COMP3702/COMP7702 Artificial Intelligence (Semester 2, 2020)
Assignment 3: LaserTank MDP

Key information:

- **Due: 5pm, Friday 23 October**
- This assignment will assess your skills in developing continuous motion planning problems.
- Assignment 3 contributes 10% to your final grade.
- This assignment consists of two parts: (1) programming and (2) a quiz.
- This is an individual assignment.
- Both code and quiz are to be submitted via Gradescope (`https://www.gradescope.com/`). You can find instructions on how to register for the COMP3702/COMP7702 Gradescope site on Blackboard.
- Your program (Part 1, 60/100) will be graded using the Gradescope code autograder, using the testcases in the support code provided at `https://gitlab.com/3702-2020/assignment-3-support-code`.
- Your quiz answers (Part 2, 40/100) are to be answered directly in Gradescope using the Online Assignment tool (not a document upload).

The LaserTank AI Environment

LaserTank is an open source computer puzzle game requiring logical thinking to solve a variety of levels, originally developed by Jim Kindley in 1995. In LaserTank, a player controls a tank vehicle which can move forwards, turn clockwise and counterclockwise, and shoot a laser which can be used to interact with special map elements. The game takes place in a 2D map divided into grid cells. The player must navigate from a **starting position** to the **flag** in as few moves as possible while avoiding "game over" conditions. The game is over if the player moves into a dangerous cell (i.e. water cells or any cell in the line of fire area of an anti-tank).

The LaserTank codebase comes as part of your assignment support code; while a high-level description is provided in a separate LaserTank AI Environment document, which you can find on Blackboard.

LaserTank as an MDP

In this assignment, you will write the components of a program to play LaserTank, with the objective of finding a high-quality solution to the problem using various sequential decision-making algorithms based on the Markov decision process framework. This assignment will test your skills in defining a MDP for a practical problem and developing effective algorithms for large MDPs.

What is provided to you

We will provide supporting code in Python only, in the form of:

1. A class representing LaserTank game map and a number of helper functions
2. A parser method to take an input file (testcase) and convert it into a LaserTank map
3. An animated policy visualiser
4. A tester script
5. Testcases to test and evaluate your solution
6. A solver file template

The support code can be found at: `https://gitlab.com/3702-2020/assignment-3-support-code`. Autograding of code will be done through Gradescope, so that you can test your submission and continue to improve it based on this feedback — you are strongly encouraged to make use of this feedback.

Your assignment task

Your task is to develop three algorithms for computing paths (series of actions) for the agent (i.e. the Laser Tank), and to answer a quiz on your algorithms' performance. You will be graded on both your submitted **program (Part 1, 60%)** and the **quiz (Part 2, 40%)**. These percentages will be scaled to the 10% course weighting for this assessment item.

The provided support code formulates LaserTank as an MDP, and your first task is to submit code implementing all three of the following MDP algorithms:

1. Value Iteration (VI)
2. Policy Iteration (PI)
3. Monte Carlo Tree Search (MCTS).

The support code will include methods allowing the autograder to verify and evaluate each algorithm separately. The testcase files indicate which algorithm should be applied to them. After you have implemented and tested the algorithms above, you are to complete the questions listed in the section "Part 2 - The Quiz" and submit them directly into Gradescope's Online Assignment tool (nb. *not* a document upload; this differs from Assignments 1 and 2).

More detail of what is required for the programming and quiz parts are given below. Note that:

- The testcases for PI and MCTS are considerably more difficult to effectively solve than the testcases for VI, which will typically fail on the harder cases.
- The transition function supplied does not handle the testcases for MCTS, so PI and VI cannot be used to solve them with the code supplied.

Part 1 — The programming task

Your program will be graded using the Gradescope autograder, using the COMP3702 testcases in the support code provided at `https://gitlab.com/3702-2020/assignment-3-support-code`.

Interaction with the testcases and autograder

We now provide you with some details explaining how your code will interact with the testcases and the autograder (with special thanks to Nick Collins for his efforts making this work seamlessly).

Implement your solution using the supplied `solver.py` Template file. You are required to fill in the following method stubs:

- `run_value_iteration()`
- `run_policy_iteration()`
- `get_offline_value()`
- `get_offline_policy()`
- `get_mcts_policy()`

You may add to the `__init__` method if required, and can add additional helper methods and classes (either in `solver.py` or in files you create) if you wish. To ensure your code is handled correctly by the autograder, you should avoid using any try-except blocks in your implementation of the above methods (as this can interfere with our time-out handling). Refer to the documentation in `solver.py` for more details.

Grading rubric for the programming component (total marks: 60/100)

For marking your programs, we will use six different testcases to verify and evaluate each of the three different algorithm implementations you submit. Marks will be allocated according to the following rules:

- Each algorithm implementation is graded based on performance in each testcase separately.
- *Solving* a testcase means finding a *path* within the given time limit and cost bound (time limits and cost benchmarks are given in each test case file).

- If your code solves a testcase for a given algorithm, it will receive 10 marks. *Approximately solving* a testcase means finding *a sub-optimal solution path*, which is more expensive than the cost benchmark, within the given time limit.
- Approximate solutions are penalised in proportion to how far they are below the cost bound at a rate of 1 mark for each unit of cost it is below the provided benchmark cost, to a minimum of 0 marks.

Students in COMP3702 and COMP7702 face the same grading rubric in this assignment.

Part 2 — The quiz

The quiz tests your understanding of the methods you have used in your code, and contributes 40/100 of your assignment mark.

Question 1. Consider testcase `extra_t1.txt`, and explore how the policy of the agent changes with p and $game\_over\_cost$. *Hint: This question can be completed using VI.*

a) (5 marks) Fix $game\_over\_cost = -20$ and vary $p = \{0.5, 0.7, 0.9\}$. For each value of p, is the agent's optimal policy to take the safe path around the obstacle or the risky path along the cliff edge (or something else)?

b) (5 marks) Fix $p = 0.9$ and vary $game\_over\_cost = \{-15, -20, -25, -30\}$. For each value of $game\_over\_cost$, is the agent's optimal policy to take the safe path around the obstacle or the risky path along the cliff edge?

c) (10 marks) Generate a plot of the $(game\_over\_cost, p)$ indicating whether the agent's optimal policy is to traverse the safe or risky route, using colours to denote which path is optimal for each combination.

Question 2. Consider testcase `vi_t0.txt`/`pi_t0.txt` (the testcases are the same apart from solving method), and compare VI and PI according to the following measures:

a) (5 marks) Time to converge to the lower bound given for the testcase.

b) (5 marks) Number of iterations to converge to the lower bound given for the testcase.

c) (5 marks) Solution quality of VI and PI after 1 second of compute time.

d) (5 marks) Solution quality of VI and PI after 10 iterations.

Academic Misconduct

The University defines Academic Misconduct as involving "a range of unethical behaviours that are designed to give a student an unfair and unearned advantage over their peers." UQ takes Academic Misconduct very seriously and any suspected cases will be investigated through the University's standard policy (`https://ppl.app.uq.edu.au/content/3.60.04-student-integrity-and-misconduct`). If you are found guilty, you may be expelled from the University with no award.

It is the responsibility of the student to ensure that you understand what constitutes Academic Misconduct and to ensure that you do not break the rules. If you are unclear about what is required, please ask.

It is also the responsibility of the student to take reasonable precautions to guard against unauthorised access by others to his/her work, however stored in whatever format, both before and after assessment.

In the coding part of this assignment, you are allowed to draw on publicly-accessible resources, but you must make reference or attribution to its source, by doing the following:

- All blocks of code that you take from public sources must be referenced in adjacent comments in your code.
- Please also include a list of references indicating code you have drawn on in your `solution.py` docstring.

However, you must not show your code to, or share your code with, any other student under any circumstances. You must not post your code to public discussion forums (including Piazza) or save your code in publicly accessible repositories (check your security settings). You must not look at or copy code from any other student.

All submitted files (code and report) will be subject to electronic plagiarism detection and misconduct proceedings will be instituted against students where plagiarism or collusion is suspected. The electronic plagiarism detection can detect similarities in code structure even if comments, variable names, formatting etc. are modified. If you collude to develop your code or answer your report questions, you will be caught.

For more information, please consult the following University web pages:

- Information regarding Academic Integrity and Misconduct:
  - https://my.uq.edu.au/information-and-services/manage-my-program/student-integrity-and-conduct/academic-integrity-and-student-conduct
  - http://ppl.app.uq.edu.au/content/3.60.04-student-integrity-and-misconduct
- Information on Student Services:
  - https://www.uq.edu.au/student-services/

Late submission

Students should not leave assignment preparation until the last minute and must plan their workloads to meet advertised or notified deadlines. It is your responsibility to manage your time effectively.

Late submission of the assignment will **not** be accepted. Unless advised, assessment items received after the due date will receive a zero mark unless you have been approved to submit the assessment item after the due date.

In the event of exceptional circumstances, you may submit a request for an extension. You can find guidelines on acceptable reasons for an extension here https://my.uq.edu.au/information-and-services/manage-my-program/exams-and-assessment/applying-extension All requests for extension must be submitted on the UQ Application for Extension of Progressive Assessment form at least 48 hours prior to the submission deadline.