Coursework 2: Adversarial Search on Connect Four

Dr Francesco Belardinelli

November 17, 2021

- **This coursework can be completed in pairs.**
- **Deadline: December 2nd**
- **Total possible mark: 30 (to be scaled to max. 100).**
- **Submit on CATe one zip archive with your sources in Python, and one short pdf report with your answers to question 3 and 4 below.**

For this coursework you have to implement adversarial search by Minimax and α-β pruning to play a variant of Connect Four.

Consider a Connect k game played on a vertically suspended grid with m columns and n rows, or (m, n, k)-game for short. Two players, *Max* and *Min*, take turns in dropping Xs (*Max*) and Os (*Min*) on the $m \times n$ grid. The pieces fall straight down, occupying the lowest available space within the column. The goal of the game is to be the first to form a horizontal, vertical, or diagonal line of k Xs (resp. Os). Thus, Connect Four is the $(7, 6, 4)$-game.

1. **[10 marks]** Write a program in Python that implements adversarial search by Minimax on (m, n, k)-games.

   The program will include a class `Game` with a method `play()` that allows for playing the game, as well as the following:

   (a) A constructor `__init__()`; a method `initialize_game()` to initialize the empty $m \times n$ grid at the beginning; and a method `drawboard()` to output the board on the screen.

   (b) At each step the program computes the Minimax strategy for *Max* and recommend the relevant action(s) to the user.
   The user is then prompted to insert the coordinates of the chosen cell.
   Finally, the program outputs the move for *Min*.
   Write methods `max()` and `min()` to compute Minimax values for both players, as well as methods `is_valid()` and `is_terminal()` to check for valid moves and terminal states.

   Please add an appropriate number of single-line comments to your code to explain the user how it works.

2. **[10 marks]** Implement α-β pruning when computing Minimax values to speed up action selection.

   In particular, modify methods `max()` and `min()` to account for the values of α and β.

   Again, please comment briefly your code.

3. **[5 marks]** Check how the execution times for action selection for Minimax scale up depending on parameters m, n, and k of the game.
   Draw an appropriate table with increasing values of m, n, and k.

   You can use the `time` module and function `time()` to measure the time to evaluate the game tree at every move.

   Is there a significant difference in selection time for actions? What about the number of visited states?

4. **[5 marks]** Do the same as in the previous point, this time for Minimax with α-β pruning (again, you can use the `time` module).

   Is the difference with and without α-β pruning significant? Please draw a table and justify your answer with the experimental results.