Failure Report

Unfortunately, I was unable to fully complete the Rush Hour puzzle solver (at least to the point where I have confidence in running any random puzzle) since my code consistently runs apparently indefinitely. I was able to work out a significant portion however, as my code can get children accurately and return the correct path for simple puzzles. I estimate that my code can solve puzzles with path lengths up to 15 that have a lower number of possible states or branches. This distinction I found was important from trying a puzzle said to have a path length of only 15, but with an extremely high number of possible states. This led my code to run seemingly indefinitely.

It has become apparent that the reason for this infinitely long run time is the implementation of my get\_children method. When consulting my peers at the beginning of this, we came up with the same method of calculating and storing our states. We do this by storing each state as a list of cars, each car is a tuple containing a tuple of occupied coordinates, an identifier/name, and a direction. This allows for easy identification of which spaces are blocked and which are open. The problem with this is that inputting the initial state to solve took serious concentration and time as I needed to input every coordinate in a specific order. I later solved this by writing a method to convert a special, easy-type-string, into the necessary format for BFS. This allowed me to significantly increase the rate of case-testing.

As for the actual calculation of the get\_children method, my code calculates first all the blocked spaces in a specific row or column and isolates a list of possible moves. Later, new cars are made with this list of moves. At first, I did run into a few problems with this, as I forgot to incorporate restrictions on spaces beyond blocked spaces. This mistake led to the target car being able to jump cars. However, to fix this, I relied on a complicated set of for loops that could be the culprit of the runtimes.

What pains me the most is that my code theoretically works the same as my peers, and my peers successful output serves as a testimony to the potential of the implementation. I was not able to get in any serious debugging due to time restraints from other homework and activities, and so I’ve been formulating ways to improve the efficiency of my code in different ways. I optimized my original code by rearranging necessary and removing unnecessary for-loops. Still not working, I rewrote code to check only the target row or column to calculate moves. Currently, I am still in the processes of debugging this, however, I believe the next step is to move onto an entirely different way of storing my data. A way that seems to work much better than tuples of tuples of tuples, is to store the puzzle as a string. Although much harder to implement get\_children, the negatives are outweighed by the benefits of this storage method: including faster run-time and less storage.