HW 1

Arnab Dey
Student ID: 5563169
Email: dey00011@umn.edu

State Representation

I am using a number system with base of 9 to represent my state. Therefore, the state '123804765' would be encoded to as follows:

$$\begin{aligned}\text{encoded\_state} &= 1 \times 9^8 + 2 \times 9^7 + 3 \times 9^6 + 8 \times 9^5 + 0 \times 9^4 + 4 \times 9^3 + 7 \times 9^2 + 6 \times 9^1 + 5 \times 9^0 \\ &= 277857180\end{aligned}$$

Similarly I perform decoding as per the regular process of conversion to decimal from a number system with base 9.

The code snippets for encoding and decoding are implemented as static methods of *Problem* class in *utils.py* (see *encode_state()* and *decode_state()*).

Visualization

The code to visualize the encoded state as a 8-Puzzle board is implemented in *visualize()* function in *search.py*.

Problem Setup

I have followed the AIMA book's structure to build the *Problem*, *Node* classes. These implementations can be found in *utils.py*. Note that, the direction of actions are slightly different from how the book describes. There are total 4 different actions:

1. 'U': A tile is moved up into the blank space
2. 'D': A tile is moved down into the blank space
3. 'L': A tile is moved left into the blank space
4. 'R': A tile is moved right into the blank space

Uninformed Search

All the search methods are present in *search.py*. For a 8-Puzzle board the state space has 9! possible states.

I have implemented a single function *depth_limited_search()* which takes an instance of *Problem* class and a limit (integer) as arguments. When limit argument is set to *None*, it performs a depth first search and when limit is some positive integer, it performs depth limited search with depth limit set to that positive integer.

Note that, the canvas assignment page says the *iterative_deepening()* takes *state* as an argument, however, I have used a instance of *Problem* class as an argument. Literally both are equivalent but I felt keeping *Problem* instance outside of *iterative_deepening()* is better as it enables to define the goal state also outside the search method. Please let me know if you have any questions.

For iterative deepening I am using a maximum limit of 80.

Informed Search

The heuristic functions for *num_wrong_tiles()* and *manhattan_distance()* are implemented inside the *Problem* class in *utils.py*. For A* search, I have adapted the implementation of *PriorityQueue* from the AIMA book's implementation.