Problem One:

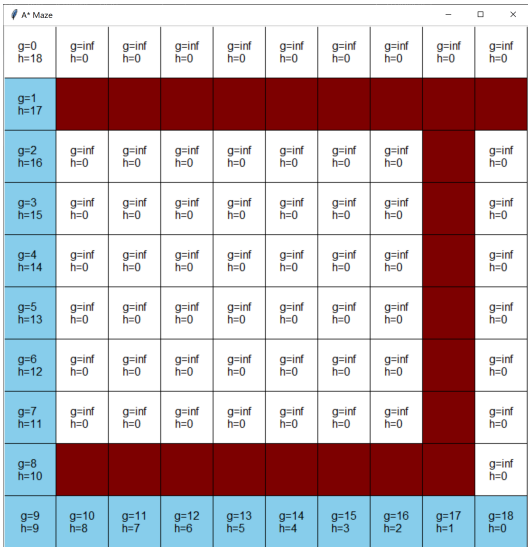
- Modify AStarMaze to compare the behaviors of the Greedy Best-First and A* search algorithms. You need to modify the maze configuration so you can visually observe differences in the optimum paths generated by the two algorithms. Your report should include a side-by-side comparison of the two approaches similar to the graph shown below along with your explanation. You only need to draw the shortest paths and not the highlighted frontiers.

Maze Used in example:

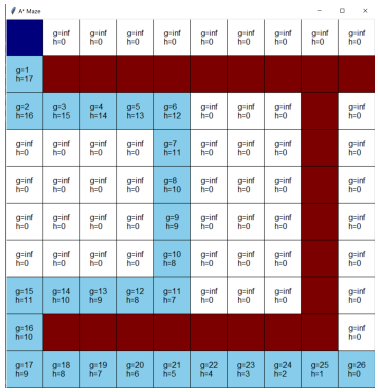
```
maze = [
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
   [0, 1, 1, 1, 1, 1, 1, 1, 1, 0],
   [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
   [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
   [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
   [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
   [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
   [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
   [0, 1, 1, 1, 1, 1, 1, 1, 1, 0],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
]
```

Start position = (4,3)

Goal Position = Bottom right

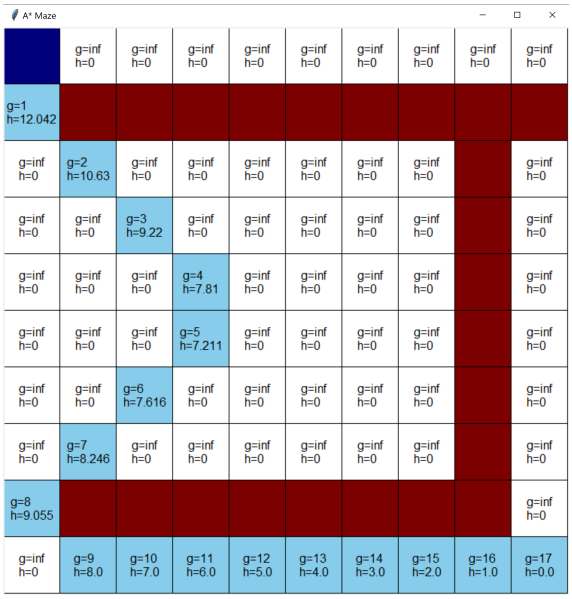
- Solution: To make the A* greedy I just simply modified the algorithm so that it only uses the heuristic to make the movement decision
- A*
    - 

    - Greedy A*

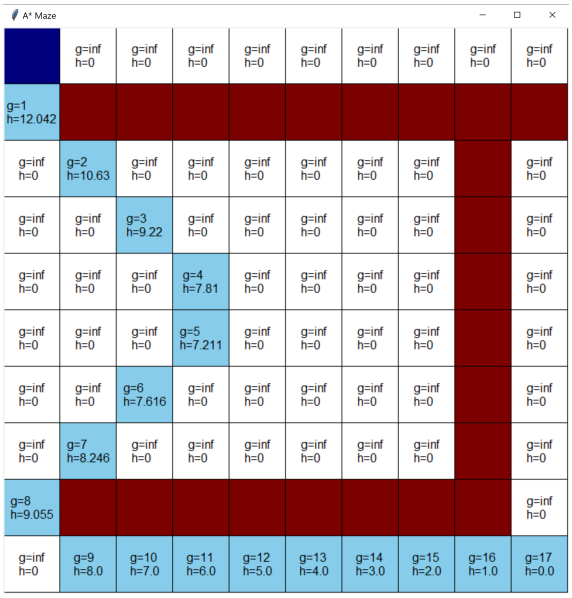


Problem 2:

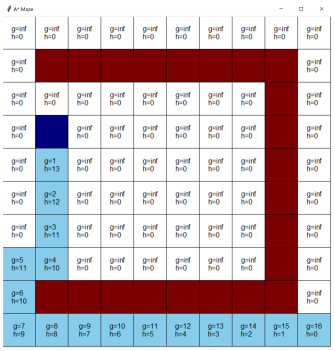
- Use the Euclidean Distance heuristic.
- The agent is allowed to make diagonal moves (i.e., NE, NW, SE, SW) in addition to the usual N, S, E, and W moves.
    - To do this I added the following directions to the move list (1, 1), (-1, 1), (1, -1), (-1, -1)
- The moves are made randomly and not in any specific order.
    - To randomize the move I used the numpy.shuffle to perform this before each direction is considered

- Greedy:
    - 

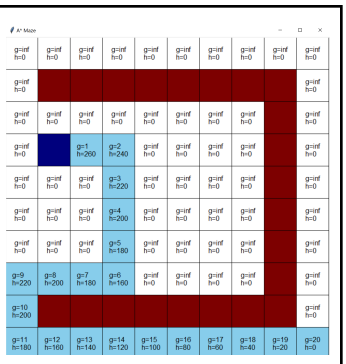
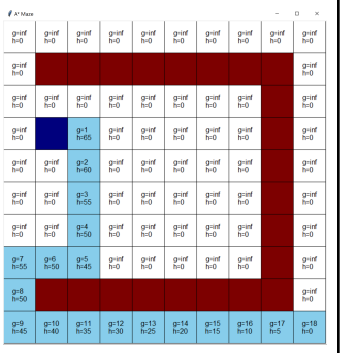
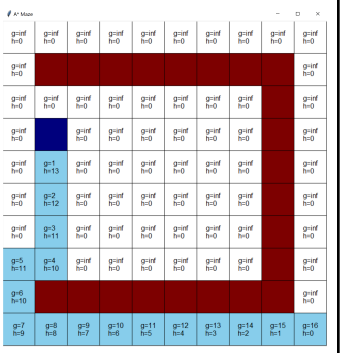
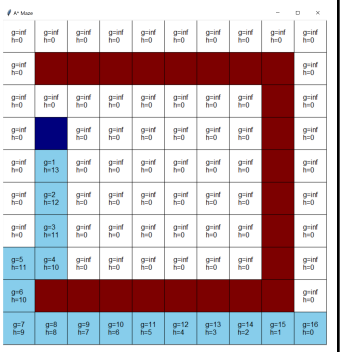
- A* Euclidean Distance heuristic
    - I tried many different possible configurations and I could not seem to find one that produced a change in the way the maze was solved

- 

Problem 3:

- The evaluation function in AstarMaze is defined as f(n) = g(n) + h(n). A weighted version of the function can be defined as: f(n) = A* *g(n)* + B* h(n)
  - I added a scaler to the calculate function as described in the equation above.
- B can be considered the algorithm's bias towards states that are closer to goal. Run the algorithm for various values of the bias to determine what changes, if any, are observed in the optimum path. Include screenshots of the path for each specific value of B along with your explanation.
-
  - Using the table below I found that changing the B value managed to increase the bias the program had to moving to the right first to find the goal state. In the first example where B is 5 it moved 1 space right before moving down and in the second example, it started to move 2 spaces to the right before moving down.
-
- See how changing A and B effect the model
  - No Changes made to the program (Control)
  - 

A	B	Behavior

1	20	
1	5	
5	1	 No change noticed
20	1	 No change noticed