National University of Singapore
School of Computing
CS3243 Introduction to AI

Project 1: An Evaluation of Uninformed and Informed Search Algorithms on the *k*-puzzle Problem

Issued: January 29, 2020 Due: 6 March, 2359hrs

Objectives

The aims of this project are to:

1. Become familiar with the implementation of uninformed and informed search algorithms through their application on a simple problem.
2. Analyse the correctness and complexity of the aforementioned search algorithms.
3. Design experiments to further empirically evaluate the aforementioned search algorithms.
4. Discuss the effectiveness of the implemented algorithms on the *k*-puzzle problem.

Introduction: The *k*-puzzle Problem

The *k*-puzzle problem is a generic version of the 8-puzzle problem, which is described in detail in the **AIMA text**.

2	3	6
1	5	8
4	7	

Figure 1: An example of an initial state for the 8-puzzle problem.

In the 8-puzzle version, you are given a 3×3 grid, where all but one grid cell contains a unique number between 1 and 8 (inclusive), and the last cell is left empty. An example of this is depicted in Figure 1.

The goal is to move the cells such that all the cells are ordered. More specifically, for the 8-puzzle to be solved, all the cells must be identical to the positions depicted in Figure 2.

1	2	3
4	5	6
7	8	

Figure 2: An example of an initial state for the 8-puzzle problem.

The only valid moves in the *k*-puzzle problem correspond to the movement of a numbered cell to a neighbouring cell that is empty. Thus, given the example in Figure 1, a solution may be achieved by taking the following sequence of valid moves:

$$8\downarrow \quad 6\downarrow \quad 3\rightarrow \quad 2\rightarrow \quad 1\uparrow \quad 4\uparrow \quad 7\leftarrow \quad 8\leftarrow$$

For the generic *k*-puzzle, you are to assume that the grid has dimensions $n\times n$, where $k = n^2 - 1$, $n \geq 2$ and $n \in \mathbb{Z}$.

General Project Requirements

The general project requirements are as follows:

- Group size: **3-4 students**
- Submission Deadline: **6 March 2020** (Friday), **2359 hours** (local time)
- Submission Platform: **LumiNUS** $>$ **CS3243 Module** $>$ **Project Submission Folder**
- Submission Format: One standard (non-encrypted) **zip file**[1] containing all necessary project files

[1]Note that it is the responsibility of the students in the project group to ensure that this file may be accessed by conventional means.

As to the specific project requirements, you must complete and submit the following:

1. An implementation of **ONE** uninformed search algorithm described in **Chapter 3.4** of the **AIMA Text**.
2. An implementation of **ONE** informed search algorithm described in **Chapter 3.5** of the **AIMA Text**.
3. Design of **THREE** admissible heuristics for the chosen informed search algorithm in **Point 2**.
4. Design of **experiments to empirically evaluate** the 1 uninformed search algorithm and chosen informed search algorithm with each of the three heuristics.
5. A **2-page report** that describes the following:

   a Problem specification

   b Technical analysis of the selected algorithms and heuristics

   c Experimental setup to evaluate the selected algorithms and heuristics

   d Results and Discussion

   Note that your report may include an extra page for references. An unlimited number of appendix pages may be included, though will not be assessed.

Do note that any material used within the report that does not originate from you (i.e., is taken from another source), should be re-written in your own words and cited. Failure to do so may constitute plagiarism.

For projects submitted beyond the submission deadline, there will be a 20% penalty per day. For example, if you submit the project 30 hours after the deadline (i.e., 2 days late), and obtain 92%, a 40% penalty applies, and you will only be awarded 52%.

Submission Specifications

Your submissions should contain the following:

- **FIVE** Python code files (one for uninformed search, three for informed search with 1 heuristic each, and one for the experiment), and

- **ONE** report in PDF format.

The files mentioned above should be named:

- **`CS3243_P1_XX_1.py`** - for your uninformed search implementation
- **`CS3243_P1_XX_2.py`** - for your informed search implementation using the 1st heuristic
- **`CS3243_P1_XX_3.py`** - for your informed search implementation using the 2nd heuristic
- **`CS3243_P1_XX_4.py`** - for your informed search implementation using the 3rd heuristic
- **`CS3243_P1_XX_5.py`** - for your empirical experiments
- **`CS3243_P1_XX.pdf`** - for your report

The zip file should be named **`CS3243_P1_XX.zip`**. For all the filenames listed, **`XX`** refers to your assigned group number. For example, **`CS3243_P1_03.zip`**. Note that any single-digit group numbers should be named with **`03`** and not **`3`**.

Failure to adhere to these naming conventions will result in loss of marks.

Input

The input is provided in a text file, which will contain n lines, with n integers on each line. Each such integer included will correspond to one integer from the sequence 0 to $(n^2 - 1)$, with 0 representing the empty cell. This will correspond to the initial state of the *k*-puzzle. For example, the initial state given in Figure 1 would be encoded in the text file as follows:

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

Output

The output of your code should correspond to a List containing the strings: `'LEFT'`, `'RIGHT'`, `'UP'`, `'DOWN'`, and `'UNSOLVABLE'`. For example, the output specified as the solution to the initial state given in Figure 1, would be: `['DOWN', 'DOWN', 'RIGHT', 'RIGHT', 'UP', 'UP', 'LEFT', 'LEFT']`.

If the given puzzle (input) is not solvable, your output should be `['UNSOLVABLE']`. Else, you should output the actions of each step required to solve the puzzle, where each step here corresponds to the movement of the non-empty cell (i.e., the non-zero cell).

You must ensure that all moves performed are valid. For example, you may not specify `'DOWN'` if the empty cell (i.e., `0` cell) is on the top row. Further, the sequence of moves specified must bring the puzzle to the goal state, with the empty cell in the bottom-right corner.

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 necessary. However, you may not use any external libraries. In addition, please note the following:

- You may only change the code inside the Puzzle class for each search implementation file.
- You are required to implement ONE informed and ONE uninformed search algorithm, with THREE admissible heuristics chosen for the latter.
- You must also design and implement experiments to empirically evaluate your search implementations.
- Your code must be executable; if your program cannot be executed, you will get 0 for the code components.
- Your code should be correct. It must follow the given input and output specifications and pass all test cases. It must run in reasonable time.

Report

Your report must follow the LNCS template (refer to the Templates, sample files and useful links sub-section). Failure to comply with this format will result in the loss of marks.

Your report should at least be 2 pages long, and no longer than 4 pages.

It must include the following sections:

1. Problem specification

2. Technical analysis
3. Experimental setup
4. Results and Discussion

You must submit the report in PDF format.

Marking Rubrics

The following rubrics will be applied to your submission.

Marks	Requirements	Submission Evaluated
1	Uninformed search implementation (1 algorithm) • Correct implementation based on algorithm specified • Runs in reasonable time and passes all test cases • Implementation must correspond to the description given in the report	Code
1	Informed search implementation (1 algorithm including the 3 heuristics) • Correct implementation based on algorithm specified with the 3 heuristics implemented as specified • Runs in reasonable time and passes all test cases • Implementation must correspond to the description given in the report	Code

1	Implementation of experimental setup • Experimental setup implemented as specified • Generates performance results for all implemented algorithms and heuristics • Experiments should correspond to the description given in the report	Code
1	Problem specification • A suitable representation should be specified to define the problem • It should facilitate the formulation of search-based solutions	Report
1	Technical Analysis (1) • Each algorithm implemented should be analysed for correctness and complexity	Report

3	Technical Analysis (2) • Your heuristic must be provably admissible, or (preferably) consistent. In either case you must formally prove this property. Note that consistency implies admissibility but not the other way around. Thus, if you show admissibility, you should provide a (preferably simple) counterexample where your heuristic violates consistency. • The 3 heuristics should be distinct and non-trivial; for example, your heuristic should not be a constant, a simple linear transformation of other heuristics, or any simple function of another heuristic like a square root of another • 1 mark per heuristic	Report
1	Experimental Setup • The goal of the experiments should be defined (including a rationale for the chosen algorithms and heuristics), and in particular the metrics designed/used should be justified • The entire experimental process should be clearly defined such that the experiments may be replicated	Report
1	Results and Discussion • The results from your experimentation should be clearly summarised (you may include more detailed results in an Appendix, which would not count towards the given page limit – these will only be optionally viewed at the markers' discretion) • A concise discussion comparing the theoretical analysis and empirical results should be given to elucidate the effectiveness and efficiency of your implementations	Report

This project is worth 10% of your module grade.