**Members**

Lisa Chen

**File Listing**

* **search.py** 
  + File defining search algorithms
  + Only Depth First Search (DFS) and Breadth First Search (BFS) implemented with an added helper method I created outside of the original defined methods: search\_structure().

**What I’ve learned**

I learned a little bit more on how to code in Python and using the Terminal instead of an IDE. I was also not familiar with using tuples, so that took a while to learn how to use. It took longer than I expected to take given I’m quite familiar with the algorithms. However, the implementation was much simpler than I expected as I was able to simplify it to one method that is called by both BFS and DFS, and they just pass a stack/queue to that one method. This was because besides using different data structures, the pseudocode is near identical with using pop/push and checking for visited in the same order.

Note I updated the search.py file later (after 5pm) for optimization and comment clean-up. It doesn’t change my answers.

Question 1 Questions

Is the exploration order what you would have expected? Does Pacman actually go to all the explored squares on his way to the goal?

For depth first search, it was the order I expected, and Pacman was made so he doesn’t go to all the explored squares on his way to the goal since I implemented backtracking in the algorithm, but passed the solution to Pacman himself.

If you use a Stack as your data structure, the solution found by your DFS algorithm for mediumMaze should have a length of 130 (provided you push successors onto the fringe in the order provided by getSuccessors; you might get 246 if you push them in the reverse order). Is this a least cost solution? If not, think about what depth-first search is doing wrong.

It is not the least cost solution. Depth first search finds the first path to the goal, even if it is the longest. Because of the implementation, it follows a path until it finds out it is not the correct path before trying other paths. It thus takes the first path from start and won’t try 2nd, 3rd, etc path from the start location unless it determines the first path is impossible, unlike Breadth First Search, where the first path to hit the goal would be the shortest path through enumerating every path from the start location.

Question 1: DFS - Pacman Terminal Prints

A screenshot of text

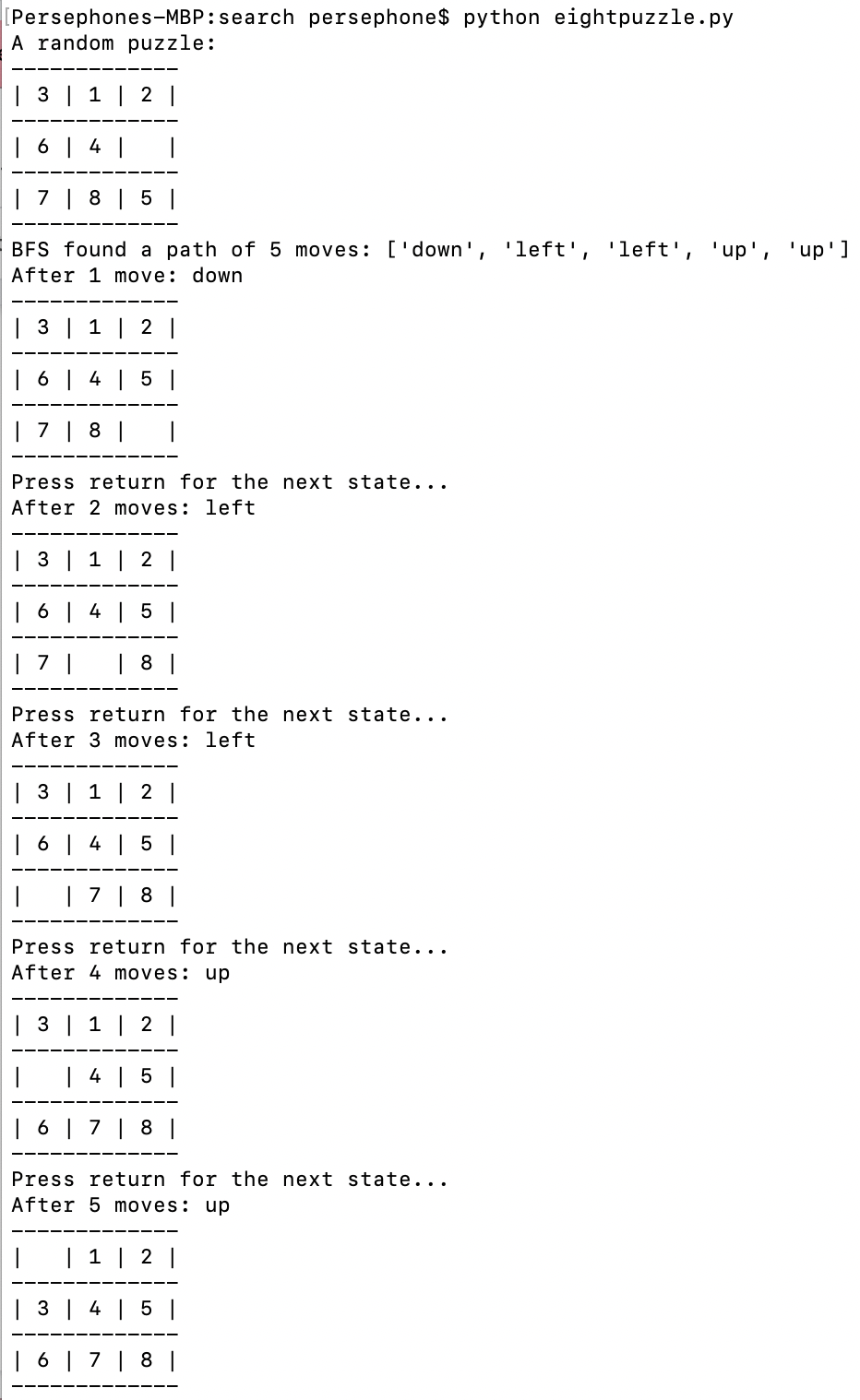
Description automatically generated

Question 2: BFS - Running Pacman Terminal Prints

A screenshot of a cell phone

Description automatically generated

Question 2: BFS - Eight Puzzle Terminal Print



Autograder Question 1

A screenshot of text

Description automatically generated

Autograder Question 2

A screenshot of text

Description automatically generated