Programming Assignment 1 — Connect Four

In the first assignment, we condiser the two-player game *connect four*, also known as "vier op een rij" in Dutch.

Figure 1: Illustration of connect four.

Game description

In connect four, two players (red and blue) aim to connect four discs of their own color on the board, either horizontally, vertically, or diagonally. The red and blue players start off with an equal number of discs (each with half of the number of entries on the board). A disc can be placed by dropping it in a column that is not yet filled. The game ends as soon as one of the players has succeeded in creating such a pattern, in which case that player wins; or when no more moves can be played, in which case the red player wins because blue had a slight advantage by playing the first move. Note that this variant is a slight variation on the more commonly used rules of the game.

Board implementation In the code that is provided in `a1.zip`, the board is implemented as a matrix with n rows and m columns. The coordinate system to index the individual entries of a 4×5 board is displayed in Figure 2.

Programming

The goal of the assignment is to write an exhaustive search algorithm that determines whether the player that is on turn can win the game, *assuming that both players play optimally.*

We provide you with a template program. It consists of two files:

- `connect.py`: The main assignment file
- `test_connect.py`: A Unit Test file

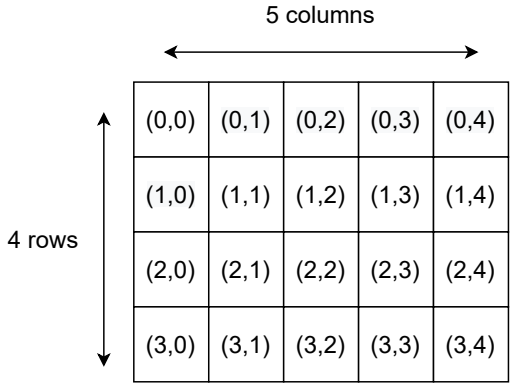

Figure 2: Caption

The file `connect.py` contains various functions that are not implemented yet (in total: 9). Read the documentation about these functions and implement them. Most of the functions require less than 20 lines of code. None of the functions require more than 30 lines of code. Do not change the headers of the functions provided. You are allowed to add functions yourself if you document them, but we strongly suggest you use the existing template for efficiency.

The file `test_connect.py` consists of several test functions. You can use these to give you an idea of whether functions are implemented correctly. Note that we do not test everything and a passing unit test does not mean that everything is implemented correctly! For this reason, we also ask you to come up with unit tests for (tricky) cases that the provided tests do not check for. The unit tests can be executed by running the following command:

```
python -m unittest test_connect.py
```

Make sure to have the right packages installed. Do not use other packages than those present in the template. Feel free to add more unit tests. Do not alter the unit tests that are provided. If your program does not succeed on all unit tests that are provided, it is likely that there is still a problem in your code. Make sure that all the unit tests succeed, before submitting the code.

Additionally, also hand in a file named `test_private.py`. In here, you can create additional unit tests to verify the working of your program. Write at least 3 additional unit tests, and hand these in along with the assignment. Also keep in mind that all unit tests should be able to run within a matter of seconds on any computer.

Report

Write a report in LaTeX(at most 3 pages), addressing the following points / research questions:

- Introduction: describe the problem. Describe a state and action.
- Draw the state-space of a small example, e.g., on a partially filled 4×4 board You are allowed to draw this on paper and scan the result.
- Analysis of optimal method: How much time does the optimal method take on various board sizes? Make a plot or a table of this. What is the limit that you can run within 10 minutes?

- How well does the greedy approach work? How often and how much does the answer differ from the result of the optimal algorithm? Include an adversarial test-case in your report where the greedy approach does not yield an optimal move.
- Summary and Discussion. Always end a report with some summarizing remarks: what did you investigate, what were the observations, what do we learn in a scientific sense about these algorithms.

The deadline of this assignment is the **18th of March 2022, 23:59 CET (Central European Time)**.