QUT

CAB203 Graphs 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

The South East Queensland rail network has 6 lines and over 150 stations.[1] The Queensland Rail planning office has hired you to improve their planning processes by automating several processes that are done in the office. This will improve the speed an accuracy of these processes, allowing the planners to concentrate on other aspects of their jobs. They have assigned you 4 tasks to start with.

All tasks are intended to make significant use of graph theoretic tools and problems.

[1]The scenario and problems here are inspired by real problems that Queensland Rail — or another rail service — might have to solve, but we have simplified them to make them tractable with the tools available in this unit.

2.1 Transfers

A rail network consists of a number of lines. A line, for our purposes, can be thought of as a list of stations where consecutive stations in the list are connected by tracks. Trains run along lines, in both directions.

Planners sometimes look at possible extensions to the rail network which have additional lines. As part of this work they want to ensure that the trains remain a practical option for travellers going between any two stations. In particular, the number of transfers required (i.e. getting off one train and getting on another train) needs to be minimised.

For this task the planners would like you to calculate the minimum number of transfers required to travel between each pair of stations, and report the maximum of all of those. In other words, find the minimal t such that "You can get from any station to any other station with at most t transfers" is true.

The planners use a tool that can save their rail network to a CSV file. Each row corresponds to a train line, and lists the name of the line and the stations in that line, like so:

```
Springfield line,Springfield,Indooroopilly,Brisbane Central,Eagle Junction,Kippa Ring
Cleveland line,Cleveland,South Bank,Brisbane Central,Eagle Junction,Shorncliffe
Ferny Grove line,Ferny Grove,Brisbane Central,South Bank,Beenleigh
```

In the above example you can get from any station to any other station with at most 1 transfer. You can see this since all lines go through a common station (eg. Brisbane Central), where you could transfer between any two lines.

Your task is to write a Python function `maxMinTransfers(fileName)` which reads in a CSV file with filename contained in `fileName` and returns a single integer, which is the value of t as explained above.

2.2 Assigning crew

Every day there are dozens of trains running at different times, and they all need a crew, consisting of one driver and one guard[2]. There are multiple considerations that need to be taken into account when assigning drivers and guards to trains:

- Every train needs one driver and one guard.
- Each person can be assigned to one train on a given day.
- People have a shift, and the train's journey must fall entirely within their's shift. The shifts are:

  ```
  shiftTimes = {
      'Morning': (4, 12),
      'Day': (9, 17),
      'Night': (16, 24)
  }
  ```

- Some drivers and guards are restricted from being crew during peek times. Peek times are defined as being from 8am to 10am and from 4pm to 6pm.
- Trains running on lines equipped with *European train control system*[3] (ETCS) must have a driver that has been trained for ETCS.
- Some people are qualified to work in both guard and driver roles, but they cannot serve as both guard and driver at the same time.

Your task is to write a Python function `assignCrew(people, trains)` that either returns an assignment of people to driver and guard roles in train crews, or indicate that no assignment is possible. A separate system, developed by someone else, will call your function with appropriate arguments, which will be in data structures like:

```
people = {
    # Driver,   Roles allowed         ETCS certified, shift,     peekTimeRestricted
    ('Alice',   ('Guard', 'Driver'), True,           'Morning', False),
    ('Bob',     ('Driver',),         False,          'Day',     True),
    ('Charlie', ('Guard',),          True,           'Morning', False),
    ('Denise',  ('Guard', 'Driver'), False,          'Day',     True)
}

trains = {
    # Line,  StartTime, EndTime, ETCS required
    ('IPNA', 6,         9,       True),
    ('CASP', 11,        13,      False)
}
```

Your function should return a data structure like:

[2]Guards are also known as *conductors* in some places. They ride along in the train and are responsible for safety and communication with passengers, among other things.

[3]`https://en.wikipedia.org/wiki/European_Train_Control_System`

```
{
    # Timeslot   Driver    Guard
   'IPNA-6-9':   ('Alice', 'Charlie'),
   'CASP-11-13': ('Bob',   'Denise')
}
```

Note that in the returned data structure the trains need to be named like `'{line}-{startTime}-{endTime}'`. Times will always be an integer from 0 to 24, indicating the hour. 24 indicates midnight at the start of the following day. If no assignment is possible, your function should return `None`.

2.3 Scheduling trains

In addition to assigning drivers, actual physical trains need to be assigned to each line and time slot. There are many factors that come into this, such as the train's top speed, whether it has ETCS equipment installed, where the train is, maintenance schedules, and so on.

For this task we'll look at a simplified problem. Rather than assigning particular trains to lines/timeslots based on their features, we'll look at the minimum number of trains required to serve all lines and timeslots, ignoring all factors other than whether the timeslots overlap, allowing time for the train to relocate if necessary. To keep things simple we'll say that a train serving two timeslots in a row always needs one hour in between to relocate.

Your task is to write a Python function `trainSchedule(timeSlots)` which returns the minimum number of trains required, as discussed above. The data structure `timeSlots` looks like:

```
timeSlots = {
   # Line,  StartTime, EndTime
   ('IPNA', 6,         9),
   ('CASP', 11,        13),
   ('ABCD', 9,         12)
}
```

Your function should return a positive integer. For the above example it is 2.

2.4 Capacity to downtown

The South East Queensland rail network, like many metropolitan rail networks, has a central hub at Roma Street Station. All lines go through Roma Street Station, and many passengers travel to or through Roma Street Station as part of their journey. Hence the number of trains that can get to Roma Street Station in an hour is a reasonable measure of the useful capacity of the rail network as a whole. While there are many factors that affect this, such as the availability of rolling stock (i.e. trains) and drivers, we'll look at it purely from the point of view of tracks. There are a few factors that determine how many trains can travel along a particular piece of track, such as whether the track is double-tracked (having a separate track for each direction). We'll ignore most of these and just look at *signalling*.[4]

Queensland Rail uses *automatic block signalling* with fixed blocks. This means that tracks are divided into *blocks*. Only one train is allowed in a block at a time to prevent collisions. There are automatic signals — similar to traffic lights — at the start of each block that tell the driver whether they can proceed or not.[5] Such a system places limits on the capacity of the track. For example, if it takes 5 minutes for a train to traverse a block (including a safety margin), then no more than $60/5 = 12$ trains can travel through that block in an hour.

Your task is to write a Python function `trackNetworkCapacity(trackNetwork, blockTimes, destination)` that calculates the maximum number of trains per hour that can reach the destination. The data structures are like[6]:

```
trackNetwork = [
    [ 'Ipswitch', 'Darra', 'signal_1', 'Roma Street' ],
    [ 'Springfield', 'Darra' ],
    [ 'Varsity Lakes', 'Bogo Road', 'Merivale Bridge South', 'Merivale Bridge North',
        'signal_2', 'Roma Street' ],
    [ 'Cleaveland', 'Bogo Road' ],
    [ 'signal_1', 'signal_2']
]

blockTimes = {
    ('Ipswitch', 'Darra'):  4,
    ('Darra', 'signal_1'):  4,
    ('signal_1', 'Roma Street'): 4,
    ('Springfield', 'Darra'): 4,
    ('Varsity Lakes', 'Bogo Road'):  4,
    ('Bogo Road', 'Merivale Bridge South'):  4,
    ('Merivale Bridge South', 'Merivale Bridge North'): 6,
    ('Merivale Bridge North', 'signal_2'): 3,
    ('signal_2', 'Roma Street'):  3,
    ('Cleaveland', 'Bogo Road'):  4,
    ('signal_1', 'signal_2'): 6
}
```

[4]This is a relevant topic recently, as Queensland Rail is upgrading its signalling system to ETCS as part of the Cross-River rail project. ETCS will allow higher capacity along critical sections of the network, particularly the new tunnels under the river.

[5]See AS 7711:2018 and Observance of Signals Reaccreditation Work Book.

[6]Note that this data is not intended to be correct for the actual Brisbane train network.

Each element of `trackNetwork` is a list of signals. Signals indicate the start/end of a block. They are often located at stations, *points*[7] and other important locations. Each list of signals is a segment of track that a train can travel along. The lists always start from outer stations and proceed in the direction of travel towards the destination. `trackNetwork` thus represents multiple segments of track that together form the rail network. Between each pair of consecutive signals is a block. `blockTimes` indicates the minimum amount of time in minutes it takes for a train to traverse the block. Put differently, `blockTimes` gives the travel times between signals.

Your function should return a positive number indicating the maximum number of trains per hour that can travel from the outer stations to the destination. For the example given above with destination `Roma Street` this value is 35.

[7] A set of points is a mechanism located where two tracks join into one that allows trains to be directed to one track or the other.

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.

Additionally, your report should include a bibliography using consistent, appropriate style. Some standard citation styles are explained at The QUT Citation tool, any of which is acceptable. At minimum you should cite at least two sources, including lecture material and a standard reference such as a textbook. The sample report — available on Canvas — demonstrates reasonable citations and bibliography.

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 minimum or maximum page length, but a concise, easy to understand report is better than a long wordy report. Four or five pages is about right, not including diagrams if you have any.

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 as long as it is compatible with Python 3.10. One exception is that using loops incurs a penalty (see the marking criteria below.) The purpose of penalising loops is to encourage you to think in terms of mathematical style declarative structures rather than procedures. All tasks are solvable in the intended way without using loops.

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`. We prefer that you import these files rather than copying from them; the questions are designed so that you can use the functions directly without modifying 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`, `itertools`, and `functools` Python modules. Before using other modules, please contact the unit coordinator as the grading system may not have the module installed.

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 also compatible with `pytest` if you prefer to use it, but this is not necessary.

The test file is structured using `unittest`, which is part of the Python standard library. It can be run directly to test your solutions:

```
python test_project.py
```

The tasks are all chosen so that they can be solved with a relatively short function (Matt's solution is 125 lines, including comments and blank lines). There is no limit on the length of your program, either in number of lines or length of time it takes to run. However, the marking system will impose a time limit of a few minutes minutes to avoid problems with infinite loops. This should be plenty of time to solve the tasks given (Matt's solution runs in around 20 seconds for all tests.)

Your code submission will be a single Python file called `project.py`.

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 > Graph Project Report.

Note that some criteria are worth more than others.

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 20 tests, 5 for each task. Each test is 1/2 of a mark, for a total of 10 marks. We are using Gradescope for submission, which will automatically run the sample test cases that we have provided. You will be able to see the results of these sample tests. This is to allow you to make sure that you have submitted a working solution. *Please check these test results after submission.* In the past many students have submitted non-working solutions that could have easily been fixed (eg. submitting wrong file, wrong filename, importing modules that are not used but not present on the marking system, etc.)

For each task, one of the tests checks to see if you have used `for` or `while` loops. This is to encourage you to think mathematically rather than procedurally. Please note that *you are allowed to use loops if you desire,* but you will lose a very small number of marks (maximum 2% of your final grade). *It is much better to turn in a working solution with loops than a non-working solution without loops.*

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.

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 (named `project.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, you may *optionally* 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. Other students in the unit are not considered appropriate sources.

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 it is for the purposes of improving your understanding, and you are doing the actual project yourself. In particular, *your submission (both the report and code) must be entirely your own* unless you cite your sources. Other students in the unit are not considered to be appropriate sources.

To be clear *This is an individual assignment and the report and code must be your own work or cite sources.*

For concreteness, here are some guidelines:

- If you talk about the project with another student, don't keep any notes during the conversation. This way anything that comes out of the conversation goes through your brain, i.e. you have learned it.
- Write your code and report from scratch, on your own. Don't copy anything from other students.
- Don't share your code or report with other students.
- If you use git, please ensure that your repository is not public until after final grades have been released.

For the purposes of this unit, the use of generative AI (eg. copilot, ChatGPT) is treated the same as colluding with another person.

For Teams and other online discussions, please do not post solutions, either as descriptions or code. 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 Teams channels.

Please be aware that we use tools for detecting plagiarism in reports and code.