# Rush Hour

A program for solving the Rush Hour game using breadth first search.

## Usage

The `rushhour` module provides a general purpose `RushHour` class that can be used by any searching interface for determining new paths. This allows for the creation of new algorithms such as Best-First search or A*.

The `rushhour.py` file also acts as a script, accepting a problem file path. Sample problems are provided in the *problems* directory and can be run with:

```
python rushhour.py problems/p1
```

The script will perform a breadth first search of the sample problem (up to a maximum depth of 100) and will report the number of solutions found, the solution steps, and the total number of nodes visited in the search.

## Method

The breadth first search operates by enumerating the vehicles in a board and determining which have spaces next to them that can be moved into. For each vehicle that can be moved, the board generates a new edge state.

## Determining Difficulty

In addition to the `rushhour.py` script, there is a `difficulty.py` script. The interface to this script is the same as the former, it still takes a path to a problem file, but this script responds with **Easy**, **Moderate**, or **Hard**, depending on the difficulty of the game. Additionally, if the game cannot be solved, the script responds with **Impossible**.

The difficulty of the game is determined by analyzing both the shortest solution to the game, and the number of possible solutions. A game can be easy either by having a very short solution path, or a large number of possible solutions. Conversely, a problem can be hard by having a very long shortest solution path, or having a small number of possible solutions.

It is possible to generate a hard puzzle that does not have many steps, so long as there are not many possible solutions, meaning that the player would have little room for error.

It is possible to generate an easy puzzle that has many steps, so long as there are many possible solutions to the problem.