COMP6247 Reinforcement and Online Learning
21/22 Coursework

The coursework will consist of two parts, namely, a literature survey and a dynamic maze solving project. Each part will be weighted 50% of the total mark for this course. Each part of the coursework needs to be conducted individually.

Part 1: Literature Survey

In this part, you will pick a problem that interests you, search the literature for reinforcement learning approaches to tackle this problem, and survey and discuss the relative strengths of each approach. You will need to submit a report for this part, which addresses the following items.

- Introduction
  - What is the problem?
  - Why is it an important problem?
- Survey
  - You should read about 10 recent papers (published after 2017) which develops/uses reinforcement learning approaches to solve this problem
  - Organize and summarize these techniques through highlighting their strengths and weaknesses. This summary should not be a laundry list of techniques. It should be well organized based on desirable properties of the techniques
- Analysis
  - What is the state of the art?
  - Any open problem?
- Conclusion
  - What have you learned?
  - What future research do you recommend?

Part 2: Dynamic Maze Solving Project

It involves writing programs for the task described below. The purpose of this part is to adopt (use or modify) a reinforcement learning (RL) technique to solving a dynamic maze problem to compute the minimum time path from the top left corner to the bottom right corner of the maze.

- Consider a maze of size 199*199, whose outside are walls. You need to compute the minimum time path from the top left corner (1,1) to the bottom right corner (199,199) of the maze. There is a binary number associated with each location of the maze, with 1 being an empty location and 0 being a wall (i.e., a blockage you cannot pass), respectively. In addition, there could be fire randomly distributed on some empty locations of the maze. This means that you need to handle a dynamic maze.
- You can choose to solve this problem based on any RL technique such as PPO, Q-Learning, A2C, etc.
- You can choose any programming language for this project. However, your code needs to read a python file (read_maze.py) which describes the environment of the game. In the beginning of your code, you need to call load_maze in read_maze.py to load the maze from COMP6247Maze20212022.npy into a global variable maze_cells. Note that you are not allowed to directly use maze_cells in

your code and you can only call load_maze one time in your code. You are not allowed to directly read COMP6247Maze20212022.npy in your code.

- You can call get_local_maze_information(x, y) to get the situations of eight locations surrounding the location (x,y). Note that this is the only interface you can get any information about the maze in your code. This function returns an array called around, whose elements correspond to different locations centred at (x,y) as follows.

```
# [around[0][0], around[0][1], around[0][2]
# around[1][0], (x, y) , around[1][2]
# around[2][0], around[2][1], around[2][2]]
```

around[i][j][0]=0 means that it is a wall while around[i][j][0]=1 means that it is possibly empty subject to the fire. As mentioned before, there would be random fire on some locations. For a surrounding location specified by around[i][j][0]=1, you also need to read the value of around[i][j][1]. If this value is 0 (around[i][j][0]=1 and around[i][j][1]=0), it means that there is no fire. Otherwise (around[i][j][0]=1 and around[i][j][1]>0), you cannot visit that location. The fire on that location will be extinct after around[i][j][1] time units. Note that each time you can only choose to go left, up, right, or down if the corresponding location is empty with no fire. If you are at (x,y), those locations are specified by around[1][0], around[0][1], around[1][2], and around[2][1], respectively. You can also choose to stay in the current location.

- Note that whenever you call get_local_maze_information(x, y), you might induce fire on some locations surrounding (x,y). That is, observing the environment would possibly change the environment.
- Your code should compute the path with the minimum traversal time from the top left corner (1,1) to the bottom right corner (199,199) of the dynamic maze, respecting those randomly located fire. Your code needs to output that path together with the timing information. The output file also needs to show your trace in the environment, i.e., at each time unit, what you observed surrounding you and what action you took.
- In the report, you should give a clear description of the problem, and your algorithm.
- You need to discuss the performance of your algorithm and analyse the result.
- You should make good use of a software repository (e.g., git or github). Your code needs to be maintainable and designed with a make-based build structure. How to compile and run the code needs to described in the report.
- Your code should contain sufficient comments.
- Your code should be laid out neatly with consistent indentation.

Assessment criteria

Quality of part 1 report 50%

- Problem Description (5%)
- Survey (20%)
- Analysis (20%)
- Conclusion (5%)

Correctness of the algorithm (part 2) 25%

Clarity of program and comments (part 2) 10%
Quality of part 2 report 15%

Submission
Due Date: May 23, 2022
You are required to submit the part 1 report (Part1Report.pdf) and the part 2 report (Part2Report.pdf), the source code files (for part 2), and your output file (for part2), into the submission system.