CS3243: Introduction to Artificial Intelligence **(Due: September 20, 2019; 2359hrs)**

Assignment 1: 8-Puzzle

Important: Read all the instructions below carefully before you start working on the assignment, and before you make a submission.

- You must do this assignment in groups of two or three (lone-wolves are discouraged). Team members must be from the same tutorial group. Please write the names and matriculation numbers of your team members in the report and the code. Please submit no more than one submission per group.
- All sources of material must be cited. The University Academic Code of Conduct will be strictly enforced.
- Please zip your two python files and report, and submit it via LumiNUS folder.
- The python files' name should be `puzzle_A1_xx_1.py` and `puzzle_A1_xx_2.py`, your report filename should be `report_A1_xx.pdf`, and the zip file should be name as `puzzle_A1_xx.zip` where `xx` is your group number (e.g. `puzzle_A1_03.zip`, note that it is 03 and not 3). Please follow the file naming system closely. Points will be deducted for not following the naming convention.
- Kindly check that the submitted code will be able to run on SoC's Sunfire account. We will be using Sunfire (our UNIX server) to evaluate all submissions.
- Sign up your group on LumiNUS → Class and Groups → Class Groups.
  1. Tutorial 2: Groups A1-01 to A1-09
  2. Tutorial 3: Groups A1-10 to A1-20
  3. Tutorial 4: Groups A1-21 to A1-31
  4. Tutorial 5: Groups A1-32 to A1-42
  5. Tutorial 6: Groups A1-43 to A1-53
  6. Tutorial 7: Groups A1-54 to A1-64
  7. Tutorial 8: Groups A1-65 to A1-70
  8. Tutorial 9: Groups A1-71 to A1-81
  9. Tutorial 10: Groups A1-82 to A1-87

1 Introduction

In this assignment, you will be implementing two different search algorithms of your choosing on the 8-puzzle game introduced in the lecture. At the end of this assignment, you should be able to identify and comment on the difference in performance of the algorithms.

2 Background

The 8-puzzle is a sliding puzzle that consists of a frame of numbered square tiles in random order with one tile missing. The objective of the puzzle is to arrange the tiles in order by making sliding moves that use the empty space. Figure 1 shows a sample initial state for 8-puzzle.

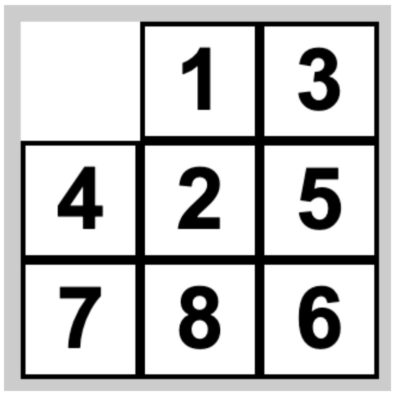

Figure 1: A sample initial state

3 Problem Statement

An initial state will be given as input. Your job is to solve the puzzle to reach the goal state (see Figure 2) and output the **actions**. Notice that some initial states may not be solvable. For example, Figure 3 shows an unsolvable initial state. In this case, you only need to output **UNSOLVABLE**.

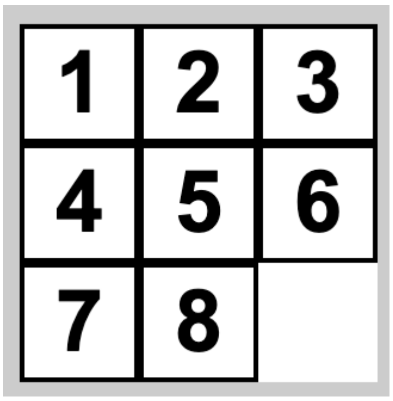

Figure 2: Goal state

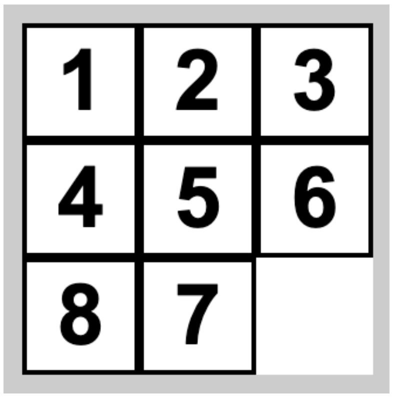

Figure 3: An unsolvable initial state

3.1 Input

The input will be provided in a text file containing three lines to represent the initial state. Each line contains three numbers. They must include the numbers 0–8, where 0 represents the empty space, and 1–8 represents the 8 tiles of the puzzle. For example, the initial state in Figure 1 is encoded in the text file as follows:

```
0 1 3
4 2 5
7 8 6
```

3.2 Output

- Your output should be written to a file in the form of a **List**. If the initial state is Figure 1, one possible output could be [**"LEFT", "UP", "LEFT", "UP"**]. This will be stored in the file as shown below.
  `[''LEFT'', ''UP'', ''LEFT'', ''UP'']`
- If the input puzzle is not solvable, output [**"UNSOLVABLE"**]. Otherwise, output the action of each step. The action could only be **"UP"**, **"DOWN"**, **"LEFT"**, **"RIGHT"**, to represent the direction of moving the tile towards the space.
- You should ensure each action is valid (e.g., you **CANNOT** move **DOWN** when the space is in the first line) and the puzzle should be the same to the goal state after the last action.

4 Submission

Your submissions should contain the following:
- TWO python code files (one for each algorithm), and
- ONE report in PDF.

You will lose five points for not submitting either the report or the code files.

4.1 Code

Please use Python 2.6 (the default Python version on SoC's Sunfire) to do this assignment. The template has been provided to you. You may import Python Standard libraries if you need (but no external libraries). Things to note:
- You may only change the code inside the **Puzzle** class.
- You are required to implement two different algorithms to solve the puzzle (implemented in two Python files). It is also possible to use the same algorithm but with different heuristics in the implementation.
- **Executable**: If your program can not be executed, you will get 0 for your code components.

- **Correctness**: Please make sure your algorithm logic is correct. You will lose mark significantly if you can not pass our test cases.
- **Number of moves**: Ensure that the solution is within 300 moves.

4.2 Report

Your report must be TYPE-WRITTEN (font size not exceeding 12pt). It should NOT be more than 2 pages. Your report should contain at least these components:

- **Explanation**: Explain briefly the algorithms and/or heuristics you used in your solution and **why did you choose it**.
- **Statistics:** For the given inputs, please report the following statistics in your report.
  - Number of nodes generated.
  - Maximum size of the frontier that is reached.
- **Analysis**: Report if the algorithm you use is *complete* and *optimal.* If you use a heuristic method, you may need to prove it is consistent or at least admissible. Using the statistics above, discuss the time complexity and memory consumption, and compare the performance of the two algorithms that you implemented (using the provided inputs and your own inputs). You can also elaborate on other points that you think is valuable to discuss.

5 FAQ

1. Can we implement the same algorithm?

A. No.

2. Can we implement the same algorithm with different heuristic?

A. Yes.

3. Are we restricted to the algorithms discussed in the lecture?

A. No, you are free to choose any search algorithm. However, you need to explain the algorithm precisely and concisely in the report.

4. Can we use a different heuristic?

A. Yes, you can! You need to show how it is admissible/consistent/both in the report.

5. Is the page length limitation fixed?

A. Yes. (Sort of!) If the report spans beyond 2 pages, we are not obliged to read the entire report. Grading will be based on the first two pages only. You can use additional pages to include references and algorithm listing (if you want to).

6. How/where do we submit the assignment?

A. We will open up a submission folder on LumiNUS. Only one person from the group has to submit.