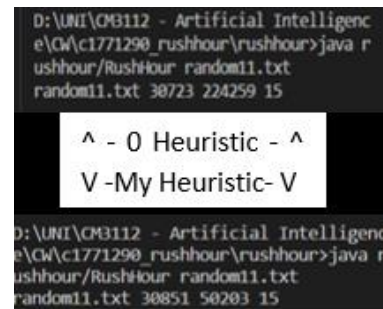


# CM3112-COURSEWORK REPORT

## 1. Heuristic Effectiveness

My chosen heuristic is effective in comparison to always returning 0 because if an A\* search's heuristic always returns 0 then it doesn't take into account the cost of each path so is effectively a breadth-first search, due to the fact all paths in this situation have a cost of one (moving any number of squares is a cost of one). With small trees that have a low number of nodes breadth-first could be as effective as A\* search in this situation. But, in the case of rush hour some of the trees with A\* search expand tens of thousands of nodes, so breadth-first won't be as effective. An example of this is shown on the right using the rush hour game random11.txt which expands 224,259 nodes when the heuristic returns 0 as for using my heuristic, only 50,203 nodes are expanded. Although the time to complete is similar and the route is the same, in other examples the optimal route may not be the first route found so breadth-first could return a result that isn't optimal as it just returns the first result found. Therefore, the heuristic is effective as when tested against different examples my heuristic always found the optimal route with less nodes expanded than when the heuristic was equal to 0. This also helps with the issue of running out of memory as if the heuristic is too small, too many nodes maybe expanded and then a memory-overload error is returned.



```
D:\UWI\CM3112 - Artificial Intelligenc
e\GW\C1771290_rushhour\rushhour>java r
ushhour/RushHour random11.txt
random11.txt 30723 224259 15

^ - 0 Heuristic - ^
V - My Heuristic - V

D:\UWI\CM3112 - Artificial Intelligenc
e\GW\C1771290_rushhour\rushhour>java r
ushhour/RushHour random11.txt
random11.txt 30851 50203 15
```

## 2. Admissibility of chosen Heuristic

A heuristic is considered admissible if it never overestimates the true cost to reach the goal state. My heuristic is based on how many cars are blocking the row spaces to the right of the goal car and whether those cars are able to move entirely out of the path of the goal car. This way it can be considered admissible as it is impossible to reach a goal state if there are still cars both occupying a space in the same row and to the right of the goal car. Then if said cars cannot move either up or down to clear the goal path then the cost is increased again. This is due to the fact every puzzle has a solution, therefore the only other reason the car can't move up or down is that another car is blocking the way. This means that another car must also move to free the goal car and therefore the heuristic must be one higher. This means it's admissible as it's impossible to reach the goal without completing all of these moves.

## 3. Heuristic Improvement

One way to further improve this heuristic would be to calculate how many cars need to be moved for each vertical car, that occupies a space along the same row as the goal car and is to the right of the goal car, to move out of the path of the goal car. This heuristic would be a lot tougher to implement but for each vertical car you can calculate whether the car blocking it (if there is one) can be moved then calculate if there's a car blocking that car and so on, until you calculate how many cars need to be moved to free each vertical car blocking the goal car. A way to implement this would be to have a while loop that stays active, increasing the cost by one each time it's incremented, until a car that can be moved is found. The spaces occupied could be kept in an array list that can then check if each location the car would like to move too is taken. This way, say a vertical car can't move until 3 other cars move then the heuristic would be 2 higher than in my current heuristic so even less nodes will be expanded and it would still be admissible.