EECS 592
FOUNDATIONS OF ARTIFICIAL INTELLIGENCE

Issued: Feb. 8, 2018
Due: Fri., Feb. 23, 2018 (11:59pm to Canvas)

PROJECT #2: Class Credit for Games and Puzzles: Adversarial Search and CSPs

Please take care to complete all parts of this project independently. Do not use any solutions or code found online or written by other students. If you are struggling please go to office hours, post to Piazza, and/or send us email. We ask that you use one programming language for the search code in this assignment; either Python or C/C++ is fine, but not both.

Part I: Problem-solving without code (30 points)

Please submit a single PDF file `PartI.pdf` of all non-code problem solutions to Canvas. The PDF file can contain a scan of handwritten or typewritten/typeset solutions, and any supporting graphics.

1. **R&N Problem 5.12:** (10 points)
   Describe how the minimax and alpha–beta algorithms change for two-player, non-zero-sum games in which each player has a distinct utility function and both utility functions are known to both players. If there are no constraints on the two terminal utilities, is it possible for any node to be pruned by alpha–beta? What if the player's utility functions on any state differ by at most a constant *k*, making the game almost cooperative?

2. **R&N Problem 6.8:** (10 points)
   Consider the graph with 8 nodes A_1, A_2, A_3, A_4, H, T, F_1, F_2. A_i is connected to A_{i+1} for all i, each A_i is connected to H, H is connected to T, and T is connected to each F_i. Find a 3-coloring of this graph by hand using the following strategy: backtracking with conflict-directed backjumping, the variable order A_1, H, A_4, F_1, A_2, F_2, A_3, T ,and the value order R, G, B.

3. **R&N Problem 7.7:** (6 points)
   Consider a vocabulary with only four propositions, A, B, C, and D. How many models are there for the following sentences?
   a. $B \lor C$
   b. $\neg A \lor \neg B \lor \neg C \lor \neg D$.
   c. $(A \Rightarrow B) \land A \land \neg B \land C \land D$.

4. **R&N Problem 7.14:** (4 points)
   a. According to some political pundits, a person who is radical (R) is electable (E) if he/she is conservative (C), but otherwise is not electable. Which of the following are correct representations of this assertion?
   (i) $(R \land E) \Leftrightarrow C$
   (ii) $R \Rightarrow (E \Leftrightarrow C)$
   (iii) $R \Rightarrow ((C \Rightarrow E) \lor \neg E)$

   b. Which of the sentences in part (a) can be expressed in Horn form?

Part II: Codes that Solve a Game and a Puzzle (70 points)

1. **Tic-Tac-Toe (30 points):** The rules for the simple child's game of Tic Tac Toe were presented in lecture and also are available at https://en.wikipedia.org/wiki/Tic-tac-toe. Your task is to write and submit code `tictactoe.py` (or `tictactoe.cpp`) in which Autonomous Player X (playing first) fills a blank board space RANDOMLY, while Autonomous Player O (playing second) follows the `MINIMAX-DECISION` algorithm specified in R&N and on Slide 12 of Lecture 6. Because the game is so simple you do NOT have to implement alpha-beta pruning for this problem, although you are welcome to adopt it if you'd like. Your code should run through all turns of a single game and then exit. We will check your code with cpplint/pylint.

   To facilitate grading, we ask that you input a single integer number `seed` through the command line (e.g., `tictactoe seed`) and use the following functions:

   Python: `random.seed(seed), random.random()` (from library `random.py)`
   C/C++: `srand(seed), rand()` (from `stdlib`)

   To address the question from lecture today re: random number generation, simple code was written to verify that entering a particular seed on the command line generates the same sequence of random number return values. This can be verified with the below codes in C++ and Python. Note that the C++ and Python numbers are different but repeated executions of either C++ or Python codes give the same value sequences, as expected. Note that the values are scaled in this example to give integer results between 0-8. On the tic-tac-toe board assume 0=(0,0) (bottom left corner), 1=(1,0) (middle of bottom row), …, 7=(1,2) (middle of top row), and 8=(2,2) (top right corner).

```
#!/usr/bin/env python
import sys
import random
import math
s = int(sys.argv[1])
print "Seed = ", s
random.seed(s);
for i in range(0,5):
    print math.floor(9 * random.random())

// C/C++:
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main(int argc, char **argv)
{
  int seed = atoi(argv[1]);
  srand(seed);
  printf("seed = %d\n",seed);
  for (int i=0;i<5;i++)
    printf("%d\n", (int) floor(9*((double) rand())/RAND_MAX));
  return 0;
}
```

   Your output should be formatted as follows to show each move. Player X marks a move with lower-case **x**, Player O marks a move with lower-case **o**, and an open (blank) space is marked with a dash (-). An example program output (that may or may not follow Minimax convention) is shown below. Do not use spaces between entries or on the blank line to help with grading. Output terminates when either a player wins or when all board spaces are marked.

<u>Sample output file `tictactoe.txt`:</u>

```
-,-,-
-,-,x
-,-,-

-,-,-
-,o,x
-,-,-

-,-,x
-,o,x
-,-,-

-,-,x
-,o,x
-,-,o

-,-,x
x,o,x
-,-,o

-,-,x
x,o,x
-,o,o

-,-,x
x,o,x
x,o,o

-,o,x
x,o,x
x,o,o
```

2. **Sudoku (40 points):** Write and submit a code `sudoku.py` (or `sudoku.cpp`) that formulates the Sudoku game as a constraint satisfaction problem, using AC-3 to determine all values possible for each cell given arc consistency constraints. You can choose what combination of higher-level *k*-consistency and search algorithms to apply. Your code should be able to solve "easy" and "hard" Sudoku boards, so make sure your code does not annoy the grader by running slowly at least on intermediate boards. We will check your code with cpplint/pylint. If you have more than one source file provide a README and submit your code as `sudoku.tar.gz`.

   The input Sudoku board will be read as `suinput.csv` with the following format (no spaces between entries please). Note that a 0 indicates that board space is "unknown". The output solution board must be written as `suoutput.csv` with the same format. Examples are provided below using the lecture puzzle.

suinput.csv:

```
0,0,0,0,0,0,0,0,2
1,0,0,3,6,0,5,4,0
0,0,0,7,0,8,9,0,0
0,4,0,0,2,0,1,0,0
0,6,0,9,0,1,0,8,0
0,0,2,0,8,0,0,3,0
0,0,4,1,0,5,0,0,0
0,5,8,0,3,4,0,0,6
2,0,0,0,0,0,0,0,0
```

suoutput.csv:

```
4,8,6,5,1,9,3,7,2
1,9,7,3,6,2,5,4,8
3,2,5,7,4,8,9,6,1
8,4,9,6,2,3,1,5,7
7,6,3,9,5,1,2,8,4
5,1,2,4,8,7,6,3,9
6,7,4,1,9,5,8,2,3
9,5,8,2,3,4,7,1,6
2,3,1,8,7,6,4,9,5
```