Rush Hour Write-Up

Trishal Muthan, Partner: Alexander Peal

October 22, 2020

For the Rush Hour programming assignment, Alexander and I have decided to declare failure.

1 What have we done?

At the moment, our code does a few things.

1.0.1 Modeling

The first thing we decided to do was to decide how to model the board itself. We settled on a matrix of characters, where blank spaces were held by a "0", cars/trucks were held by letters from "A"-"F" (or however many letters are necessary), and the target car was held by a "*" This model seemed to work well because we could easily access different positions in the matrix and also because it is easier to visualize what the actual board might look like.

1.0.2 getChildren, getHorizCars, getVertCars

After creating the model itself, we decided to make the getchildren function (which returns a list of all the children boards that originate from one given board). Once we started to attempt the getchildren function, we realized that it would be helpful to create another helper function that returns all the vertically and horizontally positioned boards. This way we would be able to move the vertical cars and horizontal cars independent of each other. So we made two functions, one called getHorizCars (which returned all the horizontally positioned cars) and one called getVertCars (which returned all the vertically positioned cars). The way these functions work is that they traverse through the rows and columns in each board. Then, for every position, the horizontal function looks if the index to the right also has the same character as the current index. If it does, starting from the current index, it loops over the entire row to calculate the length of the car/truck. Then, to a list, it appends a tuple containing the letter, the row, the column, and the length of the board. After this section of the method, the list would contain many duplicates. So I made the function remove cars/trucks from the list that are duplicates. Then I returned the list. I did the exact same thing for the getVertCars method except I would check one row down rather than the column to the right. Alexander and I were able to get these methods to work properly. Then with these functions in our toolkit we began working on the getchildren method. We weren't able to entirely get the getchildren method to work, but we were able to make it work partially. First we called the getHorizCars method and tried to get the boards where the horizontal cars are moved 1 to the right or left. This was very similar to sliding puzzle code where we basically had to make sure that we were within the correct dimensions and then swap the characters at the correct indices. We were able to get it to work for both the 1 car shift horizontally (left and right) and vertically (up and down), as both of these are essentially the same code except we move a different direction.

2 Why doesn't it work?

But where we faced struggles was when we were trying to make the vehicles move more than one space in one move. We have a general idea of checking in a certain direction and continuing moving until we find a space that doesn't hold a "0", but we weren't able to implement this in code. This is the place we left the code at before we ended class. We think that after thinking about it a little bit more and figuring out more of how to actually implement the ideas we have, we might be able to finish the getchildren method, but at the moment, we feel that declaring failure is the best option for us moving forward.