Exam 1 - Takehome

CSCI 5511: Artificial Intelligence

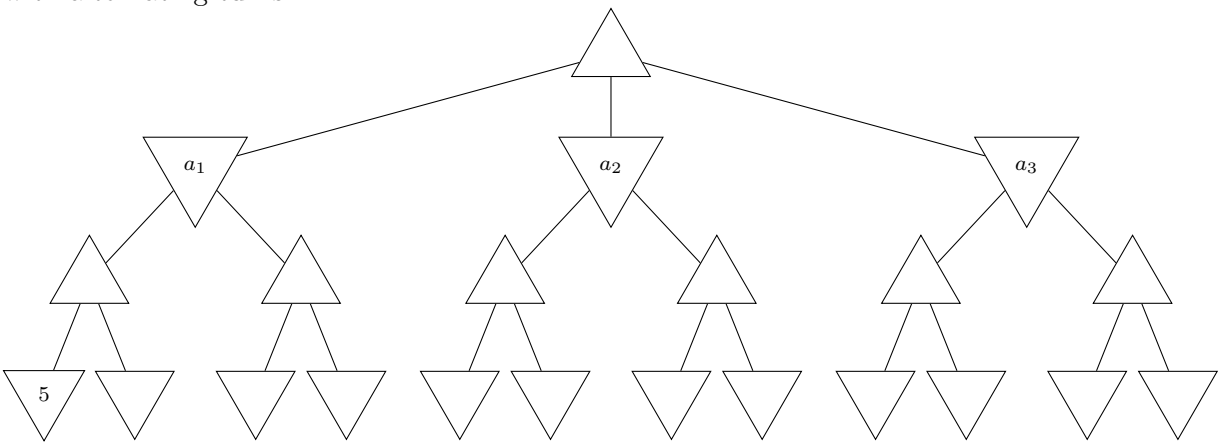
Fall 2021

Due November 6, 11:59 pm

Read all of the following information before starting the exam:

- Show all work, clearly and in order, if you want to get full credit. I reserve the right to take off points if I cannot see how you arrived at your answer (even if your final answer is correct).
- This exam is open-book and open-notes.
- Please keep your written answers brief; be clear and to the point. I reserve the right to take points off for rambling and for incorrect or irrelevant statements.
- To turn this in, either print, write and scan your answers or use any appropriate program to create a nicely formatted document. Convert it to a PDF and submit via Canvas.
- This exam has 3 problems and is worth 24 points.
- Good luck!

1. Adversarial Search (*8 points*) Consider the following game tree for a two-player game with alternating turns:



a. (*4 pts*) Write utility values for the rest of the leaf nodes so that alpha-beta pruning will prune the *most* possible nodes. Also indicate which nodes are pruned.

b. (*2 pts*) What is the ideal way for alpha-beta pruning to order nodes so that it can prune the most things from the tree? Consider ordering for children of both Max and Min nodes.

2. Constraint-Satisfaction Problems (*8 points*)

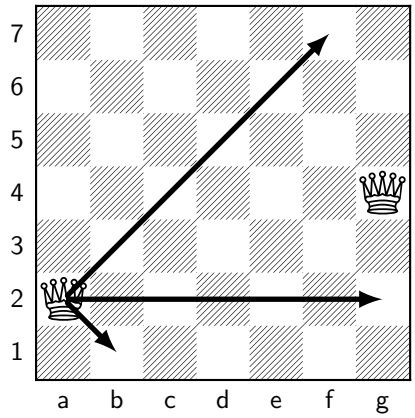
Consider the N-Queens problem with $N = 7$.

In the N-Queens problem, each variable is a Queen that we need to place is associated with a specific column of the board. Call them $\{Q_a, Q_b, ...\}$.

Each variable then has a domain that corresponds to the possible rows of the board on which the queen could be placed. So at the beginning of the problem, each of these domains is $\{1, 2, ..., N\}$.

Furthermore, there is a binary constraint between each pair of variables. (Because each queen will attack squares that others could be on.)

Suppose that our backtracking search has already chosen the following two queen placements for Q_a and Q_g:



The arrows on the board are intended to remind you how a queen attacks on a chessboard. All subsequently placed queens attack in this manner as well.

a. (*2 pts*) Assuming that we use the *most constrained variable* heuristic for selecting the next variable, which column should backtracking search choose next? If there is a tie, choose the leftmost column.

b. (*2 pts*) Suppose that my code chooses $Q_e = 1$ as its next assignment. (This may or may not be correct.) After running AC3, what is removed from each of the remaining queens' $\{Q_b, Q_c, Q_d, Q_f\}$ domains?

Note: I want specifically what is removed at this step. Some of the domains should have been reduced already, and I'm only interested in what reductions happen at *this* step via AC3.

c. (*2 pts*) Another way of formulating the 7-Queen problem is as follows:

- We have 7 variables (queens) $\{Q_0, Q_1, ..., Q_6\}$.
- Each variable has an initial domain {a1, a2, a3, ..., a7, b1, b2, ..., b7, ..., g6, g7} with one value for each of the 49 squares.
- There is a binary constraint between each pair of variables

Do you think this formulation will be computationally more expensive, less expensive, or about the same compared to the previous formulation? Write a short (4-5 sentences) paragraph to justify your claim. (To help you with your descriptions, consider each formulation with large N).

3. Long Answer (*12 points*) For each of the following questions, answer yes or no and give *at most* a paragraph (4-5 sentences) explanation for your answer.

a. (*3 pts*) Will iterative-deepening depth first search will always outperform regular depth-first seach?

b. (*3 pts*) Is is the case that two consistent and admissible heuristics can be combined into a heuristic that is *at least as good* as either of the originals by simply taking the average of the two heuristic values?

c. (*3 pts*) Will A* search still be complete in a search space with negative path costs?

d. (*3 pts*) Consider an agent that is going to play a zero-sum two-player deterministic, full observable game (i.e. othello or checkers). If we had a function *predict(s)* that could accurately predict the opponent's move in game state s, could we find an optimal move using an A* search?