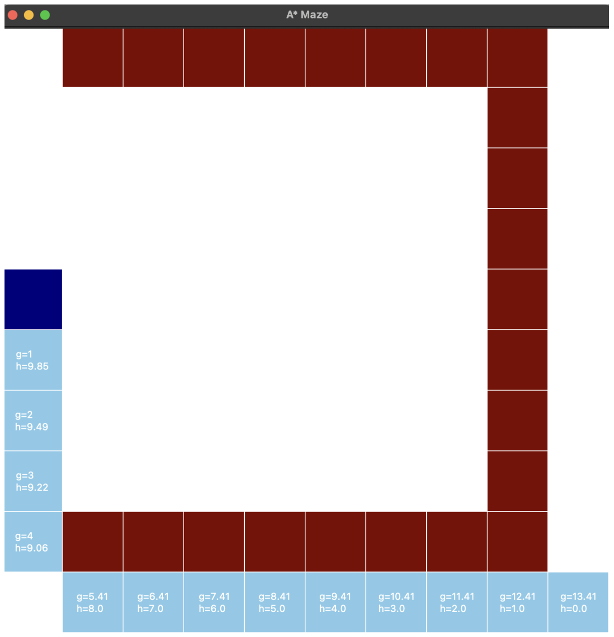
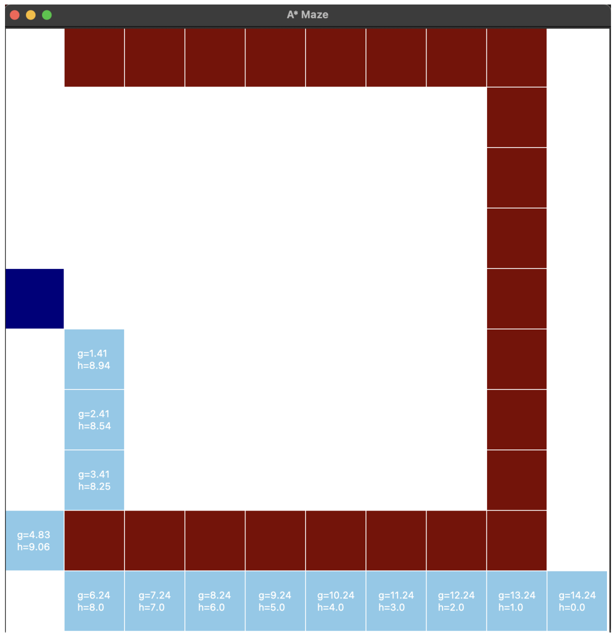
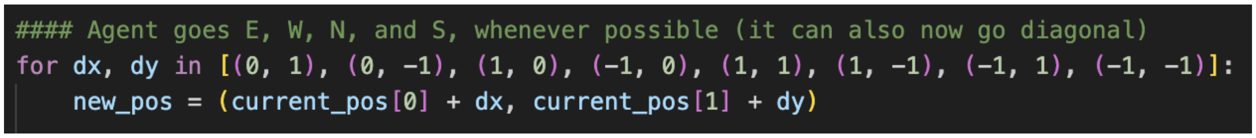
#HW3 problem 2 Simon Tice





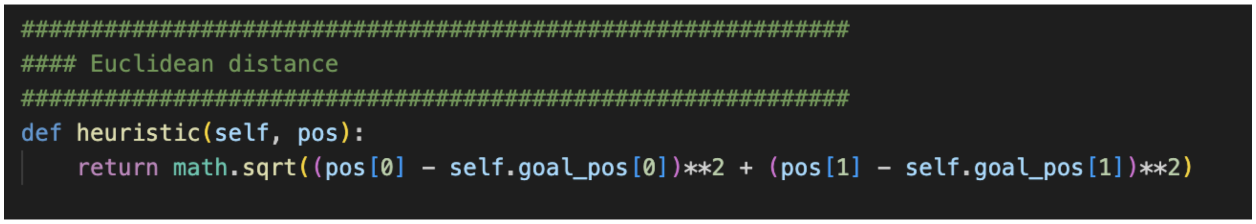
With diagonal movement enabled and Euclidean distance as the heuristic, the A* algorithm is on the left and the greedy best-first algorithm is on the right. As you can see, the A* algorithm only uses the diagonal to cut the corner and make the path more optimal, but the greedy algorithm takes unnecessary diagonals and moves towards the right at first to go towards the goal initially. The algorithms are still behaving pretty much as you would expect, despite having the ability to move diagonally now.

To add the diagonal movement, I just had to add (1,1) (-1,1) (1,-1) (-1,-1) to the list of possible directions, allowing the path to move on a diagonal plane.



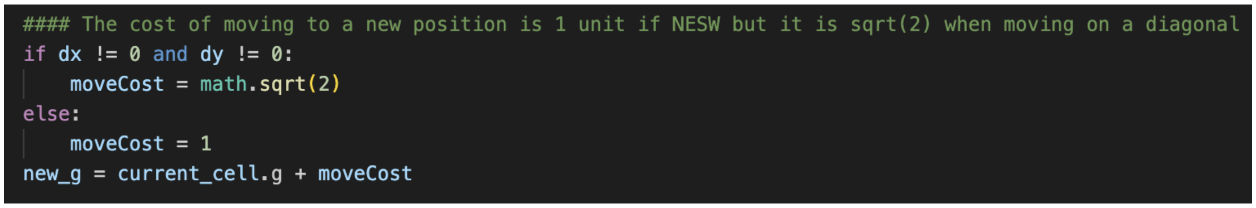
```
#### Agent goes E, W, N, and S, whenever possible (it can also now go diagonal)
for dx, dy in [(0, 1), (0, -1), (1, 0), (-1, 0), (1, 1), (1, -1), (-1, 1), (-1, -1)]:
    new_pos = (current_pos[0] + dx, current_pos[1] + dy)
```

To make the heuristic go from Manhattan distance to Euclidean distance, I changed the heuristic function to the Euclidean distance algorithm so that it will use the true distance as the heuristic.



```
############################################################
#### Euclidean distance
############################################################
def heuristic(self, pos):
    return math.sqrt((pos[0] - self.goal_pos[0])**2 + (pos[1] - self.goal_pos[1])**2)
```

I also added this if statement, where the actual cost (g) is updated so that it doesn't count a diagonal move the same distance as a horizontal or vertical move. If it's a diagonal move, the cost is sqrt(2) because of the Pythagorean theorem: (Sqrt(1^2 + 1^2) = Sqrt(2)



```python
#### The cost of moving to a new position is 1 unit if NESW but it is sqrt(2) when moving on a diagonal
if dx != 0 and dy != 0:
    moveCost = math.sqrt(2)
else:
    moveCost = 1
new_g = current_cell.g + moveCost
```