Coding++ Assessment

The assessment is formed of three mini-projects, you need to complete one of them. Each project follows on from and extends a part of the course. You have two months to complete one of the projects in your own time. Each project requires both skill and autonomy: you will need to try things out, to read documentation, to research algorithms, etc.

Each of the projects includes both development and write-up tasks. The code should be written in Python. The written-up document should be short (two pages maximum).

To submit your work, push your code and write-up (as a text file or a pdf) to a version control repository (e.g. GitHub, GitLab or BitBucket, see below) and send us the URL of the repository associated with the assignment. Setting up an account for one of these services and putting your code online part of the assessment, and it will also help you build up an online presence as a programmer.

Links:

- Github: https://github.com/
- Github help: https://help.github.com/
- Gitlab: https://about.gitlab.com/
- Gitlab help: https://docs.gitlab.com/ce/university/
- Bitbucket: https://bitbucket.org/

Project 1: Alpha-Beta Pruning AI for Connect 4

The minimax algorithm to search for optimal moves is time consuming: it searches through the play tree to analyse all possible moves. What if there were a way to eliminate some branches of the search tree when you know they will not contain the optimal move? Alpha-Beta pruning (or α-β pruning) is a variant of Minimax in which some moves are not considered. To understand how the algorithm works, read the Wikipedia article. You can also search for other resources.
https://en.wikipedia.org/wiki/Alpha–beta_pruning

Activity 1: Implement an Alpha-Beta AI for your Connect 4 game

Implement an Alpha-Beta pruning AI for Connect 4 with the same basic structure as your Minimax AI. The changes should affect `max_play` and `min_play`, in particular, some boards should not be explored, which makes the functions more efficient.

Activity 2: Evaluate the performance of your Alpha-Beta AI

Play your Alpha-Beta AI against your Minimax AI for different values of the ply parameter multiple times. Modify the code to record, for each match:

- the ply,
- which AI won, and
- how much time each AI spent choosing moves.

Create a graph as follows:

- The vertical axis is "time spent by α-β / time spent by Minimax",
- The horizontal axis is "plies",
- Add a point on the graph for each match.

Activity 3: Write up

What trends are visible on the graph? What does it reveal about the different AIs? What is the win-ratio of Alpha-Beta? Why?

Which AI would you choose to use in a tournament? What value of `ply` would you use? Why?

Project 2: Pentago

Pentago is a two-player game with a 6x6 grid decomposable to four 3x3 grids that can rotate independently. At each turn, the player places a token on the board and rotates any one of the four 3x3 grids by 90 degrees. A player wins when they have a vertical, horizontal or diagonal row of five tokens. You can find online and phone versions or watch youtube tutorials to familiarise yourself with the game. You can find the rules at: http://pentago.org/documents/Pentago-Rules-Multilingual.pdf

Activity 1: Implement the game Pentago

With the same basic framework you used for Connect 4, implement Pentago. Remember that for each move, the player has to both place a token on the board and twist one of the 3x3 grids. This is a more complex interaction than with Tic-Tac-Toe or Connect 4.

Activity 2: Develop an AI for your Pentago game

Start by implementing a random AI for your Pentago game and then develop an AI that performs better than this random AI. You can find heuristics, apply one of the algorithms from the course or research other techniques.

Activity 3: Write up

Does your AI consider moves prior to the current game state or look ahead beyond the current game state? What techniques does your AI use?

How does your AI compare to the random AI in terms of rate of winning and time to compute?

Project 3: Trivial Pursuit/Pub Quiz chatbot

Questions in Trivial Pursuit and pub quizzes tend to be well-formed with a clear question word and subject (often a proper noun), e.g. When was the Berlin Wall built? You can use these features to develop a chatbot to help you answer these questions.

Activity 1: Parse the question

Using the SpaCy NLP library, develop a terminal-based bot that given a valid input identifies:

- the target subject (e.g. Berlin Wall), which should be a noun (most likely a proper noun),
- the question type (e.g. when), and
- the question details (e.g. built).

The bot will most likely misbehave for certain questions. Note the classes of questions for which it fails.

Activity 2: Search for the answer

Get your bot to retrieve the Wikipedia article associated with the question and find the answer. Note how the (English) wikipedia article urls are structured: `https://en.wikipedia.org/wiki/Words_Separated_By_Underscores`, e.g. https://en.wikipedia.org/wiki/Berlin_Wall. Additionally, Wikipedia has a search engine: `https://en.wikipedia.org/w/index.php?search=terms+separated+by+space`, e.g. https://en.wikipedia.org/w/index.php?search=Berlin+wall.

You can use this structure to define the url based on the subject of the question. Then read in the text from the page. Use natural language processing and/or knowledge of words that tend to be associated with certain answers – e.g. “year”, “date”, “day” for “when” questions. You can also search for sentences where the question details appear.

If there is no page associated with your subject, your chatbot will need to respond apologetically to the user.

Activity 3: Write up

Give a few examples of questions that your bot processes wrongly. What type of issues are there? What types of questions do they appear on?

Propose improvements that could make your bot better and explain why you think they would work. (Note: you do not need to implement them.)