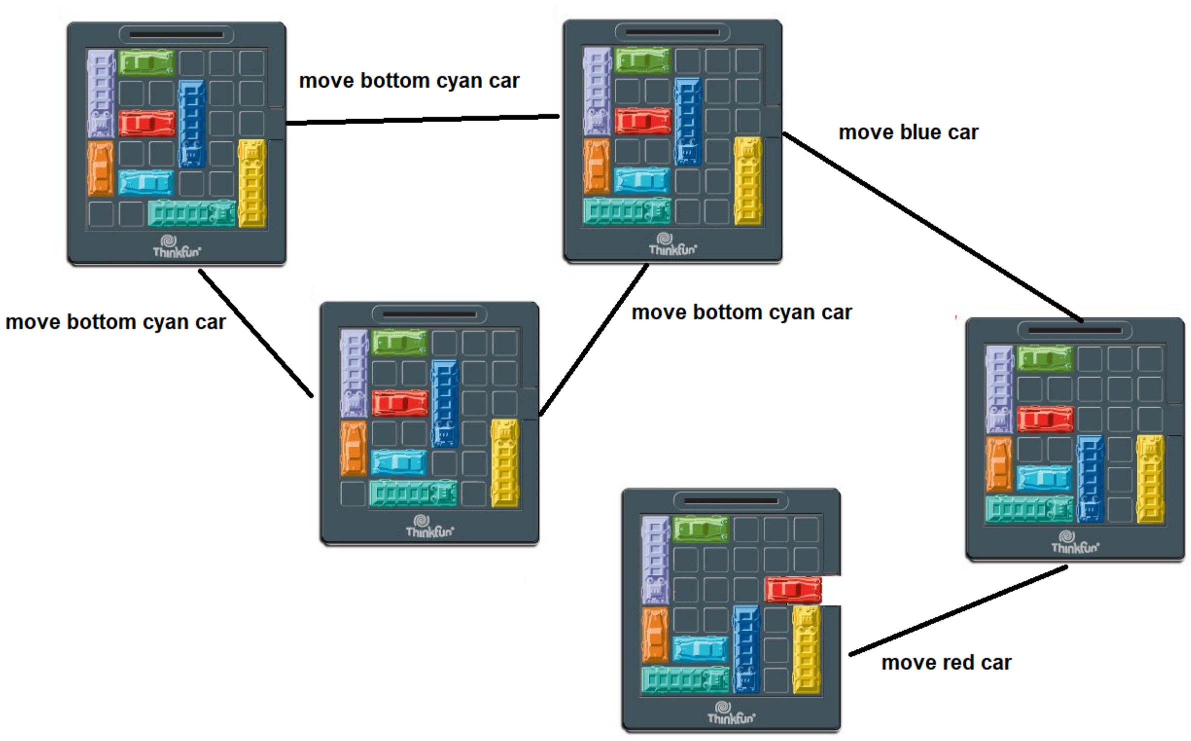
CMPT 225
Ritchie Kumar
Tony Kooliyath

Ritchie:

- Implemented BFS
- Implemented algorithm to find edges of a given node (states) using HashMap and Queue
- Implemented Node class
- Initialized data of Car class

Tony:

- Implemented Node traversal
- Implemented file reading and writing
- Testing and Debugging
- Implemented combinations of board configurations to consider



Visualization of RushHour board as a graph
(Source: https://piazza.com/class/kjhik62cw4z2as?cid=273)

We decided to go with a Breadth First Search (BFS) as our algorithm as it is one of the most efficient algorithms to find the shortest path from a source node to a destination node without the use of formal heuristics.

We used data structures such as Queue, ArrayList, HashMap, and linked lists to implement the BFS. Queues to iterate through the neighbours / adjacent nodes, ArrayList to iterate through neighbours / nodes, and HashMap to store nodes that are going / being visited while implementing linked lists to link a node to its previous node.

A* algorithm can be better in run time provided there is a good initial guess of the minimum number of moves required to reach the solution. All algorithms can solve the puzzles, but they have different properties such as run time and complexity. This is why we went with BFS as calculating the minimum number of moves for A* is more demanding with an incorrect guess.

Implementing the neighbours / adjacent nodes of the RushHour graph was the most difficult. This is because we had to consider all the possible moves of a given car one move at a time and save this information afterwards as a potential solution. Implementing the Node class and Queue for BFS search was easier because they both follow a standard formatting with the exception of a few tweaks for the purpose of RushHour.

Initially, we tried using our code from assignment1 to use the makeMove() function. However, we decided to use the professor's code for RushHour as it has the Cars hashmap, which provides useful data for where all the cars are initially on the board. To remember the visited nodes of RushHour, we initially used an ArrayList to store all the visited nodes. After testing BFS for every puzzle in the project folder with the ArrayList, the search took ~ 15 seconds in run time to solve the given sample puzzles. After switching to a HashMap, the run time was reduced to ~2 seconds, thereby improving time complexity. We also tried an object array to remember the path (moves) of the solution but once again the run time was slow. This is why we switched to the Node Class.