ReadyPython SP19 Project

Sliding Puzzle

click here to submit

A sliding puzzle is a combination puzzle that challenges a player to slide pieces along certain routes on a board to establish a certain end-configuration. In this project, we'll write a python program that finds the optimal solution on a randomly shuffled board using the BFS algorithm we learned in class, and see how the solution works out with animations.

1. Download project files from Google Drive or OneDrive. There are easy, medium, and hard versions available, with descending amount of skeleton codes.

2. Read through the starter code in puzzle.py, especially the doc-strings, to understand the game rules and get a sense of how the project is organized.

3. In the project directory, type `python3 demo.py` to view a working demo.

4. Start working on your own project! When unsure of how your program should behave, refer to the demo program provided.

Contact me if you have any questions.

---

test.py

Once you finished implementing solve(), you may run `python3 test.py` in your project directory. It runs 50 randomized tests and reports the errors to help you debug. If you pass all 50 tests, solve() total run time will be printed. Note that the tests will be the same every time you run `python3 test.py`, as the random seed is default to 0. To run a different set of tests, type `python3 test.py *seed*`, *seed* being an integer of your choice.

Possible Extra features

1. Modify the program to allow it accept custom input boards from the user, and find the solution if there is one!
2. Allow 4 x 4, or even 5 x 5 board!
3. Implement solve() in A* algorithm! Learn about different types of searching algorithms online, or here. To turn a BFS into A*, you basically replace the queue with a priority queue. Instead of appending new states at the end of the queue, we enqueue it with (past_distance + heuristic_distance) as its priority. Write your own heuristic function: remember it has to be admissible and consistent to find the optimal solution (Hint: read the previous two links, they basically tell you the answer to our problem!). When finished, run `python3 test.py` to see if you get any performance improvement than BFS, especially on a larger 4 x 4 or 5 x 5 board.