and all 5 valid sucessor s

We recommend that you use a tree data structure to represent each configuration of the HI-Q board. The children of a configuration should be the possible configurations that can be reached by jumping over a single ped. Your algorithm should traverse the tree so to locate one of the "solved" configurations. Once the solution is found, your algorithm should return a **String** containing the sequence of jumps necessary to solve the puzzle. We will use the following notation to represent moves on the HI-Q board.



For instance "8->10" means peg at position 8 jumps over position 9 and lands on position 10. So your output should be something like : "18->16, 5->17,….". We will provide you with an Object type "HIQ" that you must use for uniformity reasons. This object will contain methods for testing if the puzzled is solved, inputting a board configuration and outputting a board configuration. This will be available through the course web page soon.

In order to make your program (space) efficient, you must implement some sort of memory management policy to avoid overflowing the virtual machine memory carelessly. You will be evaluated on the ability of your program to solve various puzzles: maybe your program cannot find a solution from the START configuration, but maybe it can find a solution for a configuration with only 20 pegs on it, or only 10 pegs on it,… Examples on various configurations to be solved will be provided by us on the course web page as well. The more complex puzzles you can solve, the better grade you will get.

Note: if you choose to solve this problem *without explicitly exploring a tree*, you must explain where a *virtual* tree is being explored and *how* it is explored (DFS, BFS, etc).