Undergrad AI Capstone NCTU Spr2021 Programming Assignment #1 Due 4/6/2021

This assignment is for you to
(1) Learn to implement the search algorithms mentioned in the class for a non-trivial toy problem.
(2) Experiment with the algorithms to compare their performances for problems with different levels of difficulties.
(3) Learn to write a report that is well organized and concise to describe your work.

The toy problem for you to play with is the **Rush Hour Puzzle** (see the picture to the right). It is easy to find in toy stores, or in an app store for you to play it on your smart phones. If you are not familiar with this puzzle game, check out the rules on the web.

Problem setting: We will use a standard 6x6 board with a number of cars. The cells are identified using the (row,column) coordinates, with (0,0) at the top-left corner. With the only exit at the right side of row 2 (3rd row from top), the goal is to clear the way for the red car to exit.

The initial state of a game is given by the layout of the cars on the board. This will be given in plain text, with each line representing a car and consisting of its index, top-left cell (row, column), length (2 or 3), and orientation (1 for horizontal and 2 for vertical). For the example board shown above, the specification is given by

```
0  2  3  2  1
1  0  0  3  1
2  1  3  3  1
3  3  1  2  1
4  5  0  3  1
5  1  0  2  2
6  3  0  2  2
7  1  2  2  2
8  3  3  3  2
9  4  4  2  2
10  2  5  2  2
11  4  5  2  2
```

The first line (car index #0) is always the red car. The other cars can be listed in any order.

Goal state: Any state where the red car has its top-left cell at (row=2, column=4).

Solution: A sequence of car movements. Each action is specified by a tuple of <*car_index, new_row, new_column*>. You can output your solution in plain text.

Algorithms: You should implement and experiment with BFS, DFS, IDS, A*, and IDA*. Both tree-search and graph-search versions can be tried.

Heuristics: A common heuristic function for this puzzle is the "blocking heuristic", that is, the number of cars directly blocking the way of the red car to the exit. Its value for the example board above is one.

Write your program as a function. The first input should be an integer representing the type of search algorithm. The second input is the name of a text file containing the initial board layout. (Several such files will be supplied during the next few days.)

Evaluation: The most common metric is the number of expanded nodes during search. In addition, you can try to measure the actual time of execution as well as the number of nodes kept in the memory. These will require you to implement some accounting in your program.

Some more issues to consider and try:

- How to implement the explored set if you want to do graph search? Is the extra time and space worth it?
- Can you improve on the provided heuristic function, or devise new ones? If you do, be sure to make comparisons of how they affect the evaluation metrics.
- Can you implement an automatic puzzle generator so that you can do more systematic experiments, instead of using just a few sample puzzles?

You are not required to actually work on all three issues above. More in-depth discussions of one or two of them are better. You can also consider other issues not in the list above.

You submission is a report file in PDF format. The report (maximum 5 pages single-spaced) should describe your experiments and results, especially the comparison between the different algorithms. Also include sections that describe your observations, interpretations, things you have learned, remaining questions, and ideas of future investigation. Be sure to check your observations (such as how complexity grows with solution depth) against those given in lectures.

Include your program code as an appendix (not counting toward the 5-page limit), starting from a separate page. You can use C/C++, Java, Python, or MATLAB to write your program. In general, the TAs will not actually compile or run your programs. The code listing is used to understand your thoughts during your implementation, and to find problems if your results look strange. Therefore, the code listing should be well-organized and contain comments that help the readers understand your code; this will also affect your grade.

The submission is to be through New E3. Late submission is accepted for up to a week, with a 5% deduction per day.