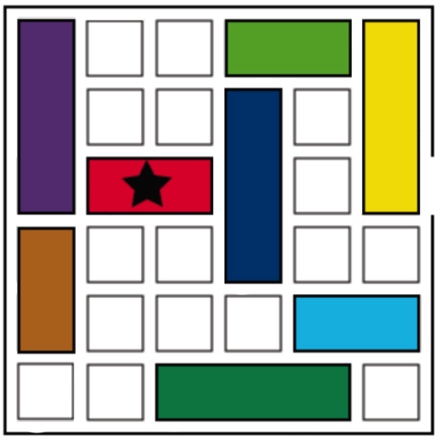
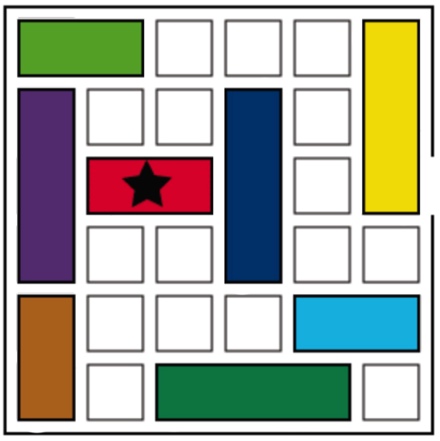
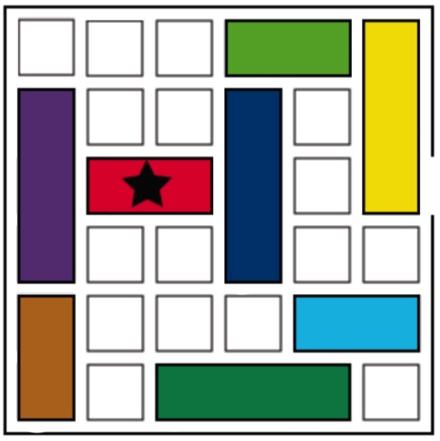
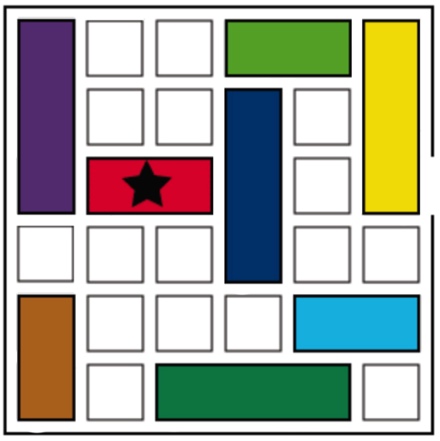
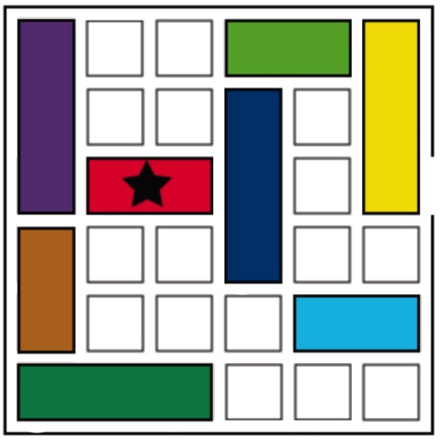
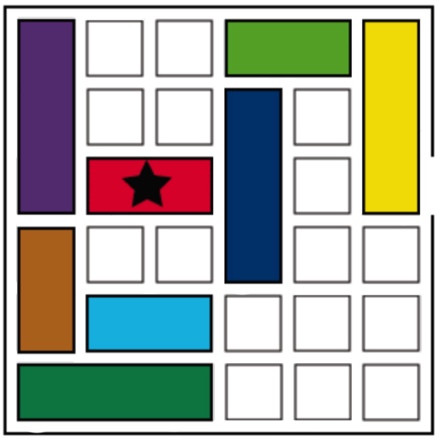
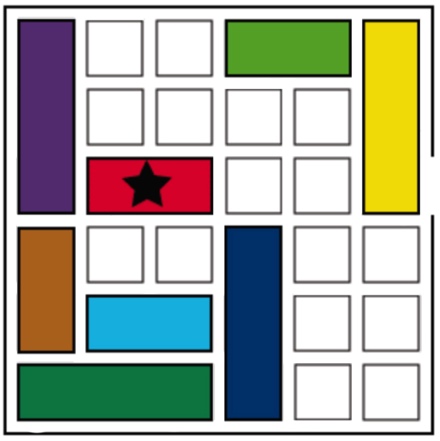
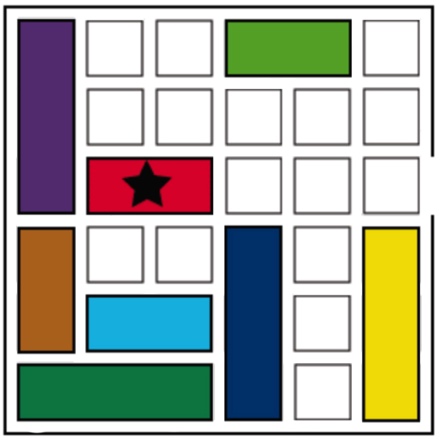
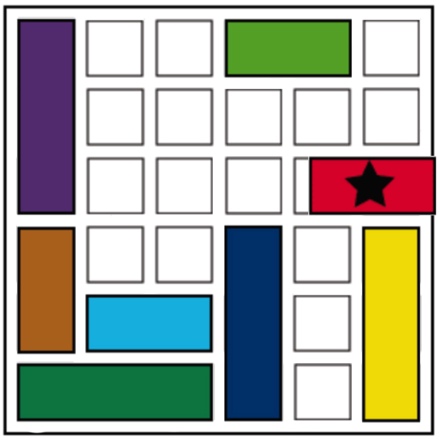
Hunter Leise

CSCI 3202

Problem Set 1

* 1.  0 1 2

 3 4 5

 6 7 8

* 1. To estimate the problem space for this puzzle, I multiplied the number of possible locations for each piece, while subtracting a tally of locations that conflict with one another:
* Light Green Car: 5 possible locations, 4 possible conflict states
* Purple Car: 2 possible locations, 2 possible conflict states
* Brown Car: 2 possible locations, 2 possible conflict states
* Red Car: 5 possible locations (goal state included), 4 possible conflict states
* Light Blue Car: 4 possible locations, 3 possible conflict states
* Dark Green Car: 4 possible locations, 5 possible conflict states
* Dark Blue Car: 4 possible locations, 7 possible conflict states
* Yellow Car: 4 possible locations, 7 possible conflicts states

Begin by adding up the possible conflict states and dividing them by two since all of the conflict states are being counted twice:

4 + 2 + 2 + 4 + 3 + 5 + 7 + 7 = 34 / 2 = 17 Conflict states

Then multiple all of the possible locations of each car and subtract the conflict states:

5 \* 2 \* 2 \* 5 \* 4 \* 4 \* 4 \* 4 = 25,600 – 17 = 25,583 states in the problem space

* 1. A plausible “h” function for A\* search would be the number of cars blocking the red car from the goal state. This would be easily computable and will always underestimate the number of moves remaining to solve the puzzle. For example, in the starting state of the rush hour puzzle from question 1.1, the “h” function would return 2 (since the dark blue and yellow cars are blocking the exit).
  2. My solution in 1.1 definitely has features in common with depth-first search because I committed to my early moves and continued until I found that my earlier moves didn’t result in a helpful state. Only then would I go back and try a different move. I don’t think my solution had many features in common with breadth-first search because I didn’t start off by trying every single possible first move, and then trying every possible second move for those, etc. My solution did have some features in common with means-ends analysis because I did analyze the difference between my current state and the goal state, and I did use that to help dictate what my next move would be.

Bidirectional search would be a great idea for this puzzle

2)

3)

4.1) I edited my program from question 5.2 by eliminating the goal state test such that it would run through every single possible state in the problem space. Once this is finished, I could just test the length of the visited states array. By doing so, I found that there were 63 possible states in the problem space.

4.2)