CS 380 Artificial Intelligence, Spring 2019
Sect: 002
Homework 4 - Adversarial Search

by JL Popyack

Due Date: Wednesday, June 5, 2019, 11:55 PM

PURPOSE:

This assignment provides introductory experiences with adversarial search systems.

THE ASSIGNMENT:

This assignment consists of one section:

1. A program to be completed.
2. Some machine learning activities which will be started in class during Week 9.
3. Extra Credit for the programming assignment.

Click here to return to top of this page.

---

1. **Programming Assignments**
   A. ***Pentago™ (60 points)*:**
   You are to complete a program that can play a 2-player game intelligently. You have been provided with a program (`https://www.cs.drexel.edu/~jpopyack/Courses/AI/Sp19/assignments/HW4/Pentago/Pentago_base.py`) that will allow two players to play a game of Pentago™, as explained below.

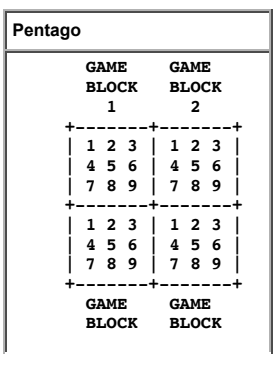
   The provided program provides a specific representation for the game, players and moves. Your are to provide a heuristic for evaluating game states, a function that determines if a game state represents a win for a given player, and a minimax routine with several-move lookahead, board evaluation heuristic and (for extra credit) alpha-beta pruning. Specific instructions for board representation, variable names, function headers and more are provided below, so that your program's structure will allow your heuristic to be used in a class-wide tournament. The rules of the game are summarized as follows:

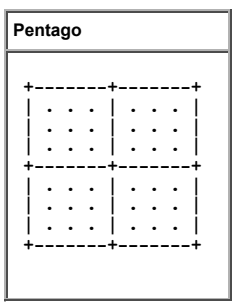
   Pentago is a 2-player game played on a 6×6 grid. The players alternate turns. The two players are referred to here as "**W**" and "**B**", which also signifies the colors of the tokens (white and black) they place on the board. The rules are summarized below as explained at the Pentago web site, under "The Rules".

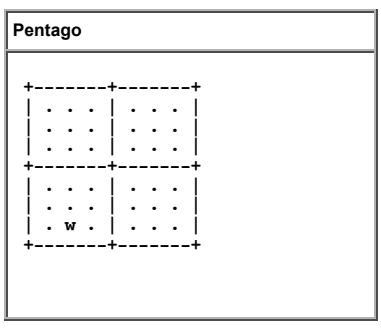
   STARTING: Start with an empty board and decide who starts, and who's playing what color.
   OBJECT: The object is to get five marbles in a row, in any direction, before your opponent does.
   PLAYING: Each turn consists of placing one marble, anywhere on the board and twisting any of the game blocks 90 degrees, in either direction. You can place your marble on one game block and twist any other game block.
   WINNING: First to five in a row wins!

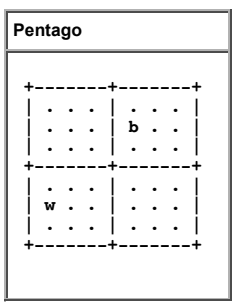
   Notes:

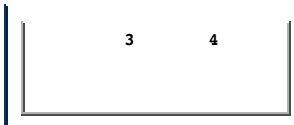
   - If the board is filled without a winner declared, the game ends in a tie.
   - Twisting the board can cause two players to win simultaneously, which is also a tie.
   - It is possible to win the game by placing a token *before* twisting a game block -- in this case the game block does not need to be twisted.







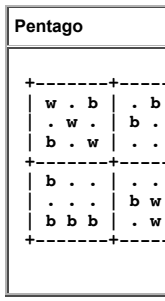


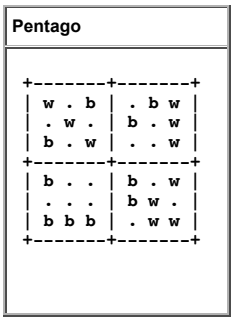


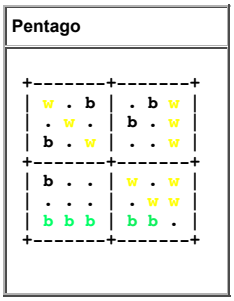
Cell numbering. Each cell is numbered by its game block and its position, e.g., `3/8`	Empty Board.	W moves first in this game, with the move `3/8 1R .` Because game block `1` is empty, rotating it has no effect	B moves next, with the move `2/4 3R .`

Program notes:

- Each player can be either a human or computer player.
- The program allows either player to make the first move.
- The program allows either player to use the **w** or **b** tokens.
- The program displays a reasonable representation of a Pentago board after each of its moves and twists, and each of its opponent's moves and twists.
- Your program should be able to recognize and declare a winner. This should be checked after each token is placed and again after each board is twisted. Because it is possible that the opponent may have one or more lines with 5 tokens in a row after a twist, the opponent may also win after a twist. In the example below, player B gets 5 tokens in a row after the twist, but player W also gets 5 tokens in a row in two separate locations. While the rules do not discuss this circumstance explicitly, we will consider it a tie.
- ***Specifying Moves:*** The squares of the board are numbered by *game block* and *position*. The square `3/8` refers to Game Block `3`, position `8` (see figures above). A move will have the form
  `b/p bd` , where `b/p` is the *block* and *position* describing the location in which a token is placed, and `bd` is a *block* and *direction* for rotation. Unlike the game as descirbed online, a player *must* provide a block and rotation for each move, even if empty blocks are present. Thus, the sample complete moves shown above: `3/8 1R` and `2/4 3L`. Your program should be able to accept input as either upper-case or lower-case letters interchangeably.







It is `B`'s move. `B` makes the move `4/1 4L .`	After `B` places a token in position `4/1 .`	After `B` rotates Game Block 4 to the Left. Notice that `B` has 5 tokens in a row. However, `W` now also has *two* lines with 5 tokens in a row. The game is declared a tie.

NOTES:

- A sample (incomplete) Python program that plays this game is provided for your use, so that everybody's program will use the same state and rule representations. A few important points about the program representation are given below:
  - You are to complete the function `win(player,Board)`, which determines whether there are five consecutive tokens with value `Player` anywhere on `Board`. (*This was a class exercise*.)
  - You are to complete the function `getComputerMove(player,Board)`, which determines what move to make when it is the computer's turn to move, and the computer's token is `player`. Currently, that function simply determines all possible moves for `player`, and chooses one at random. To do this, you should implement ***Minimax*** and use a heuristic to determine the Player's move.
  - You are to write a heuristic function named ***`userid`*`_h(player,Board)`**, where ***`userid`*** is your Drexel userid. This heuristic should return a high value for board states that are favorable for `player`, and a low value for board states that are unfavorable. This heuristic (*and not the rest of your program*) will be used to represent you in a class-wide Tac-Tical tournament at the end of the term. Your heuristic ***must*** be provided in a separate file, named ***`userid`*`_h.py`**. Any support routines you write which are used by your heuristic should also be named with the prefix ***`userid`*`_`**, and be placed in this file. You will need to use the name and arguments as shown.

- **Machine Learning Problems**:

A. **Machine Learning *(40 points)*:**
In-class exercises and followup.

Our in-class activities have introduced concepts of machine learning, along with tools and techniques for finding and improving solutions. Included have been:

Concepts:

- algorithms that improve their performance with experience
- algorithms that learn from observation and make predictions
- classification problems
- training and testing

Techniques:

- use of various machine learning algorithms (support vector machines, neural networks, etc.)
- ensemble learning/voting
- parallelism

Tools:

- numpy, pandas, sklearn, matplotlib, seaborn
- Kaggle
- Anaconda, Jupyter

For this portion of the assignment, you should do the following:

- complete the tutorials:
  - Machine Learning, Titanic, etc. (`https://www.cs.drexel.edu/~jpopyack/Courses/AI/Sp19/assignments/HW4/Titanic_etc.html`) (includes Jupyter Notebook for Beginners: A Tutorial, Titanic: Submit your first Kaggle prediction, and Titanic Data Science Solutions)
  -
- prepare a final report (1-2 pp.), which includes results and a reflection on the machine learning exercises. Your report should summarize items you recorded in your Jupyter notebooks. Some of the things you might want to reflect on are:
  - problems encountered in preconditioning data
  - applicability of these tools and techniques to problems you foresee enountering in the future
  - benefits of using parallelism to approach larger scale problems
- submit this report, along with any Jupyter notebooks or other materials you prepared.

Click here to return to top of this page.

Extra Credit:

A. **Alpha-Beta Pruning*(10 points)*:** Make use of alpha-beta pruning in your program. You should collect data on number of states evaluated with and without pruning, amount of time used, and depth achieved. Summarize your results in a table.

Click here to return to top of this page.

WHAT TO SUBMIT:

All homework for this course must be submitted electronically using Bb Learn. ***Do not e-mail your assignment to a TA or Instructor!*** If you are having difficulty with your Bb Vista account, ***you*** are responsible for resolving these problems with a TA, an Instructor, or someone from IRT, before the assignment it due. It is suggested you complete your work early so that a TA can help you if you have difficulty with this process.

For this assignment, you must submit:

- A PDF document with written documentation for your program (including discussion of heuristics, I/O for at least one sample game, and results of your testing).
- Source code for your program :
  - Your should provide a *program file* and a *heuristic file*.
  - Your heuristic ***must*** be named **`userid_h(Player,Board)`**.
  - Your heuristic ***must*** be provided in a separate file, named **`userid_h.py`**. Any support routines you write which are used by your heuristic should also be named with the prefix **`userid_`**, and be placed in this file.
  - Materials from the Machine Learning exercises (1-2 page reflection, Jupyter notebooks).

Click here to return to top of this page.

ACADEMIC HONESTY:

You must compose all program and written material yourself, including answers to book questions. All material taken from outside sources must be appropriately cited. If you need assistance with this aspect of the assignment, see a consultant during consulting hours.

Click here to return to top of this page.