Coursework 2: Adversarial Search on (m, n, k)-games

November 6, 2019

- **This coursework can be completed in pairs.**
- **Deadline: November 29.**
- **Total possible mark: 17 (to be scaled to max. 100)**

For this coursework you will have to implement adversarial search by Minimax and α-β pruning to play (m, n, k)-games.

An (m, n, k)-game is a board game in which two players, Max and Min, take turns in placing Xs (Max) and Os (Min) on an $m \times n$ board, the winner being the player who first gets k Xs (resp. Os) in a row, horizontally, vertically, or diagonally. Thus, tic-tac-toe is the $(3, 3, 3)$-game.

More information can be found at `https://en.wikipedia.org/wiki/M,n,k-game`

1. **[6 marks]** Write a program in Python that implement 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__()`, method `initialize_game()` to initialize the empty $m \times n$ board at the beginning, and 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.

2. **[6 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.

3. **[2 marks]** Compare the times for action selection for Minimax with and without α-β pruning.

   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?

4. **[2 marks]** Check how the execution time of your tool scales up depending on parameters m, n, and k of the game, both with and without α-β pruning (again, you can use the `time` module).

   Is the difference significant?

5. **[1 mark]** Computer search by W. Hsu and C. Ko has shown that both $(7, 7, 5)$-game and $(8, 8, 5)$-game are draws `https://content.iospress.com/articles/icga-journal/icg180061`.

   Try to replicate their results by running your tool with these parameters.