School of Electrical Engineering and Computer Science, Washington State University
Fall 2016

CptS 440/540 Artificial Intelligence

Homework 3

Due: Thursday, November 10, 2016 (before midnight / 11:59pm)

General Instructions:
Your answers to non-programming problems should be submitted as a single PDF document. Your answer to Problem 1 (programming exercise) should include an Agent.h and an Agent.cc file, a readme.txt file, and a short report (see below). Please collect all these homework files (the ones pertaining the programming exercise as well as the pdf file with the non-programming exercises) into one zip file and submit as an attachment under Homework3 on BlackBoard-Learn system by the above deadline.

PLEASE SUBMIT ONLY ONCE, as tracking things for the 57 or so students gets really messy if many/all people make multiple submissions.

NOTE: this and subsequent assignments will include some modest hands-on coding components. Please include the code, a README file explaining your implementation and how to run your code, and a short report on your findings (including a screenshot and scores/results of your test-cases – see below for details) as parts of your Programming Exercise. Please include brief, crisp comments in your code / .cc file, explaining for each implemented agent functionality, in one short phrase, what that function/method does. The TA may contact you after you have submitted to ensure that your code runs correctly / as claimed in your report.

Note: For Pullman students, paper hard copy submissions (for the non-coding problems) remain acceptable. However, ***please also make sure to upload your HW assignment pdf file online***, so that we don't run into problems with BB-Learn and entering your score there.

1. **(Programming exercise – 50 points total)**
   A C++ Wumpus World simulator, following the rules from the textbook, is available at prof. Larry Holder's teaching website, http://www.eecs.wsu.edu/~holder/courses/AI/wumpus (the current version is 2.7). For this problem, you will implement an agent that solves (grabs the gold and climbs out of the cave) any Wumpus World satisfying the following conditions.
   a. The world is 4x4.
   b. The gold is in location [4, 4].
   c. The Wumpus is located in a square along the left, right, top or bottom wall, but not in [1,1] or [4,4].
   d. There are no pits.

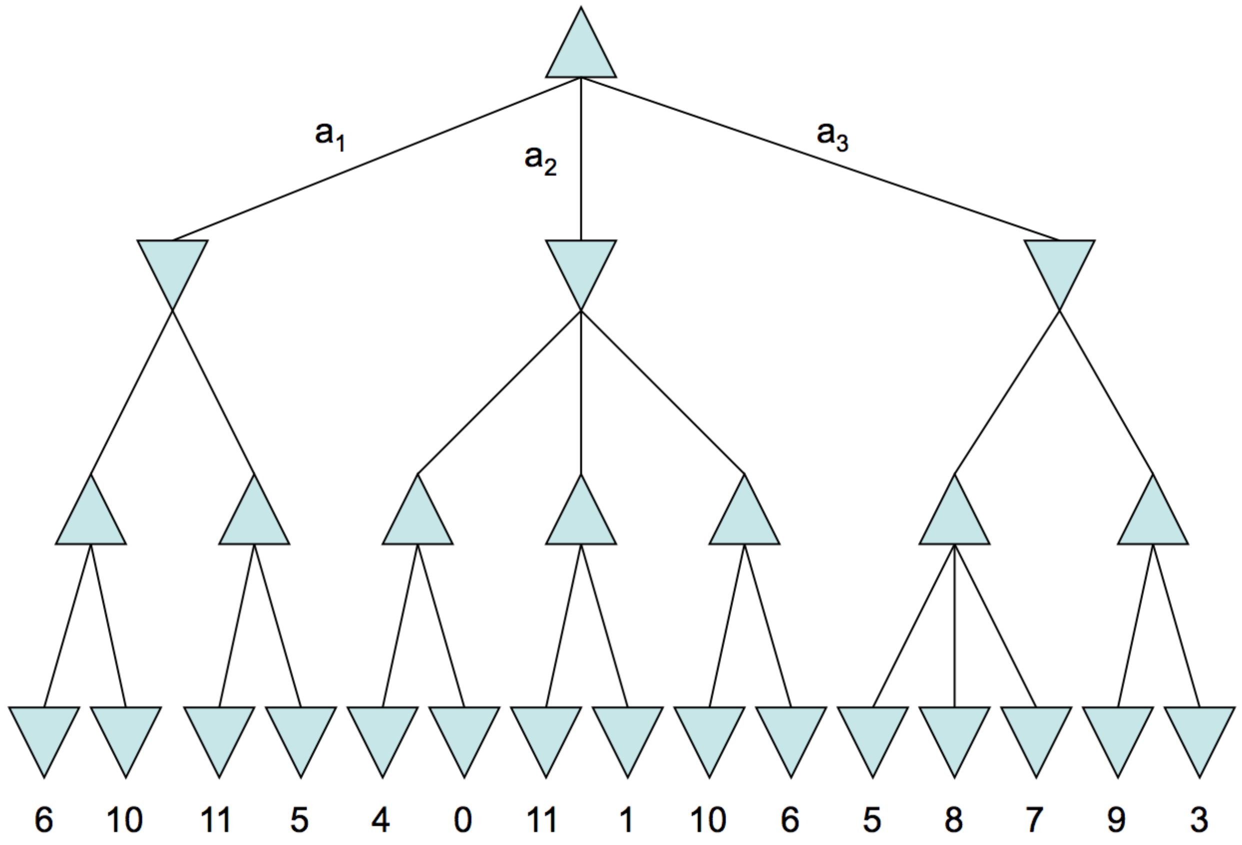
   Your goal is to design an agent for the Wumpus World with the properties (you are welcome to add additional capabilities, but not required to beyond the specified 'agent capabilities').
   a. The agent should always succeed in Grab(ing) the gold before leaving the cave.
   b. The agent should always Climb when in location [1,1] and has already (previously) grabbed the gold.
   c. The agent should keep track of safe locations, i.e., locations you are sure do not contain a live Wumpus (recall, no pits in our setting).
   d. The first time the agent encounters a stench, it should Shoot. If the agent kills the Wumpus, then stenches should be ignored when determining safe locations.

   Begin by reading the `readme.txt` file that comes with the simulator. You will be modifying the `Agent.h` and `Agent.cc` files to implement your agent. You should also submit a short `readme.txt` file containing any information you think we may need to know about your agent and implementation. Your agent shouldn't require any user input.

   You should test your agent on at least 3 initial configurations (with different choices of the location of Wumpus). You should take a screen-shot of at least one such run and include in your report your test-run results, screenshot(s) and a very brief explanation how well your 'agent' performs w.r.t. specified objectives and the configurations on which you tested it). You should also include test scores of your agent for three runs (on different initial configurations), and the average score over those three runs. Therefore, your submission for Problem 1 includes four pieces: the code (Agent.cc file), the headers/libraries file (Agent.h), the very short README plain text file (summarizing implementation) and 1-2 page long (incl. 1-2 screenshots) report focusing on performance/test-runs and their explanation (so, the short report should be written at "application level", as contrasted to the README file).

2. This problem is about a deterministic 2-player game with complete information.

   Perform Alpha-Beta pruning on the given game tree. Put an "X" over each node that is pruned, i.e., not evaluated (including all nodes in the corresponding pruned subtree). Recall, Player1 (MAX) goes first, followed by the move by Player2 (MIN), followed by the 2nd half-move by Player1 (MAX), which in this case is also the last action in the game.



Put the final value next to all other (that is, non-pruned) nodes. Indicate which action MAX should take at the beginning: a_1, a_2 or a_3. Explain in a single sentence, why is that the optimal action/move for Player1.

3. **Problem 6.2 in AIMA**. (So, this is a problem on CSP and should be answered using CSP ideas and techniques.)
   For part d), consider just even values of n (for $n \geq 2$).

Bonus Problem: in the knights-on-chessboard problem (see Problem 3 above), for even board sizes *n*, what is a smart short-cut to quickly place the maximum possible # of knights on the board? What is that number, as a function of *n* ? Explain briefly.