Griffith UNIVERSITY
Queensland, Australia

Assignment 1

2802ICT Intelligent System
School of ICT, Griffith University
Trimester 1, 2023

Instructions:

- **Due:**

  Milestone 1 (code only) is due on 11:59pm Friday **31st March 2023**
  Peer review demonstrations on **Week 5**
  Milestone 2 (peer review report) is due on 11:59pm Monday **17th April 2023**
  Milestone 3 (full code + report) is due on 11:59pm Monday **24th April 2023**

  with demonstrations to be held on **Week 7** (In-person) and **Week 8** (online)

- **Marks:** 100 marks - 50% of your overall grade

- **Late submission:** Late submission is allowed but a penalty applies. The penalty is defined as the reduction of the mark allocated to the assessment item by 5% of the total weighted mark for the assessment item, for each working day that the item is late. A working day is defined as Monday to Friday. Assessment items submitted more than five working days after the due date will be awarded zero marks.

- **Extensions:** You can request for an extension of time on one of two grounds: medical or special circumstances ((e.g. special family or personal circumstances, unavoidable commitments).

  All requests must be made through the myGriffith portal.

- **Individual Work:** You must complete this assignment individually and submit an authentic work. This should be your own work. Anyone caught plagiarising will be penalised and reported to the university.

- **Presentation:** You must present/demonstrate your work (milestone 1 - peer review, milestone 3 - one on one interview with a teaching staff member). Your work will not be marked if you do not present it.

Maximum marks are:

- **20** for **Milestone 1**
- **10** for **Milestone 2**
- **50** for **Milestone 3 (30 - code, 20 - report)**
- **20** for **interview**

Total: 100 marks

- **Objectives:**
  - ✓ Formulate a given problem
  - ✓ Demonstrate AI programming skills
  - ✓ Carry out empirical evaluations of different algorithms

Overview

The purpose of the assignment is to assess your ability to implement various search strategies, to refine them and study their results by running experiments. Please make sure that all code that you submit is your own and is not taken/copied from anywhere else.

The algorithms should be implemented in Python only. Your code should be easy to follow. You will need to put your findings and analysis in a neatly written up report.

“Help the Robot” problem

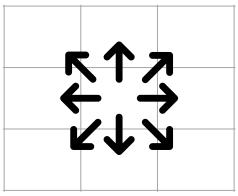
In this assignment, you are going to explore different search strategies by developing an intelligent agent that finds a path for a robot in a map.

A topographic map is described by an NxN board. Each square of the board represents the type of a corresponding area cell, which can be *normal ground* or *cliff*.

The goal is to find the an optimal path from a starting point to a destination point using different search strategies[1].

The rules:

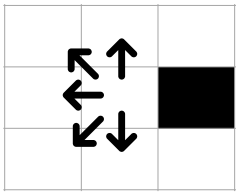
1. The robot may move one cell at a time to one of the eight cells surrounding it:



2. If a cell is a cliff, the robot cannot move to that cell.

3. When one of a diagonal’s directions is a cliff, the robot cannot move on that diagonal (for example, the robot cannot walk "Up-Left" if either the Up or Left cells are a cliff).

   For example, in the following image the black square represents a cliff. Hence, the robot can only move to one of the following five cells:



The cost:

The robot's wheels are designed in such a way that travelling diagonally is the simplest motion. therefore it just costs **one**.

Moving in one of the four main directions (Up, Down, Left, and Right) costs **two**.

---

[1] An optimal path is a legal path with minimum cost. Note that not all algorithms will necessarily result in an optimal path

The input:

The code will read all input from a single text file named **input.txt** (See example on L@G).

The first line of the file specifies which algorithm to run (1- BFS, 2-UCS, 3-IDS, 4-A*, 5-Local Search):

The second line of the file is the size of the board (N).

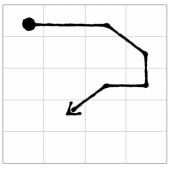
The rest of the file describes the board using to the following keys:

Symbol in the input file	Meaning
R	Normal ground
X	Cliff
S	Starting point
G	Goal point

The output:

1. You should create a text file called **output.txt**, and display your results in that file (see example on L@G)
2. The following details should be included in the file using the following format:
    a. the solution found by the algorithm, presented as a sequence of moves separated by a dash (-). The eight potential moves are L (left), UL (up-left diagonal), U (up), UR (up-right diagonal), R (right), DR (down-right diagonal), D (down), and DL (down-left diagonal).

       For example, the sequence **`R-R-DR-D-L-DL`** represents the path: right, right, down-right, down, left, down-left.

       If no solution found you should print "**`no path`**"

    b. the total path cost

    c. the path length (number of moves in the path)

    d. number of explored states

    e. a visualisation of the path (if found) - *this can be done by simply replacing the letter 'R' with 'o' in the original board (see the example output.txt on L@G). You are free to choose another way to visualise a path, as long as the path and board are easy to understand*.

    f. any other statistics you want to add - feel free to add here

Requirements

Software:

1. Implement a program that reads a problem from an input file **input.txt** as described above

2. For each problem you need to find a solution, or report of a failure to find one, using the requested algorithm. The algorithms you need to support are:

   (1) Breadth First Search (BFS)

   (2) Uniform Cost Search

   (3) Iterative Deepening

   (4) A*

   (5) Local Search - *more information about the algorithm you'll need to implement will be release by week 4*.

3. Output your results to the **output.txt** file as described above.

Experiments:

4. Compare and contrast the different algorithms. For example, compare the solution found, number of explored states, different heuristics when applicable, etc.

5. You should test your code on a variety of test cases, including cases where the goal point is not reachable from the starting point.

You are encouraged to test your program and think of experiments you can perform to assess the effectiveness and efficiency of your algorithms. *This will be marked as part of your report (see below).*

Report:

6. Provide a problem formulation for the "help the robot" problem.

7. Provide a detailed description of your code and how it works. Include details on key functions and data structures you are using in your program.

8. Report all of your experiments and results in a neatly and clear way.

9. Discuss the results, include any observations or insights you gained from running the tests.

10. Write a conclusion paragraph that summaries and explains your findings.

Submission

Milestone 1 - code only - BFS:

For the first milestone, you need to hand in the following in **one zip file**[2]:

a. **Source code** (*.py file) that only supports the Breadth-First-Search algorithm. The other algorithms don't need to be implemented just yet.

b. A copy of your source code in one of the following formats: Microsoft Word (.doc/.docx), Plain text (.txt), or Adobe PDF (.pdf).

* Make sure your code is properly documented with comments to clarify key parts.

---

[2] The name of your zip file should be: `firstname_lastname_sNumber_A1M1.zip` (for example, `Bart_Simpson_s123456_A1M1.zip`)

Milestone 2 - Peer-review:

During the workshop on week 5 you will need to:

a. Demonstrate your work (milestone 1 submission) to a small group of peer students.

b. Review the code of the other students in your group.

c. Submit a feedback report. A template for this report will be provided via L@G before week 5.

Milestone 3 - full code + report:

For the final milestone you should hand in the following in **one zip file**[2]:

a. Your full **source code** (*.py file/s)

b. A copy of your source code in one of the following formats: Microsoft Word (.doc/.docx), Plain text (.txt), or Adobe PDF (.pdf).

c. Example input files and their corresponding output files (at least one example per algorithm). Be sure that it is easy to match an output to its input by looking at the file names.

d. Your **report** as per requirements 6 to 10 above, in Microsoft Word (.doc/.docx) or Adobe PDF (.pdf) format.

* Make sure your code is properly documented with comments to clarify key parts.

* REMEMBER: You will need to present your work in a one-on-one interview with one of the teaching staff.

Submission should be done through the link provided in L@G by the due date.