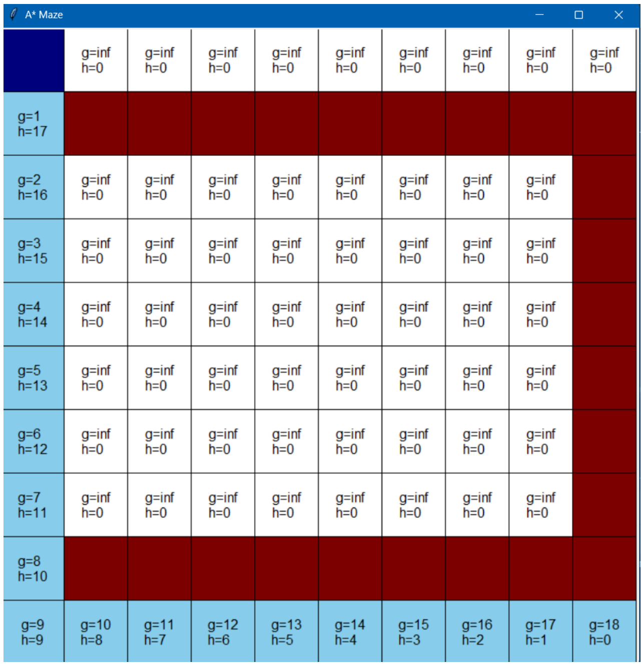
Problem 1:

Output with original A Mapping:*



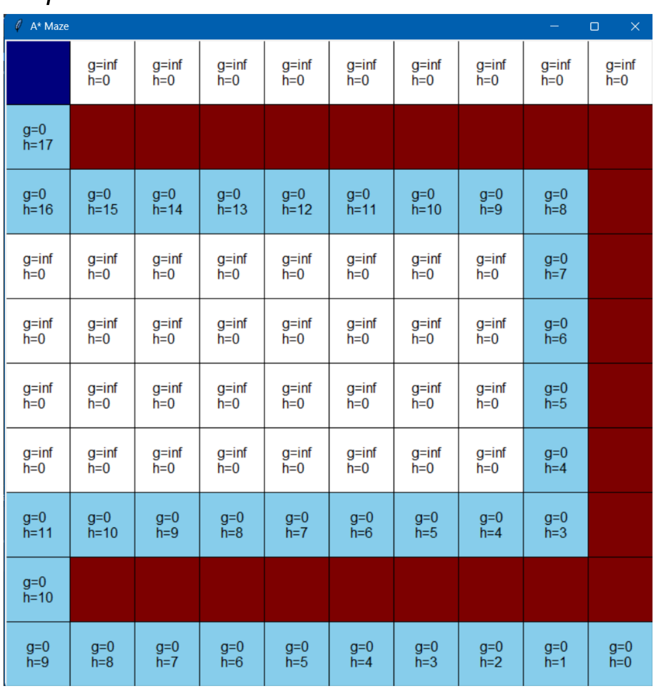
Modification to make it a Greedy Best-First Algorithm:

```
new_g = current_cell.g
```

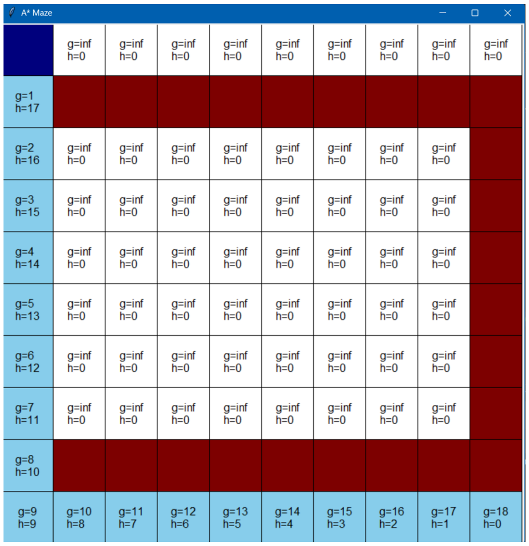
Becomes:

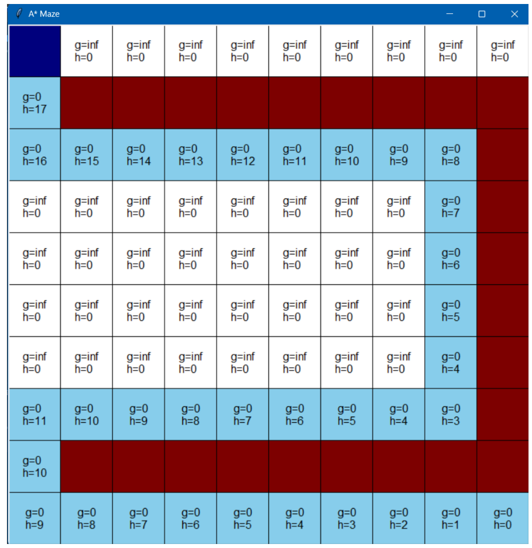
```
new_g = 0
#Makes g() in the equation equal 0
```

Output:



Comparison: Original vs. Greedy:





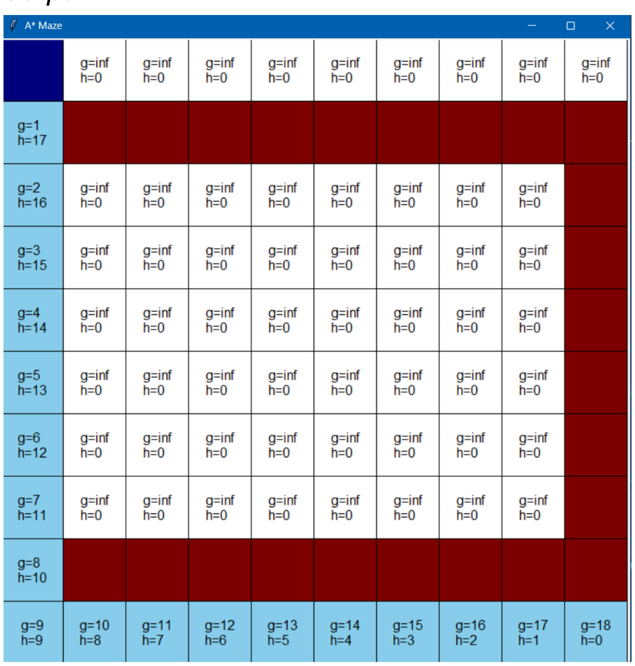
Original Greedy

Conclusion:
The greedy path is always choosing the next move that is closest to the goal, without considering the total cost of the moves it makes.

Problem 2:
Original pathing makes the agent go only N, S, E, and W:

```
for dx, dy in [(0, 1), (0, -1), (1, 0), (-1, 0)]:
```

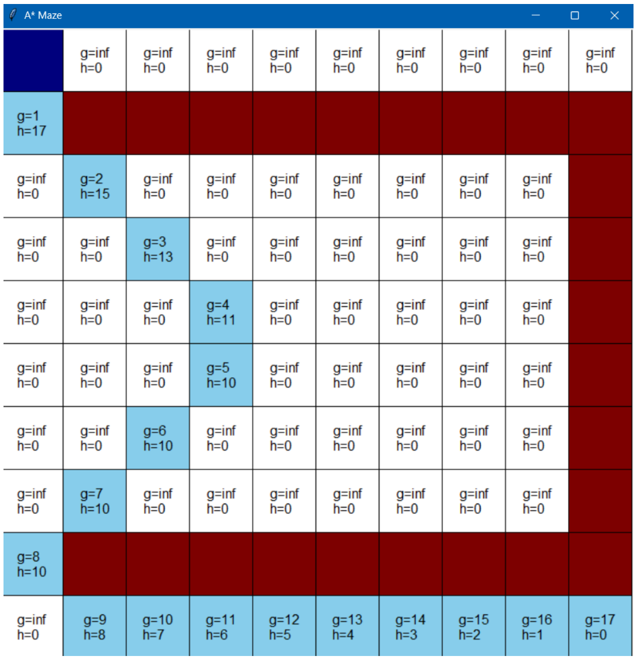
Output:



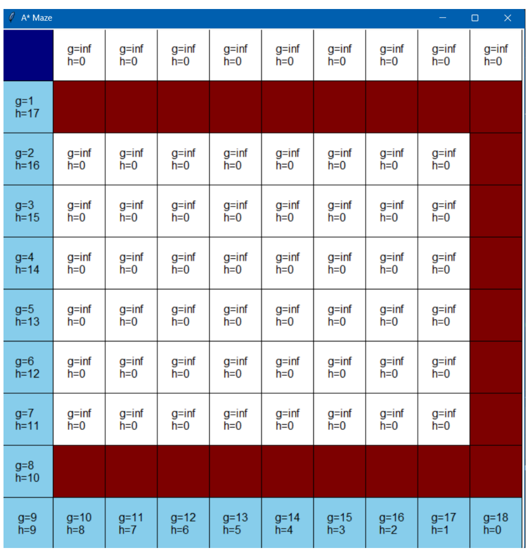
New pathing adding NE, NW, SE, and SW:

```
for dx, dy in [(0, 1), (0, -1), (1, 0), (-1, 0), (1,1), (-1, -1), (1, -1), (-1,1)]:
```

Output:



Comparison Between Original vs. Euclidean Pathing:



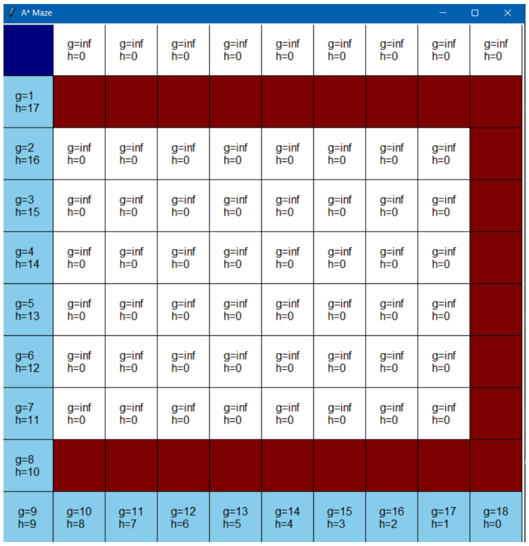
Original Euclidean

Conclusion:

The Euclidean pathing uses only 17 moves to reach the goal, while the original uses 18. This is because Euclidean pathing can skip the corner moves the original has to take, as it can move diagonally. This Eclidean pathing uses the greedy algorithm from Problem 1.

Problem 3:

Original grid with no weights:



New code:

```
self.cells[new_pos[0]][new_pos[1]].g = a * new_g
self.cells[new_pos[0]][new_pos[1]].h = b * self.heuristic(new_pos)
self.cells[new_pos[0]][new_pos[1]].f = new_g +
self.cells[new_pos[0]][new_pos[1]].h
```

Outputs:

a = 3.0, b = 3.0

a = 2.0, b = 2.0

a = 5.0, b = 2.0

a = 2.0, b = 5.0

Conclusion:

The paths are all the same as the original weightless grid. However, all of the weighted paths are the same, just with different numbers attached for g() and h(). This could be because the heuristic is similar in the outputs that are weighted. Possible closer to each other than to the original unweighted heuristic.