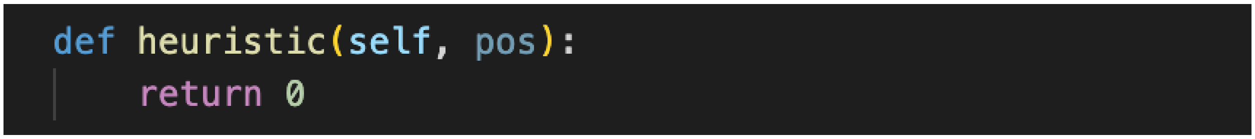
Sean Rock

Assignment 4 A* Algorithm

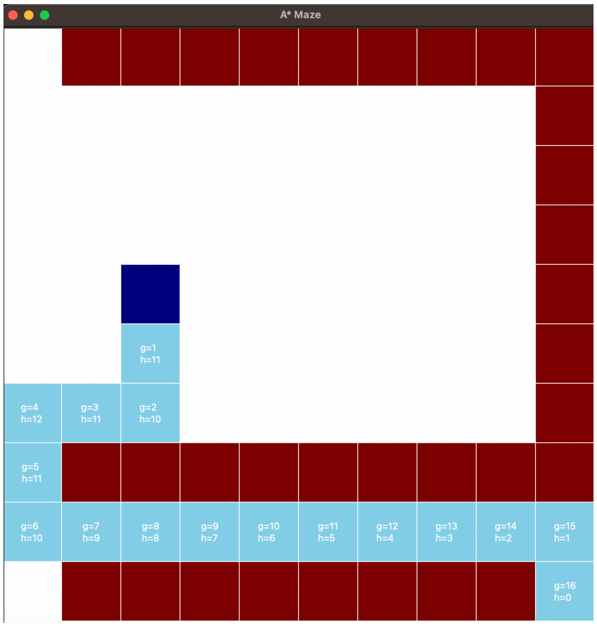
Purpose: Investigate the result of the A* maze algorithm. The first two problems compare its result against the Greedy Best First Search (GBFS) algorithm and the third problem compares the effects of coefficients for the heuristic value (h(n)) and the moving cost (g(n) on the algorithm's path.

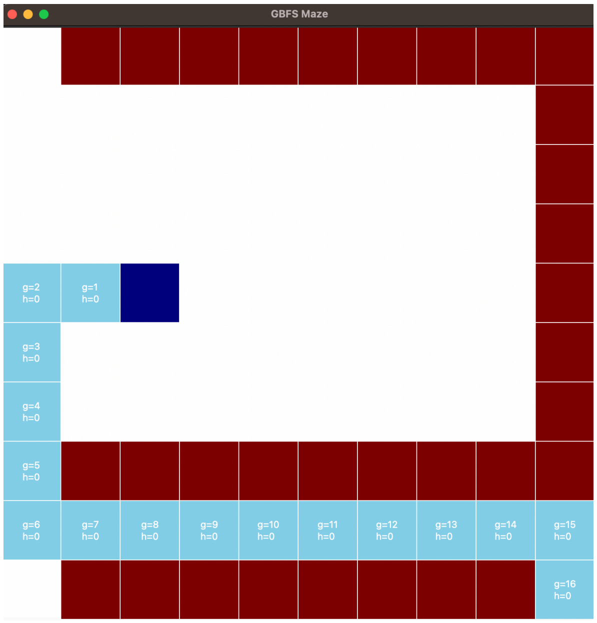
1. Modify **AStarMaze** to compare the behaviors of the **Greedy Best-First** and **A*** search algorithms. You need to modify the maze configuration so you can visually observe differences in the optimum paths generated by the two algorithms. Your report should include a side-by-side comparison of the 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.

Change to AStar algorithm to make it GBF



```
def heuristic(self, pos):
    return 0
```

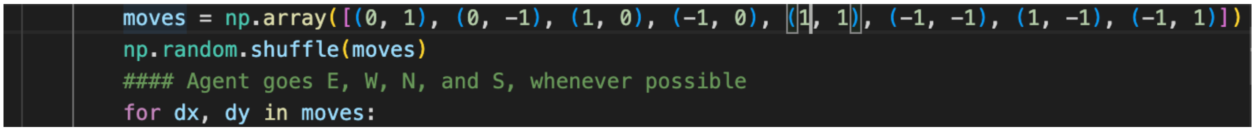




As shown on the right, the A* algorithm considers the heuristic value 'h' which is the distance of the location to goal. Thus, it goes straight down before moving around the barriers since those values are closer to the goal state. The GBFS does not consider the distance of the node from the goal state, so it goes left first.

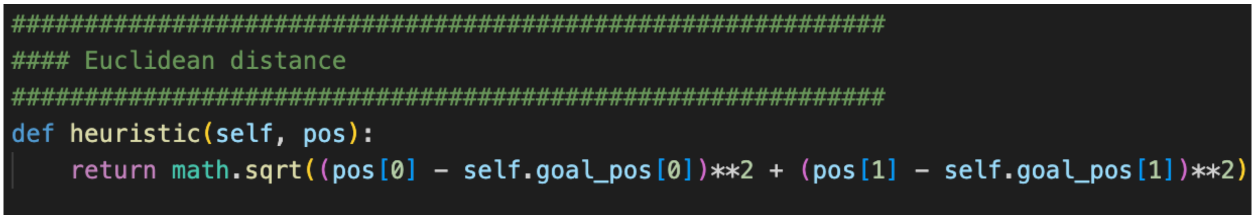
2. Repeat the above experiment but this time:

- 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.
- The moves are made randomly and not in any specific order.



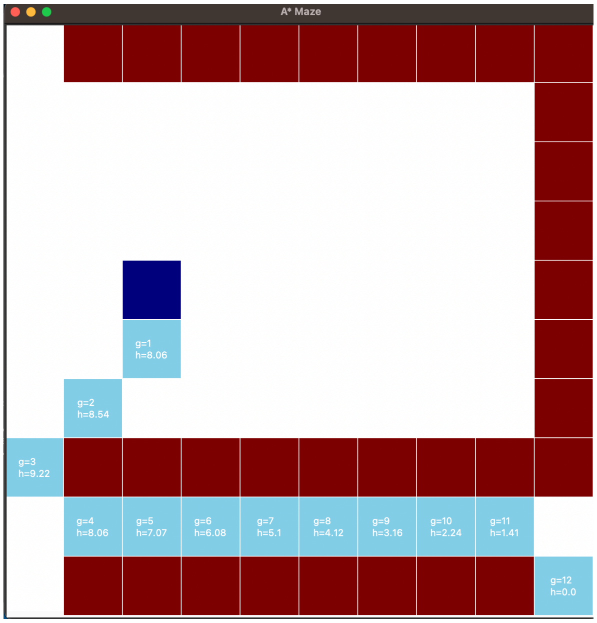
```
    moves = np.array([(0, 1), (0, -1), (1, 0), (-1, 0), (1, 1), (-1, -1), (1, -1), (-1, 1)])
    np.random.shuffle(moves)
    #### Agent goes E, W, N, and S, whenever possible
    for dx, dy in moves:
```

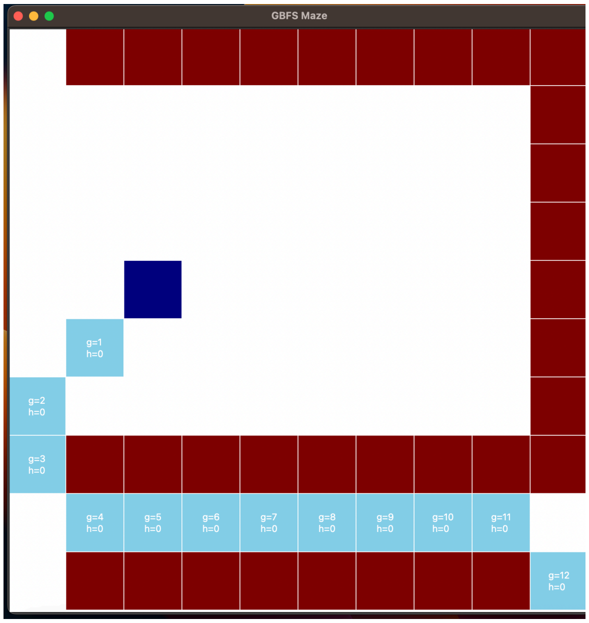
Change to A* and GBFS algorithm that allows diagonal movement



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

Change to A* to make the heuristic calculated with euclidean distance.(Uses numpy array to randomize directions)





The A* algorithm goes down first to a lower heuristic value before going diagonally. The GBFS algorithm goes diagonally immediately as it only considers g, the step cost.

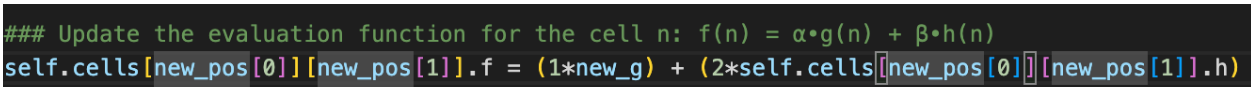
3. .

    a.

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) = α•g(n) + β•h(n) where α,β≥0

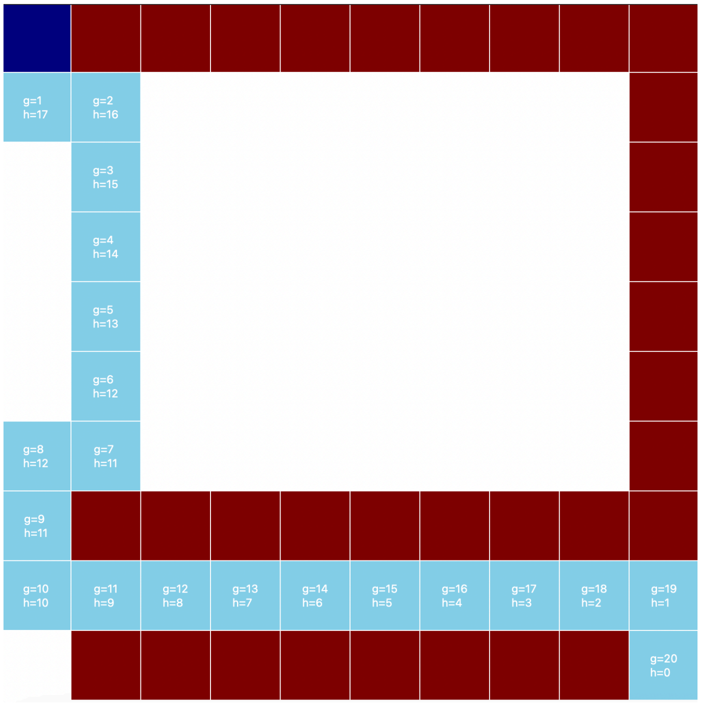
Explain how different values of α and β affect the A* algorithm's behavior. Tabulate your results:

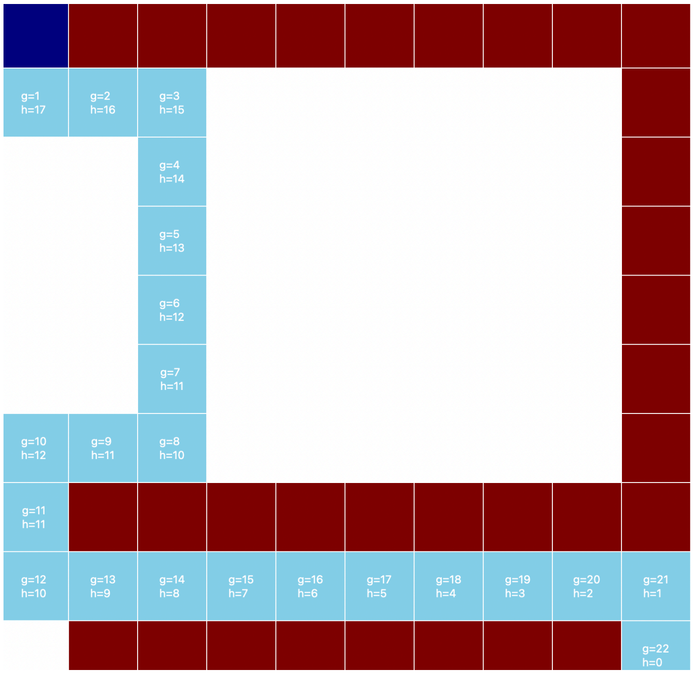


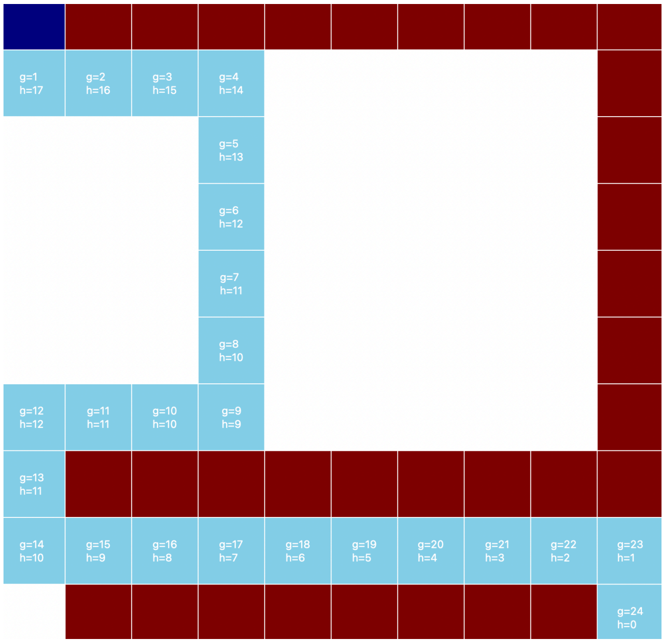
```
### Update the evaluation function for the cell n: f(n) = α•g(n) + β•h(n)
self.cells[new_pos[0]][new_pos[1]].f = (1*new_g) + (2*self.cells[new_pos[0]][new_pos[1]].h)
```

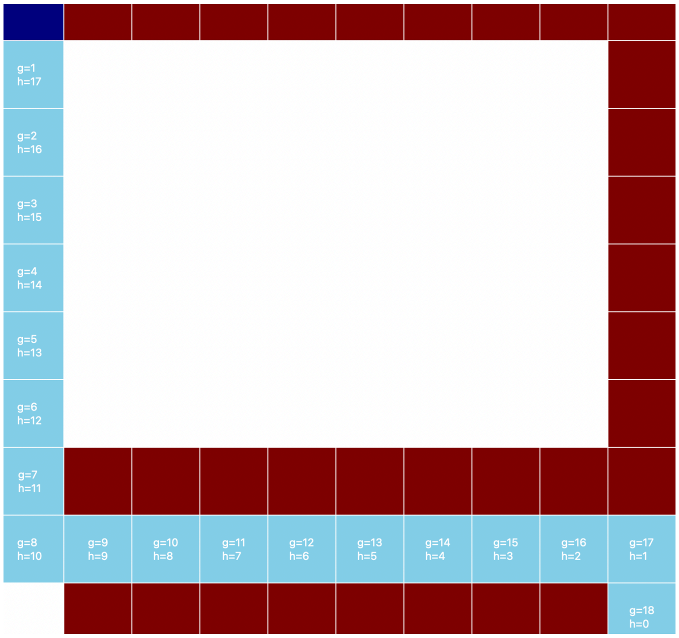
Change to A* code to factor coefficients into the algorithm

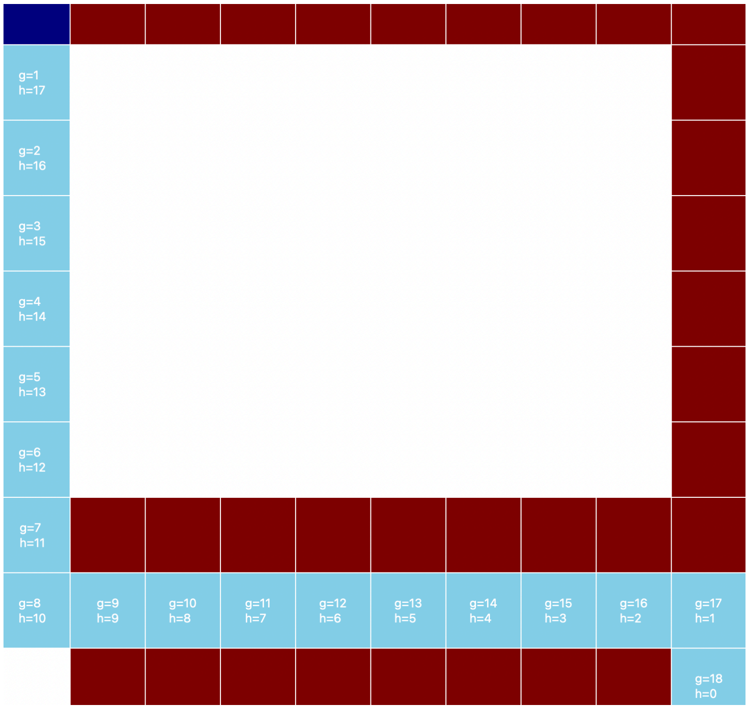
α	β	Result

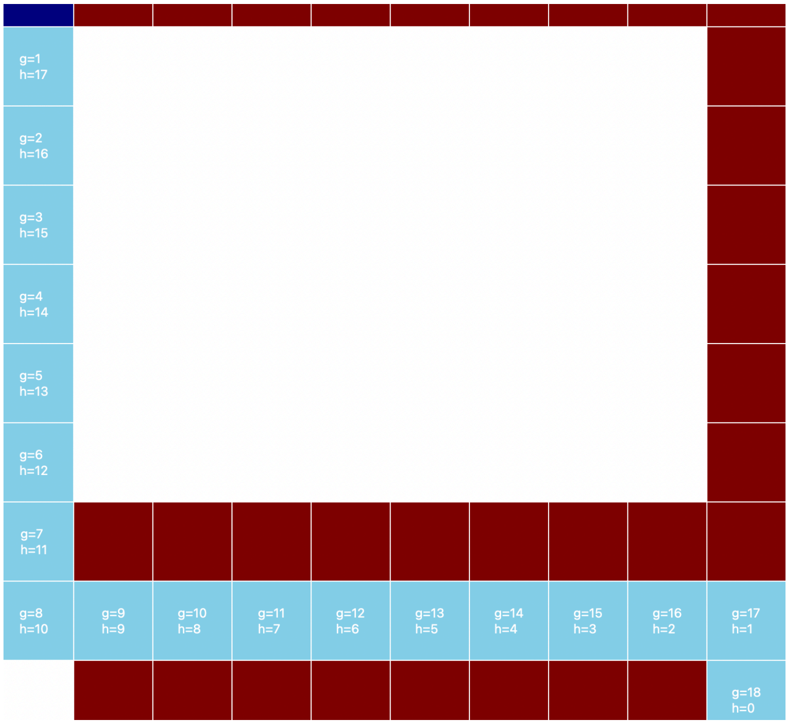
1	2	 g=1 h=17
		g=2 h=16
		g=3 h=15
		g=4 h=14
		g=5 h=13
		g=6 h=12
		g=8 h=12
		g=7 h=11
		g=9 h=11
		g=10 h=10
		g=11 h=9
		g=12 h=8
		g=13 h=7
		g=14 h=6
		g=15 h=5
		g=16 h=4
		g=17 h=3
		g=18 h=2
		g=19 h=1
		g=20 h=0

1	3	g=1 h=17
		g=2 h=16
		g=3 h=15
		g=4 h=14
		g=5 h=13
		g=6 h=12
		g=7 h=11
		g=10 h=12
		g=9 h=11
		g=8 h=10
		g=11 h=11
		g=12 h=10
		g=13 h=9
		g=14 h=8
		g=15 h=7
		g=16 h=6
		g=17 h=5
		g=18 h=4
		g=19 h=3
		g=20 h=2
		g=21 h=1
		g=22 h=0

1	5	g=1 h=17
		g=2 h=16
		g=3 h=15
		g=4 h=14
		g=5 h=13
		g=6 h=12
		g=7 h=11
		g=8 h=10
		g=12 h=12
		g=11 h=11
		g=10 h=10
		g=9 h=9
		g=13 h=11
		g=14 h=10
		g=15 h=9
		g=16 h=8
		g=17 h=7
		g=18 h=6
		g=19 h=5
		g=20 h=4
		g=21 h=3
		g=22 h=2
		g=23 h=1
		g=24 h=0

2	1	g=1 h=17
		g=2 h=16
		g=3 h=15
		g=4 h=14
		g=5 h=13
		g=6 h=12
		g=7 h=11
		g=8 h=10
		g=9 h=9
		g=10 h=8
		g=11 h=7
		g=12 h=6
		g=13 h=5
		g=14 h=4
		g=15 h=3
		g=16 h=2
		g=17 h=1
		g=18 h=0

3	1	g=1 h=17
		g=2 h=16
		g=3 h=15
		g=4 h=14
		g=5 h=13
		g=6 h=12
		g=7 h=11
		g=8 h=10
		g=9 h=9
		g=10 h=8
		g=11 h=7
		g=12 h=6
		g=13 h=5
		g=14 h=4
		g=15 h=3
		g=16 h=2
		g=17 h=1
		g=18 h=0

5	1	 g=1
		h=17
		g=2
		h=16
		g=3
		h=15
		g=4
		h=14
		g=5
		h=13
		g=6
		h=12
		g=7
		h=11
		g=8
		h=10
		g=9
		h=9
		g=10
		h=8
		g=11
		h=7
		g=12
		h=6
		g=13
		h=5
		g=14
		h=4
		g=15
		h=3
		g=16
		h=2
		g=17
		h=1
		g=18
		h=0

b. While increased α values had no effect on the algorithm's path as this affected each space equally, the β value had an impact on the performance. As seen in the screenshots above, the higher the value of β was the more the algorithm prioritized tiles with lower Heuristic H(n) cost. As a result, the algorithm moves more towards the middle of the maze since those tiles are closer to the goal state. This caused the algorithm to perform worse than it normally would because it would have to move around the barrier that prevented the goal from being reached from the center of the maze. The final g cost when β = 5 was 24 compared to 18 normally. This is because our heuristic calculation did not take into account any obstacles leading to it being an inaccurate guide.

Conclusion: The A* algorithms can be implemented in a variety of ways to match the situation. It differs from GBFS algorithm because it considers the heuristic cost (in our case the distance from the maze exit). Euclidean or Manhattan distance can be used to calculate the heuristic cost depending on the mobility of the agent. Also, increasing the weight of the heuristic h(n) value at least in our case can lead to worse performance.