CSCE 775: Deep Reinforcement Learning
Coding Homework 3

Your code must run in order to receive credit.

For grading purposes, while you can add helper functions to your file, do not change the signature of the functions you should implement. These functions must accept and return exactly what is specified in the documentation.

Key building blocks:

- `env.state_action_dynamics(state, action)`: returns, in this order, the expected reward $r(s, a)$, all possible next states given the current state and action, their probabilities. Keep in mind, this only returns states that have a non-zero state-transition probability.
- `env.get_actions(state)` function that returns a list of all possible actions that can be taken in that state
- `env.states_to_nnet_input(states)` converts a list of states to a numpy array for the input to a neural network
- `env.is_terminal(state)` returns true if state is terminal

Installation

We will be using the same `conda` environment as in Homework 0.

The entire GitHub repository should be downloaded again as changes were made to other files. You can download it with the green "Code" button and click "Download ZIP" or pull from the repository.

1 Solving the 8-puzzle with a Trained DQN (100 pts)

The longest shortest path for the 8-puzzle is 31. For this exercise, you will be randomly given states from the 8-puzzle whose optimal path costs range from 0 to 31. For each state, your implementation of `search` with a trained DQN given to it as an argument. Your code should solve 100% of states, your code should find an optimal path 50% of the time or more, and the **total** time your code takes to solve **all** states (not just one individual state) should be no longer than 30 seconds. **I will also test your code on other environments with different DQNs. Therefore, your code should not be specific to the 8-puzzle.**

Running the Code

To run:
`python run_hw3.py`
When run, the code should output the results for each state and, at the end, a summary of the performance on all states.

2 Extra Credit (20 pts)

The speed of each student's code will be ranked based on fastest time and extra credit will be given to the top ranking students, given it finds an optimal path 50% of the time or more.

What to Turn In

Turn in your implementation of `coding_hw/code_hw3.py` (**the .py file, not a .zip or any other format**) to Blackboard.