Project_2

November 18, 2024

1 Libraries

[1]:
```
import numpy as np
import random
import heapq
import math
import matplotlib.pyplot as plt
from collections import defaultdict
```

2 Ship Design

2.1 1. Setting up the Grid Size and Sensitivity Parameter [Alpha]

[2]:
```
GRID_SIZE = 30
alpha = 0.1  # Sensitivity parameter for the rat detector
random.seed(42)
```

2.2 2. Intializing the Ship with an Open and Blocked Cells

[3]:
```
def initialize_ship():
    grid = np.ones((GRID_SIZE, GRID_SIZE), dtype=int)
    for i in range(1, GRID_SIZE - 1):
        for j in range(1, GRID_SIZE - 1):
            grid[i, j] = 0 if random.random() < 0.7 else 1  # 70% chance of␣
 ↪being open cells in the layout
    return grid

# Initializing the ship layout
ship_grid = initialize_ship()
```

[4]:
```
ship_grid
```

[4]:
```
array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1],
       [1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
        0, 0, 0, 1, 0, 0, 0, 1],
```

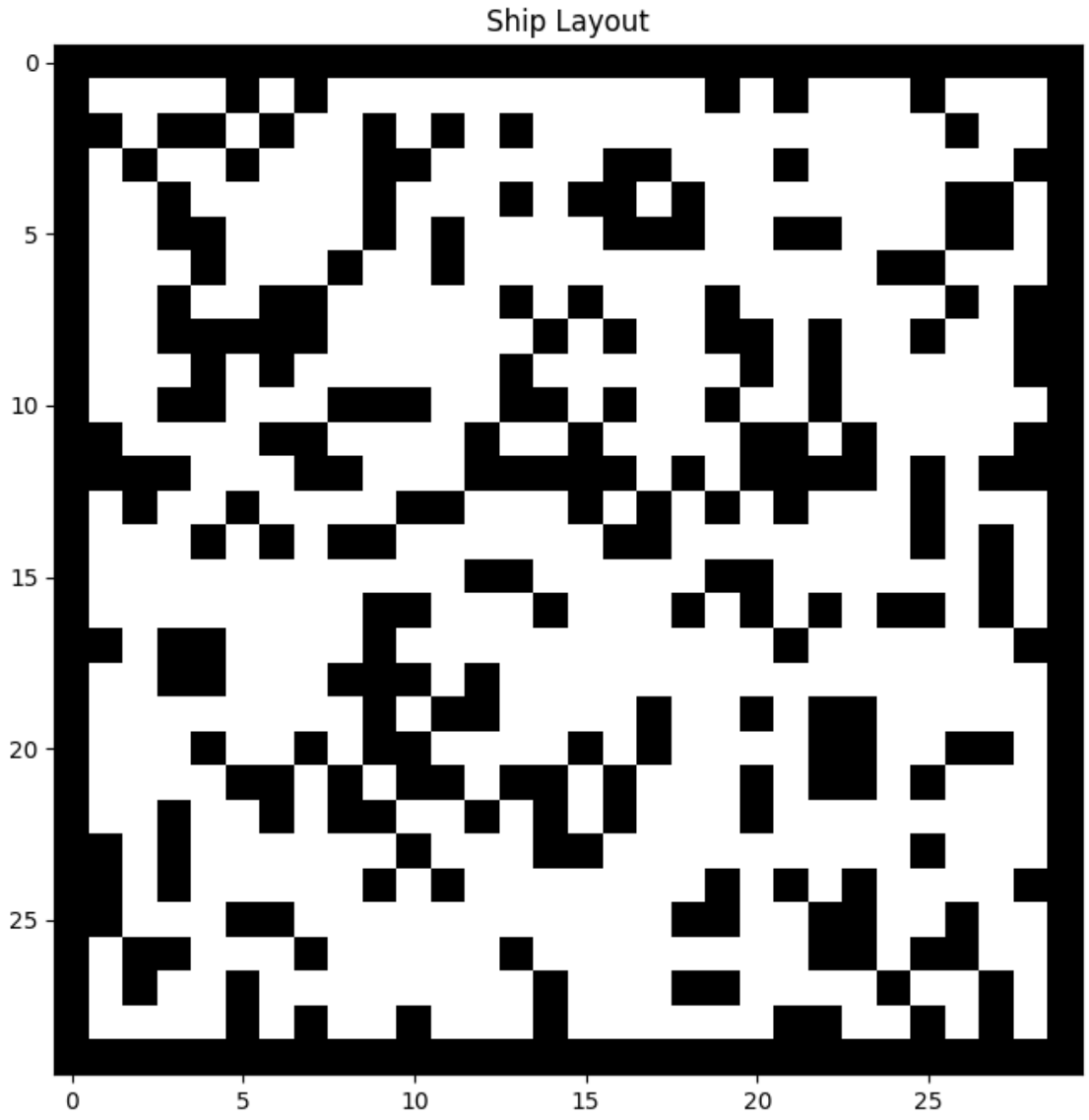
```
[1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 1, 0, 0, 1],
[1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
 0, 0, 0, 0, 0, 0, 1, 1],
[1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
 0, 0, 0, 0, 1, 1, 0, 1],
[1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
 1, 0, 0, 0, 1, 1, 0, 1],
[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 1, 1, 0, 0, 0, 1],
[1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
 0, 0, 0, 0, 1, 0, 1, 1],
[1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0,
 1, 0, 0, 1, 0, 0, 1, 1],
[1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
 1, 0, 0, 0, 0, 0, 1, 1],
[1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0,
 1, 0, 0, 0, 0, 0, 0, 1],
[1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1,
 0, 1, 0, 0, 0, 0, 1, 1],
[1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
 1, 1, 0, 1, 0, 1, 1, 1],
[1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1,
 0, 0, 0, 1, 0, 0, 0, 1],
[1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
 0, 0, 0, 1, 0, 1, 0, 1],
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0,
 0, 0, 0, 0, 0, 1, 0, 1],
[1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
 1, 0, 1, 1, 0, 1, 0, 1],
[1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
 0, 0, 0, 0, 0, 0, 1, 1],
[1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 1],
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
 1, 1, 0, 0, 0, 0, 0, 1],
[1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
 1, 1, 0, 0, 1, 1, 0, 1],
[1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0,
 1, 1, 0, 1, 0, 0, 0, 1],
[1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0,
 0, 0, 0, 0, 0, 0, 0, 1],
[1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 1, 0, 0, 0, 1],
[1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
 0, 1, 0, 0, 0, 0, 1, 1],
[1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
```

```
 1, 1, 0, 0, 1, 0, 0, 1],
[1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
 1, 1, 0, 1, 1, 0, 0, 1],
[1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
 0, 0, 1, 0, 0, 1, 0, 1],
[1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
 1, 0, 0, 1, 0, 1, 0, 1],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1]])
```

2.3 3. Ship Layout

[5]:
```
def plot_grid(grid, title="Ship Layout"):
    plt.figure(figsize=(8, 8))
    plt.imshow(grid, cmap='binary', origin='upper')
    plt.title(title)
    plt.show()
```

[6]:
```
plot_grid(ship_grid)
```



3 Phase 1: Localisation

3.1 1. 'sense_distances' Function:

- This Function will sense distanes to blocked neighbors in all the four directions [up,down,left,right]
- This Function will return distances in form of Tuple

[7]:
```
def sense_distances(grid, pos):
    x, y = pos
    directions = {
        "up": (-1, 0),
```

```
        "down": (1, 0),
        "left": (0, -1),
        "right": (0, 1)
    }

    distances = []
    for dx, dy in directions.values():
        distance = 0
        nx, ny = x + dx, y + dy
        while 0 <= nx < GRID_SIZE and 0 <= ny < GRID_SIZE and grid[nx, ny] == 0:
            distance += 1
            nx += dx
            ny += dy
        distances.append(distance)
    return tuple(distances)
```

3.2 2. 'generate_distance_signatures' Function:

- This function will genarate a distance signatures for each open cells in the map
- This will create knowledge base for the bot to localised its position in the ship

[8]:
```
def generate_distance_signatures(grid):
    distance_signatures = {}
    for x in range(1, GRID_SIZE - 1):
        for y in range(1, GRID_SIZE - 1):
            if grid[x, y] == 0:
                signature = sense_distances(grid, (x, y))
                if signature not in distance_signatures:
                    distance_signatures[signature] = []
                distance_signatures[signature].append((x, y))
    return distance_signatures

distance_signatures = generate_distance_signatures(ship_grid)
```

[9]:
```
print(distance_signatures)
```

```
{(0, 0, 0, 3): [(1, 1), (11, 8), (11, 16), (20, 11)], (0, 1, 1, 2): [(1, 2), (5,
13), (20, 12), (24, 25)], (0, 0, 2, 1): [(1, 3), (13, 8), (20, 20), (27, 22)],
(0, 0, 3, 0): [(1, 4)], (0, 0, 0, 0): [(1, 6), (2, 5), (4, 17), (11, 22), (13,
16), (19, 10), (21, 9)], (0, 4, 0, 10): [(1, 8)], (0, 0, 1, 9): [(1, 9)], (0, 1,
2, 8): [(1, 10)], (0, 0, 3, 7): [(1, 11)], (0, 9, 4, 6): [(1, 12)], (0, 0, 5,
5): [(1, 13)], (0, 6, 6, 4): [(1, 14)], (0, 2, 7, 3): [(1, 15), (15, 8)], (0, 1,
8, 2): [(1, 16)], (0, 1, 9, 1): [(1, 17)], (0, 2, 10, 0): [(1, 18)], (0, 6, 0,
0): [(1, 20), (14, 5)], (0, 3, 0, 2): [(1, 22), (3, 6), (21, 26)], (0, 9, 1, 1):
[(1, 23)], (0, 4, 2, 0): [(1, 24), (16, 13)], (0, 0, 0, 2): [(1, 26), (6, 26)],
(0, 2, 1, 1): [(1, 27)], (0, 1, 2, 0): [(1, 28), (12, 6)], (1, 0, 0, 0): [(2,
2), (2, 10), (12, 19), (20, 8), (21, 12), (22, 15), (28, 26)], (0, 4, 0, 1):
[(2, 7), (18, 1)], (1, 3, 1, 0): [(2, 8)], (1, 8, 0, 0): [(2, 12)], (1, 5, 0,
```

11): [(2, 14)], (1, 1, 1, 10): [(2, 15)], (1, 0, 2, 9): [(2, 16)], (1, 0, 3, 8): [(2, 17), (6, 15)], (1, 1, 4, 7): [(2, 18)], (0, 4, 5, 6): [(2, 19)], (1, 5, 6, 5): [(2, 20)], (0, 0, 7, 4): [(2, 21)], (1, 2, 8, 3): [(2, 22)], (1, 8, 9, 2): [(2, 23)], (1, 3, 10, 1): [(2, 24)], (0, 3, 11, 0): [(2, 25)], (1, 1, 0, 1): [(2, 27)], (1, 0, 1, 0): [(2, 28), (25, 25)], (0, 7, 0, 0): [(3, 1), (16, 19)], (0, 0, 0, 1): [(3, 3), (7, 4), (10, 20), (11, 13), (22, 10), (27, 25)], (0, 1, 1, 0): [(3, 4), (22, 11), (27, 26)], (1, 3, 1, 1): [(3, 7), (3, 19)], (2, 2, 2, 0): [(3, 8)], (0, 1, 0, 4): [(3, 11)], (2, 7, 1, 3): [(3, 12)], (0, 0, 2, 2): [(3, 13)], (2, 4, 3, 1): [(3, 14)], (2, 0, 4, 0): [(3, 15), (15, 18)], (2, 0, 0, 2): [(3, 18)], (2, 4, 2, 0): [(3, 20)], (2, 1, 0, 5): [(3, 22)], (2, 7, 1, 4): [(3, 23)], (2, 2, 2, 3): [(3, 24)], (1, 2, 3, 2): [(3, 25)], (0, 0, 4, 1): [(3, 26), (15, 25), (28, 19)], (2, 0, 5, 0): [(3, 27)], (1, 6, 0, 1): [(4, 1), (22, 4)], (0, 7, 1, 0): [(4, 2)], (1, 0, 0, 4): [(4, 4)], (0, 3, 1, 3): [(4, 5)], (1, 2, 2, 2): [(4, 6)], (2, 2, 3, 1): [(4, 7)], (3, 1, 4, 0): [(4, 8)], (0, 5, 0, 2): [(4, 10)], (1, 0, 1, 1): [(4, 11), (12, 10)], (3, 6, 2, 0): [(4, 12)], (3, 3, 0, 0): [(4, 14)], (2, 2, 0, 6): [(4, 19)], (3, 3, 1, 5): [(4, 20)], (0, 0, 2, 4): [(4, 21)], (3, 0, 3, 3): [(4, 22)], (3, 6, 4, 2): [(4, 23)], (3, 1, 5, 1): [(4, 24)], (2, 1, 6, 0): [(4, 25)], (0, 2, 0, 0): [(4, 28), (8, 15), (13, 18), (26, 1)], (2, 5, 0, 1): [(5, 1), (19, 18)], (1, 6, 1, 0): [(5, 2)], (1, 2, 0, 3): [(5, 5)], (2, 1, 1, 2): [(5, 6), (19, 14)], (3, 1, 2, 1): [(5, 7)], (4, 0, 3, 0): [(5, 8)], (1, 4, 0, 0): [(5, 10), (7, 27), (19, 21), (24, 20)], (4, 5, 0, 3): [(5, 12)], (4, 2, 2, 1): [(5, 14), (17, 7)], (0, 1, 3, 0): [(5, 15), (11, 19), (27, 23)], (3, 1, 0, 1): [(5, 19)], (4, 2, 1, 0): [(5, 20)], (4, 5, 0, 2): [(5, 23)], (4, 0, 1, 1): [(5, 24)], (3, 0, 2, 0): [(5, 25)], (1, 1, 0, 0): [(5, 28), (27, 1)], (3, 4, 0, 2): [(6, 1)], (2, 5, 1, 1): [(6, 2)], (0, 0, 2, 0): [(6, 3), (9, 3)], (2, 1, 0, 2): [(6, 5), (23, 26)], (3, 0, 1, 1): [(6, 6), (12, 5)], (4, 0, 2, 0): [(6, 7)], (0, 3, 0, 1): [(6, 9), (13, 3)], (2, 3, 1, 0): [(6, 10), (8, 18), (8, 27)], (5, 4, 0, 11): [(6, 12)], (1, 0, 1, 10): [(6, 13)], (5, 1, 2, 9): [(6, 14)], (0, 1, 4, 7): [(6, 16)], (0, 6, 5, 6): [(6, 17)], (0, 5, 6, 5): [(6, 18)], (4, 0, 7, 4): [(6, 19)], (5, 1, 8, 3): [(6, 20)], (0, 4, 9, 2): [(6, 21)], (0, 1, 10, 1): [(6, 22)], (5, 4, 11, 0): [(6, 23)], (0, 5, 1, 1): [(6, 27), (13, 23), (21, 27), (23, 12)], (2, 0, 2, 0): [(6, 28)], (4, 3, 0, 1): [(7, 1)], (3, 4, 1, 0): [(7, 2), (19, 19)], (3, 0, 1, 0): [(7, 5), (20, 25), (28, 9)], (0, 2, 0, 4): [(7, 8)], (1, 2, 1, 3): [(7, 9), (26, 9)], (3, 2, 2, 2): [(7, 10)], (0, 5, 3, 1): [(7, 11)], (6, 3, 4, 0): [(7, 12)], (6, 0, 0, 0): [(7, 14), (12, 17)], (1, 0, 0, 2): [(7, 16), (23, 11)], (1, 5, 1, 1): [(7, 17)], (1, 4, 2, 0): [(7, 18)], (6, 0, 0, 5): [(7, 20)], (1, 3, 1, 4): [(7, 21)], (1, 0, 2, 3): [(7, 22), (14, 12)], (6, 3, 3, 2): [(7, 23)], (0, 8, 4, 1): [(7, 24)], (0, 0, 5, 0): [(7, 25), (8, 13), (9, 19), (10, 28), (23, 9)], (5, 2, 0, 1): [(8, 1)], (4, 3, 1, 0): [(8, 2)], (1, 1, 0, 5): [(8, 8), (15, 21)], (2, 1, 1, 4): [(8, 9)], (4, 1, 2, 3): [(8, 10)], (1, 4, 3, 2): [(8, 11)], (7, 2, 4, 1): [(8, 12)], (2, 4, 0, 1): [(8, 17)], (2, 2, 0, 0): [(8, 21)], (7, 2, 0, 1): [(8, 23)], (1, 7, 1, 0): [(8, 24)], (0, 11, 0, 1): [(8, 26)], (6, 1, 0, 2): [(9, 1)], (5, 2, 1, 1): [(9, 2), (22, 18)], (0, 3, 0, 0): [(9, 5), (13, 1), (22, 13), (24, 10)], (0, 1, 0, 5): [(9, 7), (14, 10), (17, 22)], (2, 0, 1, 4): [(9, 8), (15, 22)], (3, 0, 2, 3): [(9, 9)], (5, 0, 3, 2): [(9, 10)], (2, 3, 4, 1): [(9, 11)], (8, 1, 5, 0): [(9, 12)], (0, 0, 0, 5): [(9, 14)], (1, 1, 1, 4): [(9, 15), (23, 5)], (0, 0, 2, 3): [(9, 16)], (3, 3, 3, 2): [(9, 17)], (3, 2, 4, 1): [(9, 18)],

(3, 1, 0, 0): [(9, 21)], (8, 1, 0, 4): [(9, 23)], (2, 6, 1, 3): [(9, 24)], (0, 2, 2, 2): [(9, 25)], (1, 10, 3, 1): [(9, 26)], (3, 2, 4, 0): [(9, 27), (26, 12)], (7, 0, 0, 1): [(10, 1)], (6, 1, 1, 0): [(10, 2), (27, 4)], (1, 2, 0, 2): [(10, 5), (14, 1)], (0, 0, 1, 1): [(10, 6), (13, 27), (25, 3), (26, 5)], (1, 0, 2, 0): [(10, 7), (28, 13)], (3, 2, 0, 1): [(10, 11)], (9, 0, 1, 0): [(10, 12)], (2, 0, 0, 0): [(10, 15), (16, 21), (24, 22), (28, 6)], (4, 2, 0, 1): [(10, 17)], (4, 1, 1, 0): [(10, 18)], (4, 0, 1, 0): [(10, 21)], (9, 0, 0, 5): [(10, 23)], (3, 5, 1, 4): [(10, 24)], (1, 1, 2, 3): [(10, 25)], (2, 9, 3, 2): [(10, 26)], (4, 1, 4, 1): [(10, 27)], (7, 0, 0, 3): [(11, 2)], (0, 0, 1, 2): [(11, 3), (28, 2)], (0, 2, 2, 1): [(11, 4)], (2, 1, 3, 0): [(11, 5)], (0, 2, 1, 2): [(11, 9)], (0, 1, 2, 1): [(11, 10)], (4, 1, 3, 0): [(11, 11), (19, 16)], (0, 0, 1, 0): [(11, 14), (28, 24)], (5, 1, 1, 2): [(11, 17)], (5, 0, 2, 1): [(11, 18), (19, 15)], (4, 4, 0, 3): [(11, 24)], (2, 0, 1, 2): [(11, 25), (27, 21)], (3, 8, 2, 1): [(11, 26)], (5, 0, 3, 0): [(11, 27)], (1, 1, 0, 2): [(12, 4), (12, 9)], (5, 0, 2, 0): [(12, 11), (23, 28)], (5, 3, 0, 0): [(12, 24)], (4, 7, 0, 0): [(12, 26)], (2, 0, 1, 0): [(13, 4)], (1, 0, 0, 3): [(13, 6)], (0, 6, 1, 2): [(13, 7)], (2, 0, 3, 0): [(13, 9), (17, 8)], (0, 1, 0, 2): [(13, 12)], (0, 1, 1, 1): [(13, 13), (16, 12)], (0, 2, 2, 0): [(13, 14), (26, 6)], (0, 1, 0, 0): [(13, 20), (21, 15)], (0, 2, 0, 2): [(13, 22)], (6, 2, 2, 0): [(13, 24)], (5, 6, 0, 2): [(13, 26)], (0, 3, 2, 0): [(13, 28)], (0, 11, 1, 1): [(14, 2)], (1, 2, 2, 0): [(14, 3), (16, 17), (23, 13)], (1, 5, 0, 0): [(14, 7)], (0, 4, 1, 4): [(14, 11)], (1, 0, 3, 2): [(14, 13)], (1, 1, 4, 1): [(14, 14)], (0, 5, 5, 0): [(14, 15)], (1, 1, 0, 6): [(14, 18)], (0, 0, 1, 5): [(14, 19)], (1, 0, 2, 4): [(14, 20)], (0, 2, 3, 3): [(14, 21)], (1, 1, 4, 2): [(14, 22)], (1, 4, 5, 1): [(14, 23)], (7, 1, 6, 0): [(14, 24)], (6, 5, 0, 0): [(14, 26)], (1, 2, 0, 0): [(14, 28)], (2, 1, 0, 10): [(15, 1)], (1, 10, 1, 9): [(15, 2)], (2, 1, 2, 8): [(15, 3)], (0, 1, 3, 7): [(15, 4)], (1, 5, 4, 6): [(15, 5)], (0, 5, 5, 5): [(15, 6)], (2, 4, 6, 4): [(15, 7)], (0, 0, 8, 2): [(15, 9)], (1, 0, 9, 1): [(15, 10)], (1, 3, 10, 0): [(15, 11)], (2, 0, 0, 4): [(15, 14)], (1, 4, 1, 3): [(15, 15)], (0, 5, 2, 2): [(15, 16)], (0, 3, 3, 1): [(15, 17)], (2, 3, 2, 3): [(15, 23)], (8, 0, 3, 2): [(15, 24)], (7, 4, 5, 0): [(15, 26)], (2, 1, 0, 0): [(15, 28), (27, 28)], (3, 0, 0, 7): [(16, 1)], (2, 9, 1, 6): [(16, 2)], (3, 0, 2, 5): [(16, 3)], (1, 0, 3, 4): [(16, 4)], (2, 4, 4, 3): [(16, 5)], (1, 4, 5, 2): [(16, 6)], (3, 3, 6, 1): [(16, 7)], (1, 1, 7, 0): [(16, 8), (26, 21)], (2, 2, 0, 2): [(16, 11), (20, 1)], (2, 3, 0, 2): [(16, 15)], (1, 4, 1, 1): [(16, 16)], (3, 2, 0, 0): [(16, 23), (21, 21)], (8, 3, 0, 0): [(16, 26)], (3, 0, 0, 0): [(16, 28), (28, 28)], (3, 8, 0, 0): [(17, 2)], (3, 3, 0, 3): [(17, 5)], (2, 3, 1, 2): [(17, 6)], (0, 0, 0, 10): [(17, 10)], (3, 1, 1, 9): [(17, 11)], (1, 0, 2, 8): [(17, 12)], (1, 3, 3, 7): [(17, 13)], (0, 3, 4, 6): [(17, 14), (25, 11)], (3, 2, 5, 5): [(17, 15)], (2, 3, 6, 4): [(17, 16)], (2, 1, 7, 3): [(17, 17)], (0, 7, 8, 2): [(17, 18)], (1, 6, 9, 1): [(17, 19)], (0, 1, 10, 0): [(17, 20)], (4, 1, 1, 4): [(17, 23)], (0, 9, 2, 3): [(17, 24)], (0, 3, 3, 2): [(17, 25)], (9, 2, 4, 1): [(17, 26)], (0, 2, 5, 0): [(17, 27)], (4, 7, 1, 0): [(18, 2)], (4, 2, 0, 2): [(18, 5)], (3, 2, 1, 1): [(18, 6)], (5, 1, 2, 0): [(18, 7)], (4, 0, 0, 0): [(18, 11)], (2, 2, 0, 15): [(18, 13)], (1, 2, 1, 14): [(18, 14)], (4, 1, 2, 13): [(18, 15)], (3, 2, 3, 12): [(18, 16)], (3, 0, 4, 11): [(18, 17)], (1, 6, 5, 10): [(18, 18)], (2, 5, 6, 9): [(18, 19)], (1, 0, 7, 8): [(18, 20)], (0, 5, 8, 7): [(18, 21)], (1, 0, 9, 6): [(18, 22)], (5, 0, 10, 5): [(18, 23)], (1, 8, 11, 4): [(18, 24)], (1, 2, 12, 3):

[(18, 25)], (10, 1, 13, 2): [(18, 26)], (1, 1, 14, 1): [(18, 27)], (0, 5, 15, 0): [(18, 28)], (1, 3, 0, 7): [(19, 1)], (5, 6, 1, 6): [(19, 2)], (0, 2, 2, 5): [(19, 3)], (0, 0, 3, 4): [(19, 4)], (5, 1, 4, 3): [(19, 5)], (4, 1, 5, 2): [(19, 6)], (6, 0, 6, 1): [(19, 7)], (0, 1, 7, 0): [(19, 8), (27, 13)], (3, 1, 0, 3): [(19, 13), (21, 1)], (2, 7, 0, 4): [(19, 24)], (2, 1, 1, 3): [(19, 25)], (11, 0, 2, 2): [(19, 26)], (2, 0, 3, 1): [(19, 27)], (1, 4, 4, 0): [(19, 28), (24, 8)], (6, 5, 1, 1): [(20, 2)], (1, 1, 2, 0): [(20, 3)], (6, 0, 0, 1): [(20, 5)], (5, 0, 1, 0): [(20, 6)], (4, 0, 2, 1): [(20, 13)], (3, 0, 3, 0): [(20, 14)], (5, 0, 0, 0): [(20, 16)], (3, 4, 0, 3): [(20, 18)], (4, 3, 1, 2): [(20, 19)], (2, 3, 3, 0): [(20, 21)], (3, 6, 0, 1): [(20, 24)], (2, 3, 0, 0): [(20, 28)], (7, 4, 1, 2): [(21, 2)], (2, 0, 2, 1): [(21, 3)], (0, 7, 3, 0): [(21, 4)], (0, 4, 0, 0): [(21, 7)], (0, 7, 0, 2): [(21, 17)], (4, 3, 1, 1): [(21, 18)], (5, 2, 2, 0): [(21, 19)], (4, 5, 0, 0): [(21, 24)], (3, 2, 2, 0): [(21, 28)], (4, 0, 0, 1): [(22, 1)], (8, 3, 1, 0): [(22, 2)], (0, 2, 1, 0): [(22, 5), (25, 21)], (1, 3, 0, 0): [(22, 7)], (1, 6, 0, 2): [(22, 17)], (6, 1, 2, 0): [(22, 19), (27, 17)], (4, 1, 0, 7): [(22, 21)], (0, 2, 1, 6): [(22, 22)], (0, 1, 2, 5): [(22, 23)], (5, 4, 3, 4): [(22, 24)], (0, 0, 4, 3): [(22, 25), (26, 18)], (1, 2, 5, 2): [(22, 26)], (1, 4, 6, 1): [(22, 27)], (4, 1, 7, 0): [(22, 28)], (9, 2, 0, 0): [(23, 2)], (2, 5, 0, 5): [(23, 4)], (0, 1, 2, 3): [(23, 6)], (2, 2, 3, 2): [(23, 7)], (0, 5, 4, 1): [(23, 8)], (0, 5, 0, 8): [(23, 16)], (2, 5, 1, 7): [(23, 17)], (6, 1, 2, 6): [(23, 18)], (7, 0, 3, 5): [(23, 19)], (0, 5, 4, 4): [(23, 20)], (5, 0, 5, 3): [(23, 21)], (1, 1, 6, 2): [(23, 22)], (1, 0, 7, 1): [(23, 23)], (6, 3, 8, 0): [(23, 24)], (2, 3, 1, 1): [(23, 27)], (10, 1, 0, 0): [(24, 2)], (3, 4, 0, 4): [(24, 4)], (2, 0, 1, 3): [(24, 5)], (1, 0, 2, 2): [(24, 6)], (3, 1, 3, 1): [(24, 7)], (1, 4, 0, 6): [(24, 12)], (2, 1, 1, 5): [(24, 13)], (0, 2, 2, 4): [(24, 14)], (0, 4, 3, 3): [(24, 15)], (1, 4, 4, 2): [(24, 16)], (3, 4, 5, 1): [(24, 17)], (7, 0, 6, 0): [(24, 18)], (7, 2, 0, 3): [(24, 24)], (3, 0, 2, 1): [(24, 26)], (3, 2, 3, 0): [(24, 27)], (11, 0, 0, 2): [(25, 2)], (4, 3, 2, 0): [(25, 4)], (4, 0, 0, 10): [(25, 7)], (2, 3, 1, 9): [(25, 8)], (0, 3, 2, 8): [(25, 9)], (1, 2, 3, 7): [(25, 10)], (2, 3, 5, 5): [(25, 12)], (3, 0, 6, 4): [(25, 13)], (1, 1, 7, 3): [(25, 14)], (1, 3, 8, 2): [(25, 15)], (2, 3, 9, 1): [(25, 16)], (4, 3, 10, 0): [(25, 17)], (2, 3, 0, 1): [(25, 20)], (8, 1, 0, 1): [(25, 24)], (4, 1, 0, 1): [(25, 27)], (0, 3, 1, 0): [(25, 28)], (5, 2, 0, 2): [(26, 4)], (3, 2, 0, 4): [(26, 8)], (2, 1, 2, 2): [(26, 10)], (1, 2, 3, 1): [(26, 11)], (2, 0, 0, 7): [(26, 14)], (2, 2, 1, 6): [(26, 15)], (3, 2, 2, 5): [(26, 16)], (5, 2, 3, 4): [(26, 17)], (0, 0, 5, 2): [(26, 19)], (3, 2, 6, 1): [(26, 20)], (9, 0, 0, 0): [(26, 24)], (5, 0, 0, 1): [(26, 27), (28, 8)], (1, 2, 1, 0): [(26, 28)], (0, 1, 0, 1): [(27, 3)], (1, 1, 0, 7): [(27, 6)], (0, 0, 1, 6): [(27, 7)], (4, 1, 2, 5): [(27, 8)], (2, 1, 3, 4): [(27, 9)], (3, 0, 4, 3): [(27, 10)], (2, 1, 5, 2): [(27, 11)], (4, 1, 6, 1): [(27, 12)], (3, 1, 0, 2): [(27, 15)], (4, 1, 1, 1): [(27, 16)], (4, 1, 0, 3): [(27, 20)], (2, 0, 0, 3): [(28, 1)], (1, 0, 2, 1): [(28, 3)], (7, 0, 3, 0): [(28, 4)], (3, 0, 0, 2): [(28, 11)], (5, 0, 1, 1): [(28, 12)], (4, 0, 0, 5): [(28, 15)], (5, 0, 1, 4): [(28, 16)], (7, 0, 2, 3): [(28, 17)], (0, 0, 3, 2): [(28, 18)], (5, 0, 5, 0): [(28, 20)], (1, 0, 0, 1): [(28, 23)]}

3.3 3. 'attempt_random_move' Funtion:

- This function is basically used for the bot movement as it will attempt to move and update its position.

[10]:
```
def attempt_random_move(grid, position):
    x, y = position
    directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]  # Right, Left, Down, Up
    random.shuffle(directions)
    for dx, dy in directions:
        nx, ny = x + dx, y + dy
        if 0 <= nx < GRID_SIZE and 0 <= ny < GRID_SIZE and grid[nx, ny] == 0:
            return (nx, ny)
    return position
```

3.4 4. Localisation Function:

- This function will tried to help the bot to find its intial position on the ship.

[11]:
```
def localize_bot_with_random_movement(grid, distance_signatures):
    current_position = random.choice([loc for sig in distance_signatures.
 ↪values() for loc in sig])
    steps = 0
    while True:
        sensed_signature = sense_distances(grid, current_position)
        if sensed_signature in distance_signatures:
            possible_positions = distance_signatures[sensed_signature]
            print(f"Step {steps + 1}: Sensed Signature: {sensed_signature},␣
 ↪Possible Locations: {len(possible_positions)}")
            if len(possible_positions) == 1:
                print("Bot's position has been identified:",␣
 ↪possible_positions[0])
                return possible_positions[0]
            current_position = attempt_random_move(grid, current_position)
        steps += 1

bot_position = localize_bot_with_random_movement(ship_grid, distance_signatures)
```

```
Step 1: Sensed Signature: (0, 0, 1, 2), Possible Locations: 2
Step 2: Sensed Signature: (1, 0, 2, 1), Possible Locations: 1
Bot's position has been identified: (28, 3)
```

4 Phase 2 : Catching Rat

4.1 1. Intializing Rat's Position:

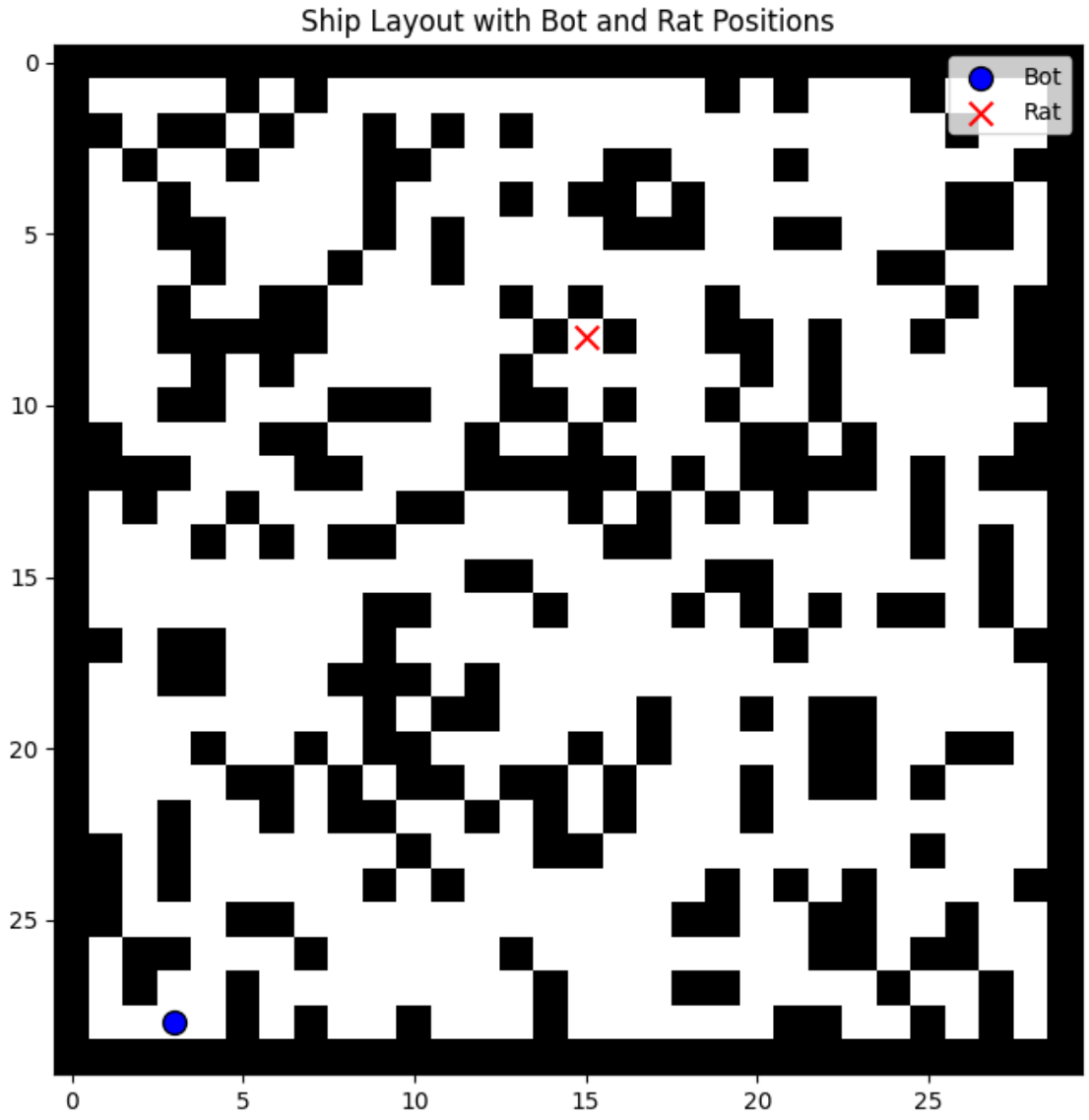
- While initializing the rat's positon we are also ensuring that every time the rat is place in the open cells of the grid

[12]:
```
#rat_position = (random.randint(1, GRID_SIZE - 2), random.randint(1, GRID_SIZE␣
↪- 2))

# Ensure the rat is only placed in open cells
while True:
    rat_position = (random.randint(1, GRID_SIZE - 2), random.randint(1,␣
↪GRID_SIZE - 2))
    if ship_grid[rat_position] == 0:  # Check if the cell is open
        break

prob_grid = np.full((GRID_SIZE, GRID_SIZE), 1 / (GRID_SIZE * GRID_SIZE))
```

4.2 2. Ship Updated Design with Bot and Rat:

[13]:
```
def plot_grid_with_positions(grid, bot_position, rat_position, title="Ship␣
↪Layout with Bot and Rat Positions"):
    plt.figure(figsize=(8, 8))
    plt.imshow(grid, cmap='binary', origin='upper')
    plt.scatter(bot_position[1], bot_position[0], color='blue', label='Bot',␣
↪s=100, edgecolors='black')  # Bot in blue
    plt.scatter(rat_position[1], rat_position[0], color='red', label='Rat',␣
↪s=100, marker='x')  # Rat in red
    plt.legend(loc="upper right")
    plt.title(title)
    plt.show()
```

[14]:
```
plot_grid_with_positions(ship_grid, bot_position, rat_position)
```



4.3 3. Manhattan Distance

[15]:
```
def manhattan_distance(cell1, cell2):
    return abs(cell1[0] - cell2[0]) + abs(cell1[1] - cell2[1])
```

4.4 4. Ping Probability Function:

- This Function will calculate probability of hearing a ping based on the distance

[16]:
```
def ping_probability(distance, alpha):
    return math.exp(-alpha * (distance - 1))
```

[17]:
```python
# Update probability grid based on ping result

def update_probabilities(prob_grid, bot_pos, heard_ping, alpha, grid):
    new_prob_grid = np.zeros_like(prob_grid)
    total_prob = 0
    for x in range(GRID_SIZE):
        for y in range(GRID_SIZE):
            if grid[x, y] == 0:  # Only open cells
                dist = manhattan_distance(bot_pos, (x, y))
                prob_ping = ping_probability(dist, alpha)
                if heard_ping:
                    new_prob_grid[x, y] = prob_grid[x, y] * prob_ping
                else:
                    new_prob_grid[x, y] = prob_grid[x, y] * (1 - prob_ping)
                total_prob += new_prob_grid[x, y]
    new_prob_grid /= total_prob  # Normalize probabilities
    return new_prob_grid
```

4.5 5. Determining Rat's most likely position in the Grid:

[18]:
```python
def most_likely_rat_location(prob_grid):
    return np.unravel_index(np.argmax(prob_grid, axis=None), prob_grid.shape)
```

4.6 6. Simulate Rat with the Detector:

[19]:
```python
def space_rat_detector(bot_pos, rat_pos, alpha):
    distance = manhattan_distance(bot_pos, rat_pos)
    prob = ping_probability(distance, alpha)
    return random.random() < prob
```

4.7 Case 1: Non-Moving Space Rat Situation:

4.7.1 Baseline Bot:

Actions performed by the baseline bot are as below:

Step 1: Listen for a Ping using 'heard_ping'

Step 2: Update the probability grid based on the ping using 'prob_grid'

Step 3: Identify and Commit to the most likely cell using 'target_pos'

Step 4: Performs movemen to the target cell and returned the position at which the Rat was caught

[20]:
```python
def baselinebot(grid, bot_pos, rat_pos, prob_grid, alpha):
    actions_taken = 0
    while True:
        # Step 1: Listen for a ping
        heard_ping = space_rat_detector(bot_pos, rat_pos, alpha)
```

```
        # Step 2: Update probability grid based on ping
        prob_grid = update_probabilities(prob_grid, bot_pos, heard_ping, alpha,␣
 ↪grid)
        # Step 3: Identify and commit to the most likely cell
        target_pos = most_likely_rat_location(prob_grid)
        # Step 4: Move all the way to the target cell
        while bot_pos != target_pos:
            # Determine the direction to move toward the target cell
            bot_x, bot_y = bot_pos
            target_x, target_y = target_pos
            if target_x > bot_x:
                bot_pos = (bot_x + 1, bot_y)
            elif target_x < bot_x:
                bot_pos = (bot_x - 1, bot_y)
            elif target_y > bot_y:
                bot_pos = (bot_x, bot_y + 1)
            else:
                bot_pos = (bot_x, bot_y - 1)
            actions_taken += 1
            # Check if bot has reached the rat's position
            if bot_pos == rat_pos:
                print(f"Rat caught at position {bot_pos} after {actions_taken}␣
 ↪actions!")
                return actions_taken
```

[21]:
```
actions_taken = baselinebot(ship_grid, bot_position, rat_position, prob_grid,␣
 ↪alpha)
print("Total actions taken to catch the rat:", actions_taken)
```

```
Rat caught at position (8, 15) after 164 actions!
Total actions taken to catch the rat: 164
```

4.7.2 Our Bot:

[22]:
```
def a_star(grid, start, goal):
    # Use defaultdict to avoid key checks
    g_score = defaultdict(lambda: float('inf'))
    g_score[start] = 0
    f_score = defaultdict(lambda: float('inf'))
    f_score[start] = manhattan_distance(start, goal)
    # Pre-compute grid dimensions
    rows, cols = len(grid), len(grid[0])
    # Pre-compute valid moves
    MOVES = [(0, 1), (0, -1), (1, 0), (-1, 0)]
    open_set = [(f_score[start], start)]
    came_from = {}
    while open_set:
```

```
        current = heapq.heappop(open_set)[1]
        if current == goal:
            # Reconstruct path more efficiently
            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            return path[::-1]  # Reverse path more efficiently
        current_g = g_score[current]
        # Optimize neighbor checking
        x, y = current
        for dx, dy in MOVES:
            nx, ny = x + dx, y + dy
            if (0 <= nx < rows and 0 <= ny < cols and grid[nx][ny] == 0):
                neighbor = (nx, ny)
                tentative_g = current_g + 1
                if tentative_g < g_score[neighbor]:
                    came_from[neighbor] = current
                    g_score[neighbor] = tentative_g
                    f = tentative_g + manhattan_distance(neighbor, goal)
                    f_score[neighbor] = f
                    heapq.heappush(open_set, (f, neighbor))
    return None
```

[23]:
```
def ourbot(grid, bot_pos, rat_pos, prob_grid, alpha, threshold_factor=0.8):
    actions_taken = 0
    rows, cols = len(grid), len(grid[0])
    # Pre-compute grid properties
    mean_prob = np.mean(prob_grid)
    last_target = None
    consecutive_failed_attempts = 0
    MAX_FAILED_ATTEMPTS = 3
    # Create a memory of visited high-probability locations
    visited_targets = set()
    while actions_taken < rows * cols * 2:  # Add reasonable maximum steps
        heard_ping = space_rat_detector(bot_pos, rat_pos, alpha)
        prob_grid = update_probabilities(prob_grid, bot_pos, heard_ping, alpha,␣
↪grid)
        # Get top N most likely positions
        flat_indices = np.argsort(prob_grid.ravel())[-5:]  # Consider top 5␣
↪positions
        potential_targets = [(idx // cols, idx % cols) for idx in flat_indices]
        # Filter out previously visited targets and blocked cells
        valid_targets = [pos for pos in potential_targets
                         if pos not in visited_targets
                         and grid[pos[0]][pos[1]] == 0
                         and prob_grid[pos] >= threshold_factor * mean_prob]
```

```
        if valid_targets:
            target_pos = valid_targets[-1]  # Most likely unvisited position
            path = a_star(grid, bot_pos, target_pos)
            if path:
                # Move along path efficiently
                next_pos = path[0]
                bot_pos = next_pos
                actions_taken += 1
                if bot_pos == rat_pos:
                    return actions_taken
                if bot_pos == target_pos:
                    visited_targets.add(target_pos)
                    consecutive_failed_attempts = 0
            else:
                consecutive_failed_attempts += 1
                if consecutive_failed_attempts >= MAX_FAILED_ATTEMPTS:
                    # Reset strategy if stuck
                    visited_targets.clear()
                    consecutive_failed_attempts = 0
                # Smart random move based on probability gradient
                possible_moves = [(bot_pos[0] + dx, bot_pos[1] + dy) for dx, dy␣
↪in [(0, 1), (0, -1), (1, 0), (-1, 0)]]
                valid_moves = [(x, y) for x, y in possible_moves
                               if 0 <= x < rows and 0 <= y < cols and grid[x][y]␣
↪== 0]
                if valid_moves:
                    bot_pos = max(valid_moves, key=lambda pos: prob_grid[pos])
                    actions_taken += 1
        else:
            # Reset visited targets if all high-probability locations have been␣
↪checked
            visited_targets.clear()
        if actions_taken % (rows * cols // 4) == 0:
            # Periodically reset probability grid to avoid getting stuck in␣
↪local maxima
            prob_grid = np.ones_like(prob_grid) / (rows * cols)
    return actions_taken
```

[24]:
```
actions_taken_by_ourbot = ourbot(ship_grid, bot_position, rat_position,␣
 ↪prob_grid, alpha)
print("Total actions taken to catch the rat:", actions_taken_by_ourbot)
```

```
Total actions taken to catch the rat: 62
```

4.8 Case 2: Moving Space Rat Situation:

4.8.1 1. Rat Movement Function:

[25]:
```python
def move_rat(grid, rat_pos):
    directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]  # Right, Left, Down, Up
    random.shuffle(directions)  # Shuffle to pick a random valid move
    for dx, dy in directions:
        nx, ny = rat_pos[0] + dx, rat_pos[1] + dy
        if 0 <= nx < GRID_SIZE and 0 <= ny < GRID_SIZE and grid[nx, ny] == 0:
            return (nx, ny)  # Return new position if open
    return rat_pos  # Stay in the same position if no open cells
```

4.8.2 2. Baseline Bot:

[26]:
```python
def baselinebot_rat_movement(grid, bot_pos, rat_pos, prob_grid, alpha):
    actions_taken = 0
    while True:
        # Step 1: Move the rat
        rat_pos = move_rat(grid, rat_pos)
        # Step 2: Listen for a ping
        heard_ping = space_rat_detector(bot_pos, rat_pos, alpha)
        # Step 3: Update probability grid based on ping
        prob_grid = update_probabilities(prob_grid, bot_pos, heard_ping, alpha,␣
 ↪grid)
        # Step 4: Find the most likely position of the rat
        target_pos = most_likely_rat_location(prob_grid)
        print(f"Bot Position: {bot_pos}, Rat Position: {rat_pos}")
        # Move the bot step by step to the target position
        while bot_pos != target_pos:
            bot_x, bot_y = bot_pos
            target_x, target_y = target_pos
            # Determine the direction to move
            if target_x > bot_x:
                bot_pos = (bot_x + 1, bot_y)
            elif target_x < bot_x:
                bot_pos = (bot_x - 1, bot_y)
            elif target_y > bot_y:
                bot_pos = (bot_x, bot_y + 1)
            elif target_y < bot_y:
                bot_pos = (bot_x, bot_y - 1)
            actions_taken += 1
            #print(f"Moving Bot to {bot_pos}, Rat Position: {rat_pos}, Actions␣
 ↪Taken: {actions_taken}")
            # Check if bot has caught the rat
            if bot_pos == rat_pos:
                print(f"Rat caught at position {bot_pos} after {actions_taken}␣
 ↪actions!")
```

```
                return actions_taken
            # Update rat position dynamically
            rat_pos = move_rat(grid, rat_pos)
            # Recalculate target position if the rat moves
            prob_grid = update_probabilities(prob_grid, bot_pos, heard_ping,␣
 ↪alpha, grid)
            target_pos = most_likely_rat_location(prob_grid)
```

[27]:
```
actions_taken_by_baselinebot_with_rat_movement =␣
 ↪baselinebot_rat_movement(ship_grid, bot_position, rat_position, prob_grid,␣
 ↪alpha)
actions_taken_by_baselinebot_with_rat_movement
```

```
Bot Position: (28, 3), Rat Position: (9, 15)
Rat caught at position (21, 4) after 3626 actions!
```

[27]: 3626

4.8.3 3. Our Bot:

[28]:
```
def ourbot_rat_movement(grid, bot_pos, rat_pos, prob_grid, alpha,␣
 ↪threshold_factor=0.8):
    actions_taken = 0
    rows, cols = len(grid), len(grid[0])
    # Pre-compute grid properties
    mean_prob = np.mean(prob_grid)
    last_target = None
    consecutive_failed_attempts = 0
    MAX_FAILED_ATTEMPTS = 3
    # Create a memory of visited high-probability locations
    visited_targets = set()
    while actions_taken < rows * cols * 2:  # Add reasonable maximum steps
        rat_pos = move_rat(grid, rat_pos)
        heard_ping = space_rat_detector(bot_pos, rat_pos, alpha)
        prob_grid = update_probabilities(prob_grid, bot_pos, heard_ping, alpha,␣
 ↪grid)
        print(f"Bot Position: {bot_pos}, Rat Position: {rat_pos}, Actions Taken:
 ↪ {actions_taken}")
        # Get top N most likely positions
        flat_indices = np.argsort(prob_grid.ravel())[-5:]  # Consider top 5␣
 ↪positions
        potential_targets = [(idx // cols, idx % cols) for idx in flat_indices]
        # Filter out previously visited targets and blocked cells
        valid_targets = [pos for pos in potential_targets
                         if pos not in visited_targets
                         and grid[pos[0]][pos[1]] == 0
                         and prob_grid[pos] >= threshold_factor * mean_prob]
```

```
        if valid_targets:
            target_pos = valid_targets[-1]  # Most likely unvisited position
            path = a_star(grid, bot_pos, target_pos)
            if path:
                # Move along path efficiently
                next_pos = path[0]
                bot_pos = next_pos
                actions_taken += 1
                rat_pos = move_rat(grid, rat_pos)  # Move the rat at each bot␣
↪step
                print(f"Bot Position: {bot_pos}, Rat Position: {rat_pos},␣
↪Actions Taken: {actions_taken}")
                if bot_pos == rat_pos:
                    print(f"Rat caught at position {bot_pos} after␣
↪{actions_taken} actions!")
                    return actions_taken
                if bot_pos == target_pos:
                    visited_targets.add(target_pos)
                    consecutive_failed_attempts = 0
            else:
                consecutive_failed_attempts += 1
                if consecutive_failed_attempts >= MAX_FAILED_ATTEMPTS:
                    # Reset strategy if stuck
                    visited_targets.clear()
                    consecutive_failed_attempts = 0
                # Smart random move based on probability gradient
                possible_moves = [(bot_pos[0] + dx, bot_pos[1] + dy) for dx, dy␣
↪in [(0, 1), (0, -1), (1, 0), (-1, 0)]]
                valid_moves = [(x, y) for x, y in possible_moves
                               if 0 <= x < rows and 0 <= y < cols and grid[x][y]␣
↪== 0]
                if valid_moves:
                    bot_pos = max(valid_moves, key=lambda pos: prob_grid[pos])
                    actions_taken += 1
        else:
            # Reset visited targets if all high-probability locations have been␣
↪checked
            visited_targets.clear()
        if actions_taken % (rows * cols // 4) == 0:
            # Periodically reset probability grid to avoid getting stuck in␣
↪local maxima
            prob_grid = np.ones_like(prob_grid) / (rows * cols)
    return actions_taken
```

[29]:
```
actions_taken_by_ourbot_with_rat_movement = ourbot_rat_movement(ship_grid,␣
↪bot_position, rat_position, prob_grid, alpha)
```

```
actions_taken_by_ourbot_with_rat_movement
```

```
Bot Position: (28, 3), Rat Position: (9, 15), Actions Taken: 0
Bot Position: (27, 3), Rat Position: (9, 14), Actions Taken: 1
Bot Position: (27, 3), Rat Position: (9, 15), Actions Taken: 1
Bot Position: (27, 4), Rat Position: (8, 15), Actions Taken: 2
Bot Position: (27, 4), Rat Position: (9, 15), Actions Taken: 2
Bot Position: (26, 4), Rat Position: (9, 14), Actions Taken: 3
Bot Position: (26, 4), Rat Position: (9, 15), Actions Taken: 3
Bot Position: (25, 4), Rat Position: (9, 14), Actions Taken: 4
Bot Position: (25, 4), Rat Position: (9, 15), Actions Taken: 4
Bot Position: (24, 4), Rat Position: (9, 14), Actions Taken: 5
Bot Position: (24, 4), Rat Position: (9, 15), Actions Taken: 5
Bot Position: (23, 4), Rat Position: (8, 15), Actions Taken: 6
Bot Position: (23, 4), Rat Position: (9, 15), Actions Taken: 6
Bot Position: (22, 4), Rat Position: (8, 15), Actions Taken: 7
Bot Position: (22, 4), Rat Position: (9, 15), Actions Taken: 7
Bot Position: (21, 4), Rat Position: (9, 14), Actions Taken: 8
Bot Position: (21, 4), Rat Position: (9, 15), Actions Taken: 8
Bot Position: (21, 3), Rat Position: (9, 16), Actions Taken: 9
Bot Position: (21, 3), Rat Position: (9, 17), Actions Taken: 9
Bot Position: (20, 3), Rat Position: (10, 17), Actions Taken: 10
Bot Position: (20, 3), Rat Position: (11, 17), Actions Taken: 10
Bot Position: (20, 2), Rat Position: (12, 17), Actions Taken: 11
Bot Position: (20, 3), Rat Position: (11, 17), Actions Taken: 12
Bot Position: (20, 2), Rat Position: (11, 16), Actions Taken: 13
Bot Position: (20, 3), Rat Position: (11, 17), Actions Taken: 14
Bot Position: (20, 3), Rat Position: (11, 16), Actions Taken: 14
Bot Position: (19, 3), Rat Position: (11, 17), Actions Taken: 15
Bot Position: (19, 3), Rat Position: (12, 17), Actions Taken: 15
Bot Position: (19, 4), Rat Position: (11, 17), Actions Taken: 16
Bot Position: (19, 4), Rat Position: (11, 16), Actions Taken: 16
Bot Position: (19, 5), Rat Position: (11, 17), Actions Taken: 17
Bot Position: (19, 5), Rat Position: (11, 16), Actions Taken: 17
Bot Position: (18, 5), Rat Position: (11, 17), Actions Taken: 18
Bot Position: (18, 5), Rat Position: (11, 16), Actions Taken: 18
Bot Position: (17, 5), Rat Position: (11, 17), Actions Taken: 19
Bot Position: (17, 5), Rat Position: (12, 17), Actions Taken: 19
Bot Position: (17, 6), Rat Position: (11, 17), Actions Taken: 20
Bot Position: (17, 6), Rat Position: (10, 17), Actions Taken: 20
Bot Position: (17, 7), Rat Position: (9, 17), Actions Taken: 21
Bot Position: (17, 7), Rat Position: (9, 18), Actions Taken: 21
Bot Position: (16, 7), Rat Position: (9, 17), Actions Taken: 22
Bot Position: (16, 7), Rat Position: (9, 18), Actions Taken: 22
Bot Position: (16, 8), Rat Position: (9, 19), Actions Taken: 23
Bot Position: (16, 8), Rat Position: (9, 18), Actions Taken: 23
Bot Position: (15, 8), Rat Position: (9, 17), Actions Taken: 24
```

```
Bot Position: (15, 8), Rat Position: (9, 16), Actions Taken: 24
Bot Position: (15, 7), Rat Position: (9, 15), Actions Taken: 25
Bot Position: (15, 7), Rat Position: (9, 14), Actions Taken: 25
Bot Position: (15, 8), Rat Position: (9, 15), Actions Taken: 26
Bot Position: (15, 8), Rat Position: (9, 16), Actions Taken: 26
Bot Position: (15, 7), Rat Position: (9, 15), Actions Taken: 27
Bot Position: (15, 7), Rat Position: (9, 16), Actions Taken: 27
Bot Position: (15, 8), Rat Position: (9, 15), Actions Taken: 28
Bot Position: (15, 8), Rat Position: (10, 15), Actions Taken: 28
Bot Position: (15, 9), Rat Position: (9, 15), Actions Taken: 29
Bot Position: (15, 9), Rat Position: (10, 15), Actions Taken: 29
Bot Position: (15, 8), Rat Position: (9, 15), Actions Taken: 30
Bot Position: (15, 8), Rat Position: (8, 15), Actions Taken: 30
Bot Position: (15, 9), Rat Position: (9, 15), Actions Taken: 31
Bot Position: (15, 8), Rat Position: (8, 15), Actions Taken: 32
Bot Position: (15, 8), Rat Position: (9, 15), Actions Taken: 32
Bot Position: (15, 9), Rat Position: (9, 14), Actions Taken: 33
Bot Position: (15, 9), Rat Position: (9, 15), Actions Taken: 33
Bot Position: (15, 8), Rat Position: (10, 15), Actions Taken: 34
Bot Position: (15, 8), Rat Position: (9, 15), Actions Taken: 34
Bot Position: (15, 7), Rat Position: (10, 15), Actions Taken: 35
Bot Position: (15, 7), Rat Position: (9, 15), Actions Taken: 35
Bot Position: (14, 7), Rat Position: (10, 15), Actions Taken: 36
Bot Position: (14, 7), Rat Position: (9, 15), Actions Taken: 36
Bot Position: (13, 7), Rat Position: (9, 16), Actions Taken: 37
Bot Position: (13, 7), Rat Position: (9, 15), Actions Taken: 37
Bot Position: (13, 8), Rat Position: (8, 15), Actions Taken: 38
Bot Position: (13, 8), Rat Position: (9, 15), Actions Taken: 38
Bot Position: (13, 9), Rat Position: (10, 15), Actions Taken: 39
Bot Position: (13, 9), Rat Position: (9, 15), Actions Taken: 39
Bot Position: (12, 9), Rat Position: (9, 16), Actions Taken: 40
Bot Position: (12, 9), Rat Position: (9, 15), Actions Taken: 40
Bot Position: (11, 9), Rat Position: (9, 16), Actions Taken: 41
Bot Position: (11, 9), Rat Position: (9, 17), Actions Taken: 41
Bot Position: (11, 10), Rat Position: (10, 17), Actions Taken: 42
Bot Position: (11, 10), Rat Position: (10, 18), Actions Taken: 42
Bot Position: (12, 10), Rat Position: (10, 17), Actions Taken: 43
Bot Position: (12, 10), Rat Position: (10, 18), Actions Taken: 43
Bot Position: (12, 11), Rat Position: (9, 18), Actions Taken: 44
Bot Position: (12, 11), Rat Position: (9, 17), Actions Taken: 44
Bot Position: (11, 11), Rat Position: (8, 17), Actions Taken: 45
Bot Position: (11, 11), Rat Position: (8, 18), Actions Taken: 45
Bot Position: (10, 11), Rat Position: (7, 18), Actions Taken: 46
Bot Position: (10, 11), Rat Position: (6, 18), Actions Taken: 46
Bot Position: (11, 11), Rat Position: (6, 17), Actions Taken: 47
Bot Position: (11, 11), Rat Position: (6, 16), Actions Taken: 47
Bot Position: (12, 11), Rat Position: (7, 16), Actions Taken: 48
Bot Position: (12, 11), Rat Position: (7, 17), Actions Taken: 48
```

```
Bot Position: (12, 10), Rat Position: (6, 17), Actions Taken: 49
Bot Position: (12, 10), Rat Position: (7, 17), Actions Taken: 49
Bot Position: (12, 9), Rat Position: (8, 17), Actions Taken: 50
Bot Position: (12, 9), Rat Position: (7, 17), Actions Taken: 50
Bot Position: (13, 9), Rat Position: (6, 17), Actions Taken: 51
Bot Position: (13, 9), Rat Position: (6, 16), Actions Taken: 51
Bot Position: (13, 8), Rat Position: (7, 16), Actions Taken: 52
Bot Position: (13, 8), Rat Position: (6, 16), Actions Taken: 52
Bot Position: (13, 7), Rat Position: (6, 15), Actions Taken: 53
Bot Position: (13, 7), Rat Position: (6, 14), Actions Taken: 53
Bot Position: (14, 7), Rat Position: (7, 14), Actions Taken: 54
Bot Position: (14, 7), Rat Position: (6, 14), Actions Taken: 54
Bot Position: (15, 7), Rat Position: (7, 14), Actions Taken: 55
Bot Position: (15, 7), Rat Position: (6, 14), Actions Taken: 55
Bot Position: (15, 8), Rat Position: (6, 13), Actions Taken: 56
Bot Position: (15, 8), Rat Position: (6, 14), Actions Taken: 56
Bot Position: (15, 9), Rat Position: (6, 13), Actions Taken: 57
Bot Position: (15, 9), Rat Position: (6, 12), Actions Taken: 57
Bot Position: (15, 10), Rat Position: (5, 12), Actions Taken: 58
Bot Position: (15, 10), Rat Position: (6, 12), Actions Taken: 58
Bot Position: (14, 10), Rat Position: (7, 12), Actions Taken: 59
Bot Position: (14, 10), Rat Position: (7, 11), Actions Taken: 59
Bot Position: (14, 11), Rat Position: (7, 12), Actions Taken: 60
Bot Position: (14, 11), Rat Position: (6, 12), Actions Taken: 60
Bot Position: (14, 12), Rat Position: (7, 12), Actions Taken: 61
Bot Position: (14, 12), Rat Position: (6, 12), Actions Taken: 61
Bot Position: (14, 13), Rat Position: (5, 12), Actions Taken: 62
Bot Position: (14, 13), Rat Position: (5, 13), Actions Taken: 62
Bot Position: (14, 14), Rat Position: (5, 14), Actions Taken: 63
Bot Position: (14, 14), Rat Position: (5, 15), Actions Taken: 63
Bot Position: (14, 15), Rat Position: (6, 15), Actions Taken: 64
Bot Position: (14, 15), Rat Position: (5, 15), Actions Taken: 64
Bot Position: (14, 14), Rat Position: (6, 15), Actions Taken: 65
Bot Position: (14, 14), Rat Position: (6, 16), Actions Taken: 65
Bot Position: (14, 15), Rat Position: (7, 16), Actions Taken: 66
Bot Position: (14, 15), Rat Position: (7, 17), Actions Taken: 66
Bot Position: (15, 15), Rat Position: (8, 17), Actions Taken: 67
Bot Position: (15, 15), Rat Position: (8, 18), Actions Taken: 67
Bot Position: (15, 16), Rat Position: (8, 17), Actions Taken: 68
Bot Position: (15, 16), Rat Position: (8, 18), Actions Taken: 68
Bot Position: (15, 17), Rat Position: (9, 18), Actions Taken: 69
Bot Position: (15, 17), Rat Position: (9, 19), Actions Taken: 69
Bot Position: (15, 18), Rat Position: (9, 18), Actions Taken: 70
Bot Position: (15, 18), Rat Position: (8, 18), Actions Taken: 70
Bot Position: (14, 18), Rat Position: (7, 18), Actions Taken: 71
Bot Position: (14, 18), Rat Position: (7, 17), Actions Taken: 71
Bot Position: (14, 19), Rat Position: (7, 18), Actions Taken: 72
Bot Position: (14, 19), Rat Position: (6, 18), Actions Taken: 72
```

```
Bot Position: (14, 20), Rat Position: (6, 19), Actions Taken: 73
Bot Position: (14, 20), Rat Position: (6, 20), Actions Taken: 73
Bot Position: (14, 21), Rat Position: (6, 19), Actions Taken: 74
Bot Position: (14, 21), Rat Position: (5, 19), Actions Taken: 74
Bot Position: (14, 22), Rat Position: (4, 19), Actions Taken: 75
Bot Position: (14, 22), Rat Position: (4, 20), Actions Taken: 75
Bot Position: (13, 22), Rat Position: (5, 20), Actions Taken: 76
Bot Position: (13, 22), Rat Position: (6, 20), Actions Taken: 76
Bot Position: (13, 23), Rat Position: (6, 19), Actions Taken: 77
Bot Position: (13, 23), Rat Position: (5, 19), Actions Taken: 77
Bot Position: (13, 24), Rat Position: (4, 19), Actions Taken: 78
Bot Position: (13, 24), Rat Position: (5, 19), Actions Taken: 78
Bot Position: (12, 24), Rat Position: (4, 19), Actions Taken: 79
Bot Position: (13, 24), Rat Position: (3, 19), Actions Taken: 80
Bot Position: (13, 24), Rat Position: (3, 20), Actions Taken: 80
Bot Position: (12, 24), Rat Position: (4, 20), Actions Taken: 81
Bot Position: (12, 24), Rat Position: (5, 20), Actions Taken: 81
Bot Position: (11, 24), Rat Position: (4, 20), Actions Taken: 82
Bot Position: (11, 24), Rat Position: (4, 19), Actions Taken: 82
Bot Position: (10, 24), Rat Position: (5, 19), Actions Taken: 83
Bot Position: (10, 24), Rat Position: (5, 20), Actions Taken: 83
Bot Position: (9, 24), Rat Position: (4, 20), Actions Taken: 84
Bot Position: (9, 24), Rat Position: (4, 19), Actions Taken: 84
Bot Position: (9, 23), Rat Position: (3, 19), Actions Taken: 85
Bot Position: (9, 23), Rat Position: (2, 19), Actions Taken: 85
Bot Position: (8, 23), Rat Position: (3, 19), Actions Taken: 86
Bot Position: (8, 23), Rat Position: (3, 20), Actions Taken: 86
Bot Position: (7, 23), Rat Position: (4, 20), Actions Taken: 87
Bot Position: (7, 23), Rat Position: (4, 19), Actions Taken: 87
Bot Position: (7, 22), Rat Position: (5, 19), Actions Taken: 88
Bot Position: (7, 22), Rat Position: (5, 20), Actions Taken: 88
Bot Position: (7, 21), Rat Position: (5, 19), Actions Taken: 89
Bot Position: (7, 21), Rat Position: (4, 19), Actions Taken: 89
Bot Position: (7, 20), Rat Position: (3, 19), Actions Taken: 90
Bot Position: (7, 20), Rat Position: (4, 19), Actions Taken: 90
Bot Position: (6, 20), Rat Position: (3, 19), Actions Taken: 91
Bot Position: (6, 20), Rat Position: (3, 20), Actions Taken: 91
Bot Position: (6, 19), Rat Position: (3, 19), Actions Taken: 92
Bot Position: (6, 19), Rat Position: (3, 20), Actions Taken: 92
Bot Position: (6, 18), Rat Position: (3, 19), Actions Taken: 93
Bot Position: (6, 18), Rat Position: (3, 20), Actions Taken: 93
Bot Position: (7, 18), Rat Position: (4, 20), Actions Taken: 94
Bot Position: (7, 18), Rat Position: (5, 20), Actions Taken: 94
Bot Position: (8, 18), Rat Position: (6, 20), Actions Taken: 95
Bot Position: (8, 18), Rat Position: (5, 20), Actions Taken: 95
Bot Position: (9, 18), Rat Position: (5, 19), Actions Taken: 96
Bot Position: (9, 18), Rat Position: (4, 19), Actions Taken: 96
Bot Position: (10, 18), Rat Position: (3, 19), Actions Taken: 97
```

```
Bot Position: (10, 18), Rat Position: (3, 20), Actions Taken: 97
Bot Position: (9, 18), Rat Position: (4, 20), Actions Taken: 98
Bot Position: (9, 18), Rat Position: (4, 19), Actions Taken: 98
Bot Position: (8, 18), Rat Position: (5, 19), Actions Taken: 99
Bot Position: (8, 18), Rat Position: (5, 20), Actions Taken: 99
Bot Position: (8, 17), Rat Position: (5, 19), Actions Taken: 100
Bot Position: (8, 18), Rat Position: (6, 19), Actions Taken: 101
Bot Position: (8, 18), Rat Position: (5, 19), Actions Taken: 101
Bot Position: (9, 18), Rat Position: (6, 19), Actions Taken: 102
Bot Position: (9, 18), Rat Position: (6, 20), Actions Taken: 102
Bot Position: (9, 19), Rat Position: (7, 20), Actions Taken: 103
Bot Position: (9, 19), Rat Position: (6, 20), Actions Taken: 103
Bot Position: (9, 18), Rat Position: (5, 20), Actions Taken: 104
Bot Position: (9, 18), Rat Position: (4, 20), Actions Taken: 104
Bot Position: (8, 18), Rat Position: (5, 20), Actions Taken: 105
Bot Position: (8, 18), Rat Position: (5, 19), Actions Taken: 105
Bot Position: (8, 17), Rat Position: (5, 20), Actions Taken: 106
Bot Position: (8, 17), Rat Position: (4, 20), Actions Taken: 106
Bot Position: (8, 18), Rat Position: (4, 19), Actions Taken: 107
Bot Position: (8, 18), Rat Position: (4, 20), Actions Taken: 107
Bot Position: (7, 18), Rat Position: (4, 19), Actions Taken: 108
Bot Position: (7, 18), Rat Position: (5, 19), Actions Taken: 108
Bot Position: (6, 18), Rat Position: (5, 20), Actions Taken: 109
Bot Position: (6, 18), Rat Position: (5, 19), Actions Taken: 109
Bot Position: (6, 19), Rat Position: (5, 20), Actions Taken: 110
Bot Position: (6, 19), Rat Position: (4, 20), Actions Taken: 110
Bot Position: (6, 20), Rat Position: (5, 20), Actions Taken: 111
Bot Position: (6, 20), Rat Position: (5, 19), Actions Taken: 111
Bot Position: (6, 21), Rat Position: (4, 19), Actions Taken: 112
Bot Position: (6, 21), Rat Position: (3, 19), Actions Taken: 112
Bot Position: (7, 21), Rat Position: (3, 18), Actions Taken: 113
Bot Position: (7, 21), Rat Position: (3, 19), Actions Taken: 113
Bot Position: (7, 22), Rat Position: (3, 20), Actions Taken: 114
Bot Position: (7, 22), Rat Position: (3, 19), Actions Taken: 114
Bot Position: (7, 23), Rat Position: (4, 19), Actions Taken: 115
Bot Position: (7, 23), Rat Position: (4, 20), Actions Taken: 115
Bot Position: (7, 23), Rat Position: (4, 19), Actions Taken: 115
Bot Position: (7, 22), Rat Position: (5, 19), Actions Taken: 116
Bot Position: (7, 22), Rat Position: (4, 19), Actions Taken: 116
Bot Position: (7, 21), Rat Position: (5, 19), Actions Taken: 117
Bot Position: (7, 21), Rat Position: (6, 19), Actions Taken: 117
Bot Position: (7, 20), Rat Position: (6, 20), Actions Taken: 118
Bot Position: (7, 20), Rat Position: (5, 20), Actions Taken: 118
Bot Position: (6, 20), Rat Position: (6, 20), Actions Taken: 119
Rat caught at position (6, 20) after 119 actions!
```

[29]: 119