CAB203 Project

1 Introduction

In this project will demonstrate your knowledge and skill with the material in the unit by developing a solution to a real world problem by translating it to mathematical language, relating the problem to well known problems on mathematical structures, and implementing software to solve the problem.

Your submission will consist of two parts: a report detailing the mathematical descriptions of the problem and solutions, and a Python file containing your implementation of the solution.

This assignment is worth 40% of your final grade.

Please see the Canvas Assignments page for due dates.

2 Scenario

Congratulations on your acceptance to the organising committee of the 2023 CAB203 Bonkers tournament! Bonkers is an extremely exciting and fictitious game. In fact it is so fictitious that the only thing that exists about it is its name!

There are many tasks that need to be done, which have been distributed amongst the organising committee members. Your acumen with discrete structures is well known, and hence you have been assigned several organisational tasks which will benefit from some computerisation.

1. The tournament structure needs to be set; that is to say, who plays who. For this tournament a simple structure is used where all games are set at the beginning. Another committee member will propose the tournament structure and your job is to determine whether it has the required properties. The committee has determined that the following property is sufficient:

   For every pair of distinct players A, B, either A plays against B or A plays against some other player that plays against B.

   Your job is to find a method that, given a tournament structure, determines whether the above property holds.

   Implement your method in a Python function like so:

   ```
   def gamesOK(games):
   ```

   where `games` is a set of proposed games to play, given as pairs of players, like so:

   ```
   games = { ("Alice", "Bob"), ("Charlie", "Bob") }
   ```

Your function should return `True` if the required property holds for the games given, otherwise `False`. You may assume that in the games given no player plays against themselves and that every player plays in at least one game.

2. All Bonkers games must have a referee, but the referee for each game cannot have any conflicts of interest for the game. In particular, the referee cannot be a player in the game, and the referee must also declare any other conflicts of interest (eg. cannot be a player's relative, friend, boss, etc..) The potential referees and their declared conflicts of interest are recorded in a CSV file where the first column in the name of the referee, and the subsequent columns each contain players where the referee has a conflict of interest. For example:

   ```
   Referee,Conflict 1, Conflict 2, Conflict 3, Conflict 4
   Joe,Ashley,Bob,Charlie
   ```

   where Joe is the referee and Joe has conflicts of interest for players Ashley, Bob and Charlie. Note the header row.

   Your task is, given the CSV file in the format described above and two players for a game, find the set of referees with no conflicts of interest for the game.

   Implement your solution as a Python function of the form

   ```
   def potentialReferees(refereecsvfilename, player1, player2):
   ```

   which returns a Python set like:

   ```
   { "Joe", "David", "Rene" }
   ```

   Note that there may be no suitable referees, in which case you should return an empty set.

3. Once the potential referees for each game have been determined, it is necessary to choose one. In order to minimise the burden on any individual referee, the policy is to assign each referee at most one game.

   Your task is to determine a method for assigning referees to games, given the games and the potential referees for each game. You must assign at most one game to each referee, and exactly one referee to each game, or determine that it is impossible to do so.

   Implement your solution as a Python function of the form

   ```
   def gameReferees(gamePotentialReferees):
   ```

   where `gamePotentialReferees` is a data structures similar to:

   ```
   # keys are games, values are sets of potential referees
   gamePotentialReferees = { ("Ashley", "Bob"): { "Rene", "David" } }
   ```

   Your function should return a data structure of games and assigned referees similar to:

   ```
   # dictionary of all (player1, player2) : referee
   { ("Ashley", "Bob"): "Rene" }
   ```

   or `None` if it is not possible.

4. The games now need to be scheduled. Your task is to find a schedule for the games. A schedule is a list of games to be played in each time slot such that

   - each person is involved in at most one game at any time, either as a player or referee
   - each game is played exactly once
   - the number of time slots used is minimal

   Implement your solution as a Python function of the form

```
def gameSchedule(assignedReferees):
```

where `assignedReferess` is in the same format as you output in the previous question, i.e. as a dictionary with games as keys and referees as values. Your output shoud be a schedule in a data structure of the form

```
# list of timeslots
# each timeslot is a set of games
# each game is a triple (player1, player2, referee)
[
   {("Ashley", "Bob", "Rene"), ("Charlie", "Dave", "Elaine") },
   { ("Bob", "Charlie", "Ashley"), ("Rene", "Elaine", "Dave") }
]
```

5. Once all games are played, it is necessary to find the tournament winners. In the ideal case we can produce a ranking so that no player lost a game to a player with a lower ranking. Your task is to either find such a ranking or determine that it is impossible. There may be more than one possible ranking that satisfies the criteria, in which case you can give one of them.

   Implement your solution as a Python function of the form

   ```
   def ranking(games):
   ```

   where `games` is a data structures similar to:

   ```
   # each pair is a game (winner, loser)
   games = { ("Ashley", "Bob"), ("Bob", "Charlie") }
   ```

   Your function should return a ranking like so:

   ```
   [ "Ashley", "Bob", "Charlie" ]
   ```

   with the top ranked player (overall winner) coming first. If no such ranking is possible, return `None`.

3 Report

Your report should have four sections, one for each of your assigned tasks. Each section should discuss the following (preferably in this order)

1. Use mathematical language, concepts and notation from the unit to describe the problem. You should make use of mathematical tools and problems discussed in the unit, for example quantified predicates or finding a shortest path in a graph. Describe the information given in mathematical terms and using mathematical notation (e.g. the games are given a set of pairs of players like $\{(a, b), (c, d)\}$).
2. Describe, using mathematical terms and notation, how to find a solution for a given instance of the problem. If applicable, describe how the information given relates to other mathematical objects that are needed. For example, how are the vertices and edges of a graph related to the given games. Describe how the solution to the mathematical problem relates to the solution to the original problem.
3. Describe your implementation in Python. Mention the role of important variables, library calls (eg. to `graphs.py` or `digraphs.py`), and source of reused code if applicable and any modifications required to it. If you need to make a choice about data structures, explain why your choice. Please note that this section should be about programming considerations, not solution methods, which is the above point.

Overall, report should be understandable by another student in CAB203 who knows the material but hasn't thought about the tasks. It is not necessary to define terms already used in the unit, but you should point out the significance of particular details about the problem and the choices that you make in modelling and solving it.

Your report will be a single file, in PDF format. There is no maximum page length, but a concise, easy to understand report is better than a long wordy report. Four or five pages is about right.

3.1 Python implementation

Your solution should be a reasonable implementation of the mathematical solution described in your report. The problems are all solvable using the Python concepts and syntax used in the unit. You can use additional syntax if you like. The marking system will use Python 3.10, so if you are using a later version be sure not to use any syntax newer than 3.10.

You are allowed to use or modify any functions defined throughout the lectures, tutorials, and assignment solutions. Many of these functions and more are collected in Python files `graphs.py` and `digraphs.py`. You can import these files rather than copying from them. You can assume that these files are available; there is no need to include them in your submission. Additionally, you are allowed to use the `csv` Python module. Before using other modules, please contact the unit coordinator.

A submission template file is available from the Canvas Assessments module. If your solution includes modified code from the unit, say so in a comment explaining where you obtained it and what modifications you made. One line is enough detail.

A test file is included to help you debug your code, available from the Canvas Assessments module. **Please make sure that you run it before submission.** You can run the test file directly: `python test_project.py`. Mac and Linux users may need to run `python3 test_project.py`. Or, run it through Thonny. Be sure that `test_project.py` is in the same directory as your solution, and that your solution is called `project.py`.

The test file is structured as a Pytest unit test file, so you can use Pytest if you like:

```
pytest test_project.py
```

You can install Pytest with `pip install pytest`.

The tasks are all chosen so that they can be solved with a short function (Matt's solution is 62 lines including comments). There is no limit on the length of your program. However, the marking system will impose a time limit of about 5 seconds to avoid problems with infinite loops. This should be plenty of time to solve the tasks given.

Your code submission will be a single Python file.

4 Marking criteria

Your mark is made of two parts. The report is graded out of 30 and the Python code is graded out of 10, for a total of 40. Each mark counts 1% towards your final grade.

4.1 Report

The marking rubric is available on Canvas under Assignments > Project Report.

4.2 Python code

Your Python code will be graded automatically by running test cases. The tests will be similar, but not identical, to those found in `test_project.py`. There will be 30 tests, 6 for each task. Each test is 1/3 of a mark, for a total of 10 marks.

Your code is not assessed for quality, format, comments, length, etc.. Only the automated tests count for marks.

5 Submission

Submission process: You will need to make two submissions through two separate links in Canvas:

- Your report, in PDF format (extension `.pdf`).
- Your Python code, as a Python file (extension `.py`).

You can find the submission pages on Canvas on the Assignments page.

Extensions: Information about extensions is available on the About Assessments module on Canvas. If you obtain an extension, please attach confirmation *as a separate file* to your report submission.

Citing your sources: You are welcome to source information and code from the internet or other sources. However, to avoid committing academic misconduct, you must cite any sources that you use. See https://www.citewrite.qut.edu.au/cite/ for guidelines on citing sources and how to properly format and acknowledge quoted material.

You are welcome to use resources, including code, from within the unit. Please cite the unit like *CAB203, Tutorial 7* or similar. This is only necessary when explicitly quoting unit material. There is no need, for example, to cite the definition of a graph or similar, unless you are directly quoting the lecture's definition.

For code, please include your citation as a comment within the code. For example

```
# modified from CAB203 graphs.py
```

Policy on collaboration: We encourage you to learn from your peers. However, for assessment you need to turn in your own work, and you will learn best if you have spent some time thinking about the problem for yourself before talking with others. For this reason, talking with other students about the project is encouraged, as long as you are putting in the effort on the problems yourself as well. But do not share your code or your report with other students and do not copy from others. It is considered academic misconduct to copy from other students or to provide your work to others for the purposes of copying.

For Slack and other online discussions, please do not post about solutions. Keep your discussions private so that everyone gets a chance to get to the solutions on their own. You can direct message or email the unit coordinator or one of the tutors if you wish to discuss specifics of your solution. Feel free, however, to ask general questions about the project on the Slack channels.