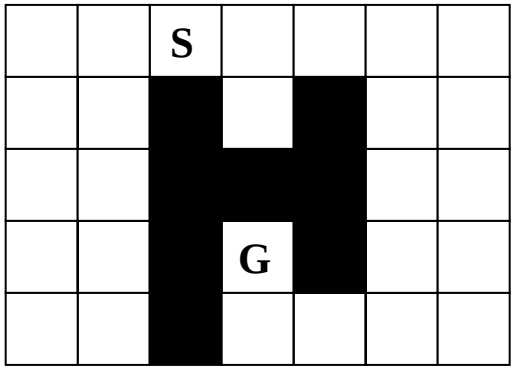
Assignment #2
Artificial Intelligence - CSCE 523
Due: 8:00 AM, Wednesday February 6, 2012
Search and Game Tree Search

Turnin: E-mail me a zip file containing your typed solution to questions 1 and 2, and your program for question 3.

1. For the following maze, show the lists of open and visited nodes (with their associated costs) for each cycle of the listed search algorithms. The start cell is *S*, and the goal cell is *G*. The agent can move north, south, east, and west. The agent expends 1 point moving south or west, and 2 points moving north or east.
    a. Decide on a heuristic estimator function and write the function out.
    b. Decide on a method to break ties (label all cells with letters and break alphabetically, first come first served, etc.) and write that out

    Then perform the following searches on the space:
    a. Beam search with a beam size of 2
    b. IDA* search



2. For the following Light-Up puzzle, show the sequence of variable assignments during backtracking with forward checking; examine cells in alphabetical order. Show assignments by writing the forward checking table process (14 columns: step number, a value, b value, ..., l value, backtrack{list the constraint violation that causes the backtrack}).

a	*b*	*c*	*d*
e	**1**	**2**	*f*
g		**0**	*h*
i	*j*	*k*	*l*

3. Below are the rules for the game Lines of Action. For this part of the assignment you are to write a board evaluator, and alpha-beta minimax search for the game. I have provided Java files to maintain the board state, to test the legality of a move, and to generate a list of possible moves as a Vector for one or all of the pieces in the class directory. You are responsible for completing the h_value(), and the min_max() methods. Do not feel constrained by my code, if you feel that you need additional elements or different functionality feel free to change it, but be sure to document the changes.

   Line of Action Rules:
   1. The black pieces are placed in two rows along the top and bottom of the board, while the white pieces are placed in two files at the left and right side of the board (Figure 1).
   2. The players alternately move, starting with Black.
   3. A player to move must move one of its pieces. A move takes place in a straight line (up, down, left, right, and all four diagonals), exactly as many squares as there are pieces of either color anywhere along the line of movement (These are the Lines of Action).
   4. A player may jump over its own pieces.
   5. A player may not jump over the opponent's pieces, but can capture them by landing on them.
   6. To win a player must move all their pieces on the board into one connected unit. The first player to do so is the winner. The connections within the group may be either orthogonal or diagonal. For example, in Figure 2 Black has won because the black pieces form one connected unit.

Figure 1: Starting board.

   7. If one player's pieces are reduced by captures to a single piece, the game is a win for this player.
   8. If a move simultaneously creates a single connected unit for both players, the player that moved wins.

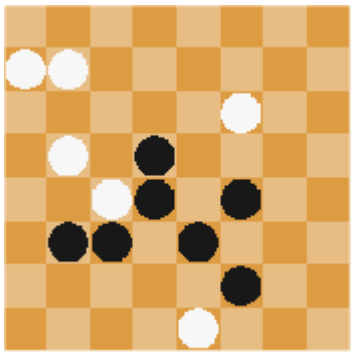


Figure 2: Black wins.

Additional Resources:
There are several articles on Lines of Action in the course directory under the subdirectory Articles. Some other links on the game:

Lines of Action home page
http://boardspace.net/loa/
U of A GAMES Group Home Page (YL and MONA)
http://webdocs.cs.ualberta.ca/~darse/LOA/
Mark Winands (MIA)
http://www.personeel.unimaas.nl/m-winands/loa/index.htm

Turn-Ins:
Turn in should include your code, instructions for compilation, and a write-up describing your implementation, heuristic details, and experiences.

Stipulations:
You are responsible for having your search halt at the set threshold depth, pruning the correct amount of space. For the threshold depth, note that one step is one players move whether it is your move or your opponents, and not your programs and your opponents countermove.