Griffith UNIVERSITY
Queensland, Australia

Assignment 1

2802ICT Intelligent System
School of ICT, Griffith University
Trimester 1, 2021

Assignment 1 (50%)
Milestone 1 is due on 11:59pm Friday 26th March 2021
Milestone 2 is due on 11:59pm Friday 16th April 2021

with demonstrations to be held on Wednesday Week 6.

Instructions:

- **Due: Milestone 1** - 11:59pm, 26th March**, Milestone 2** - 11:59pm, 16th April
- **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
  - other (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. You are encouraged to discuss the assignment with your fellow students, but eventually this should be your own work. Anyone caught plagiarising will be penalised and reported to the university.
- **Presentation:** You must present/demonstrate your work on Week 6 to a teaching team member. Your work will not be marked if you do not present it. Indeed, the final marks will be based on the interview results as follows:

```
finalMark = interviewMark * submissionMark + oralExaminationMark
```

Note: `submissionMark` is your mark of the submitted code and report. `interviewMark` is out of 1 based on how well the code and report are explained. `OralExaminationMark` is your mark of your answers to the open questions you'll be asked during the interview.

Maximum marks are:

- **5** for **Milestone 1**
- **20** for the **code**
- **15** for the **report**
- **10** for **open questions**
- **Total: 50 marks**

- **Objectives:**
  - ✓ Formulate a given problem (milestone 1)
  - ✓ Demonstrate AI programming skills (milestone 1 & 2)
  - ✓ Carry out empirical evaluations of different algorithms (milestone 2)

Overview

The purpose of the assignment is to assess your ability to implement various search algorithms, 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.

Solving the Rush Hour Game

In this assignment, you are going to explore different search strategies by developing an intelligent agent that solves the puzzle game '**Rush Hour**'.

You can find the game description here and you can also play it online here.

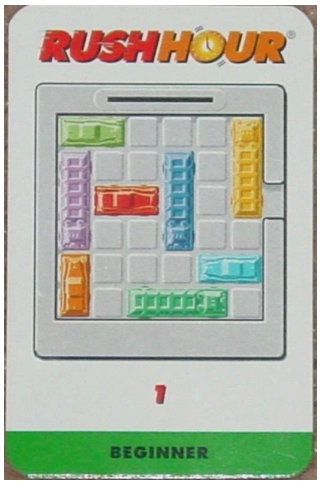
The game:

1. The game consists of a 6x6 grid (the board) and a set of cars and tracks positioned on the grid. Cars are of size 1x2 and trucks are 1x3.
2. Cars are marked with the letters A - K and X
3. Tracks are marked with the letters O,P,Q,R
4. In each problem/challenge, (a varied number of) cars and tracks are placed on the board in a specific arrangement.
5. The goal of the game is to clear a path for the red car (marked with X) out of the board to the right, by moving away the other vehicles.
6. The player can move the cars and trucks in their lanes - up and down, or left and right (according to their initial orientation).

The input:

The file `rh.txt` includes 40 different challenges (cards) in an increasing difficulty level. Each challenge is given as a 36 character string represents the 6x6 grid, where letter represents a car/ truck occupies a square, and dot ('.') represents an empty square.

For example, the string "AA…OP..Q.OPXXQ.OP..Q..B…CCB.RRR." represents the challenge/card:



The input consists of:

1. rh.txt file with 40 challenges strings
2. **maximal time** for solving each problem (as a parameter).

For your convenience, the file **rh.txt** also includes proposed solutions (not necessarily optimal but very close to) for each of the 40 challenges. Each solution is in a form of list of strings such as:

CL3 OD3 AR1 PU1 BU1 RL2 QD2 XR5

Each string in the solution list includes three characters: the first is the vehicle label, the second is the moving direction (L=left, R=right, U=up, D=down) and the third is number of squares to move. For example, the above list means: move car C 3 squares to the left, then move track O 3 squares down, etc.

You will use these solutions to test your code.

The output:

1. You should provide a txt file (**output.txt**) that contains a solution list for each of the given challenges/problems in the input file (40 lines respectively). If the program fails to solve a problem within the predefined maximal time, print out "FAILED" instead of a solution list in the appropriate line.

2. Also, your program should print out (to the screen) for each challenge/problem *at least* the following data:

    a. The initial board

    b. The proposed solution (from the input file)

    c. The solution your algorithm found, or FAILED if case of a failure

    d. The Execution (CPU) time for reaching the solution/failure

    e. The depth of the solution in the search tree

    f. The difference (in length) between the proposed solution and your solution

    g. Number of searched nodes (states)

Requirements:

Software:

1. Implement a program that reads a set of 40 problems and proposed solutions from an input file *rh.txt*

2. For each such problem you need to find a solution, or report of a failure to find one using one of the following algorithms (you will need to implement them all):

    a. BFS

    b. Iterative Deepening

    c. A*

    d. Hill Climbing

3. Print the solution to an output text file

4. Print out (to the screen) the 40 problems with their proposed solutions and your solutions, plus the statistics described above.

Experiments:

5. Find an optimal maximal time for finding as many solutions as possible.

6. Compare and contrast the different types of algorithms. For example, compare the average time for finding a solution, number of successful/unsuccessful solving tries, distance from the proposed solutions, compare the algorithms' performance with different game level categories: beginners, intermediate and advanced, etc.

In this part you are encouraged to be creative and think of experiments you can perform to test the effectiveness and efficiency of your algorithms.

Report:

7. Provide a problem formulation for solving the rush hour game.

8. Provide a detailed software design that includes details on functions and data structures you are using in your program.

9. Report all of your results from section 6 in a neatly and clear way.

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

Milestone 1 - Problem Formulation & Uninformed Search:

For the first milestone you should hand in the following in one zip file:

- A report that includes the problem formulation (requirement 7) and software design (requirement 8) for the parts you have implemented so far (at least sections a-d below).
- Well documented source code that includes:

    a. Reading a set of 40 problems and proposed solutions from the input file *rh.txt*

    b. Printing out (to the screen) the initial board and proposed solution for each of the 40 problems

    c. Implementation of a single state

    d. Implementation of the *expand* operation (i.e. create new states from an existing one)

    e. Ideally, at this point you will also complete the implementation of the uninformed search algorithms BFS and Iterative deepening. If not, you will need to complete this for Milestone 2.

Milestone 2 - Informed Search & Local Search:

For the second milestone you should hand in the following in one zip file:

a. Your (well commented) source code (*.py file).

b. A readme file explaining how to run your program and what output to expect.

c. A detailed report explaining your algorithms and experimental results according to requirements 7 to 10 above

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

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

For further marking details please refer to the marking rubric.