Slider Lab 1 – AI

5 Sep 2019

In class we discussed the structure of BFS (breadth first search) and DFS (depth first search) and saw pseudocode for solving an arbitrary slider puzzle.

Slider Lab 1: You should apply BFS to solve an arbitrary 3 x 3 slider puzzle.

Implement a Python script whose first argument indicates the starting position of a slider puzzle. The second argument, if provided, is the goal state. The default goal state is `"12345678_"`. Your script should have a function `solve(puzzle, goal)` that implements the pseudo-code shown in class.

One of the helper routines that you write should be `neighbor(puzzle)`, which returns those puzzles that are a distance of 1 away from `puzzle`. The distance between two puzzles is the minimum number of steps needed to get from one puzzle to the other.

You can expect to be modifying the two above routines in the future, so please ensure that you have implemented what is stated above, the former having been shown in class.

Your script should output:

1) A sequence of states starting from the input `puzzle` up through the `goal` puzzle that constitute a minimum length path to the `goal` from `puzzle`. If the two are the same, then only print out the starting `puzzle`. The number of steps needed in this case is 0 (it takes 0 steps for a puzzle to get to itself).

Puzzles are to be printed out as 3 x 3 arrangements and not as a string of 9 characters. You should print anywhere from 5 to 12 puzzles going across, but the final horizontal grouping of puzzles may have fewer than your selected number of puzzles going across. For example, if it takes 19 steps to get to your goal and you have selected 7 puzzles in a horizontal grouping, then you would show the first 7 puzzles, then the next 7, and for the final grouping show 5.

As a second example, if your script is called as `eights.py 1234567_8 12345678_` then you would have one horizontal grouping of two puzzles in the output

2) Your script should output the test `Steps: #` where # is the number of steps in the minimal solution. If there is no solution, then output `Steps: -1`.

3) As the final line of your output, you should print out `Time: #s` where the # is the number of seconds (should have three significant digits).