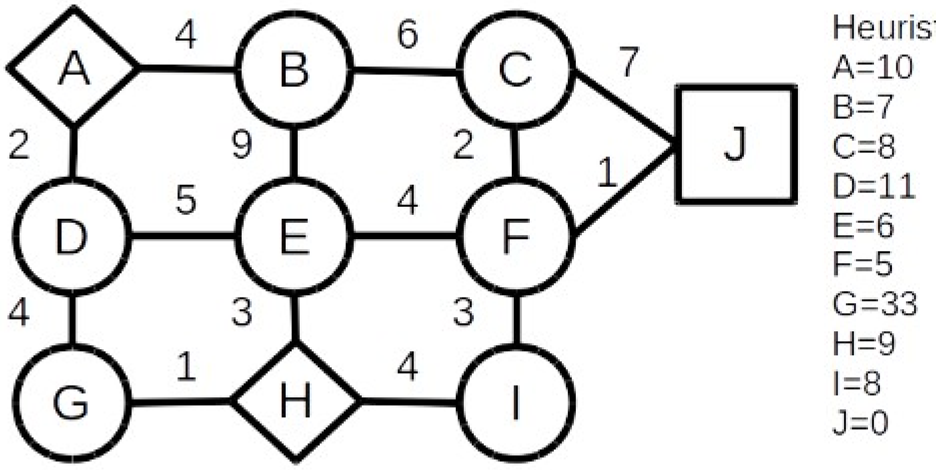
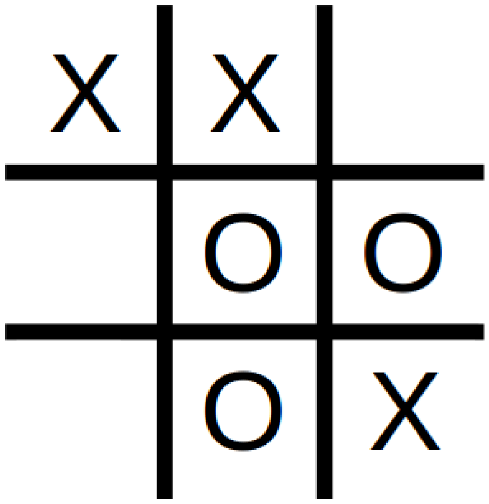
4511W, Fall-2019
ASSIGNMENT 3 :
Assigned: 10/15/19 Due: 10/22/19 at 11:55 PM (submit via Canvas, you may take a picture of handwritten solutions, but you must put them in a pdf) Submit only pdf or txt files

Written/drawn:

Problem 1. (10 points)
Below is a graph with edge costs and heuristic values. Perform local beam search (initial nodes "A" and "H" and goal node "J") with 2 beams and show enough step-by-step work to be clear on your process.



Problem 2. (15 points)
Consider the following tic-tac-toe configuration (player "X" to play). Draw a tree with max and min nodes starting from the shown state and then use the minimax algorithm to find how you should play (you need to actually run the algorithm, not just state the obvious solution). (Note: on homework 1 the code I had you run built a tree like this for tic-tac-toe and solved with alpha-beta pruning.)



Problem 3. (15 points)
Assume you are running Monte-Carlo tree search on a binary tree (i.e. branching factor of 2 everywhere). Also assume on ties you pick the left-most tied node. You see the following sequence of wins/losses from the "random rollout":

Win, Win, Loss, Loss, Win, Win, Loss, Win, Win

Show the tree and all relevant UCB (upper confidence bound) values for making the next decision on which node to pick at the indicated (i.e. numbered) points in running the algorithm (same sequence of win/loss as above):
Win, Win, Loss, (1), Loss, Win, (2), Win, Loss, Win, Win, (3)

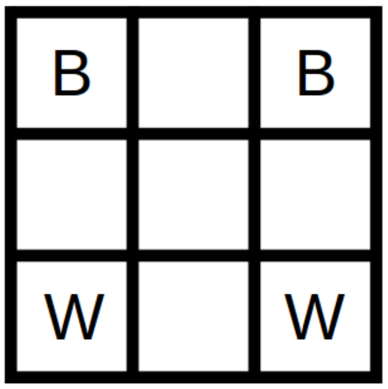
Problem 4. (20 points)
Suppose you have a depth 3 tree, but you can order the actions differently. At one depth you need a branching factor of 6, at another depth a factor of 4 and the last depth a branching factor of 2. Based on this tree setup, answer the following questions:

(1) Give a full tree with specific values at terminal states for the tree with branching factors 2, 4, and 6 at depths 1, 2 and 3, respectively (i.e. the root has a branching factor of 2, then each of these two children have a branching factor of 4...). Find what specific values let you **<u>prune the most nodes</u>** on the tree when running alpha-beta pruning and demonstrate which ones can be pruned.

(2) For each of the 5 remaining combinations (i.e. 2-6-4, 4-2-6, 4-6-2, 6-2-4, 6-4-2) of branching factors per depth, write out the maximum number of nodes that can be pruned. (You do not need to show the trees or which nodes in this case, just the numbers.)

Problem 5. (25 points)
To find a heuristic for each of the following situations, (1) state how you would relax (i.e. make easier) the problem, (2) how you would solve this relaxed problem optimally and (3) a description of how you would find the heuristic quickly (can be a formula or process, though not a recursive search (i.e. exponential))

Situation 1: Suppose you have a 3x3 board with black and white "knights" (from chess) positions as shown below. The goal is to have both white knights on top and black knights on the bottom (left or right doesn't matter). (The movement is restricted to normal "knight" movement in chess: an L shape.)



Situation 2: Four people want to cross a dangerous bridge (with holes) at night, but there is only one flashlight and it only provides enough room for one or two people to cross safely. Person A takes 1 minute to cross, person B takes 2 minutes, C takes 3 and D takes 4. The slowest person determines the overall speed of the crossing. The goal is to get everyone across in as little time as possible. (Note: the

“optimal” answer is not 2+1+3+1+4=11, though finding the optimal is not part of the required answer.)

Situation 3: You need to organize the delivery of physical letters to 10 different houses using 3 trucks. You want to do so using the least fuel possible. You may assume you have a table telling you how much fuel it takes to go between any pair of houses (or go from the starting point to any house or the other way around as it might not be symmetric such as an uphill). The 3 trucks must end at the starting point (the depot) at the end.

Programming (python/lisp):

Problem 6. (15 points)
Use the given genetic algorithm in search.py and apply it to the n-queens problem. Analyze this on a number of different configuration to answer the following questions:
(1) Is the genetic algorithm good in general to solve n-queens? Why or why not?
(2) What is the purpose of f_ thresh and how did you handle/use it in your testing?
(3) What is one positive thing about the genetic algorithm in general? One negative aspect?