# Project 2 Report

Mikey Ferguson

March 2025

## 1 Hidden Markov Model

The Hidden-Markov-Model implementation portion of this project entailed implementing the Country Dance and Fixed Lag Smoothing Algorithms, as well as the Viterbi Algorithm to determine a most probable sequence of hidden states given the evidence.
Both smoothing algorithms predicted probabilities of the hidden state at each time step, with the context of a state being whether a student received enough sleep the previous night, and the observations being a combined discrete variable of whether the student has red eyes and whether they sleep in class.
The Hidden Markov Parameters are as follows:

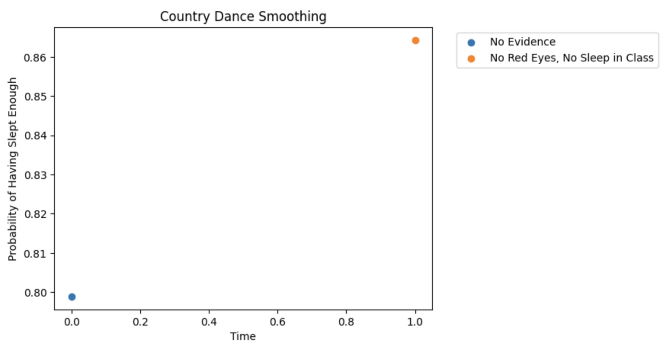
| | Initial Probability |
|---|---|
| Enough Sleep Previous Night | 0.7 |
| Not Enough Sleep Previous Night | 0.3 |

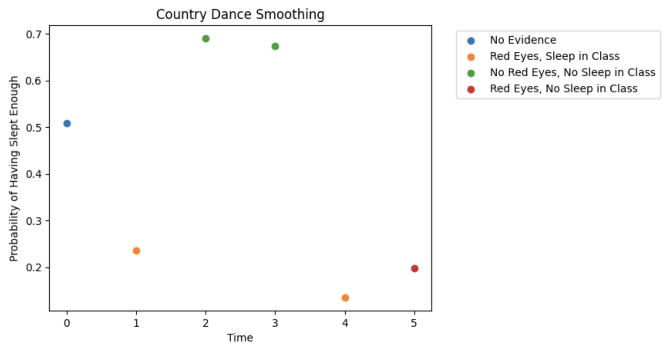
| | Enough Sleep Next Night | Not Enough Sleep Next Night |
|---|---|---|
| Enough Sleep Previous Night | 0.8 | 0.2 |
| Not Enough Sleep Previous Night | 0.3 | 0.7 |

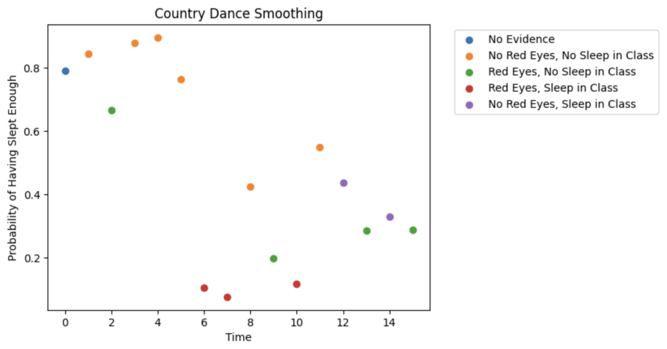
| | Sleeping in Class | Not Sleeping in Class |
|---|---|---|
| Enough Sleep Previous Night | 0.1 | 0.9 |
| Not Enough Sleep Previous Night | 0.3 | 0.7 |

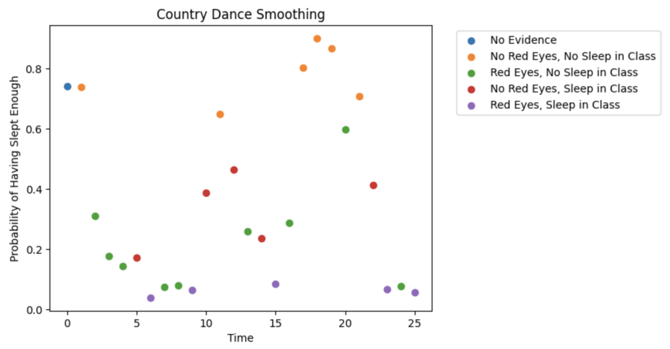
| | Red Eyes | Not Red Eyes |
|---|---|---|
| Enough Sleep Previous Night | 0.2 | 0.8 |
| Not Enough Sleep Previous Night | 0.7 | 0.3 |

The Country Dance Smoothing Algorithm modifies state probabilities based off *all* future evidence. Generating random evidence sequences of varying sizes, I observed the following results (note that each sequence of observations is generated independently – they are not subsequences of each other):

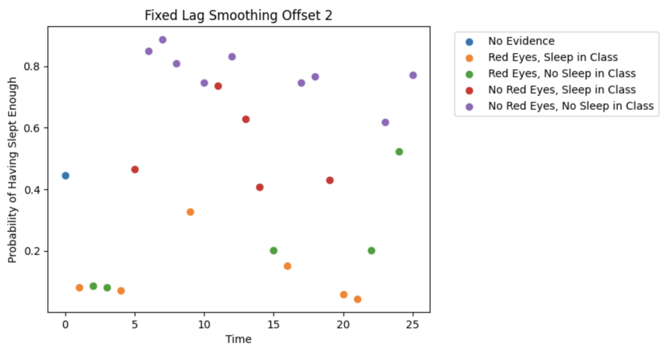


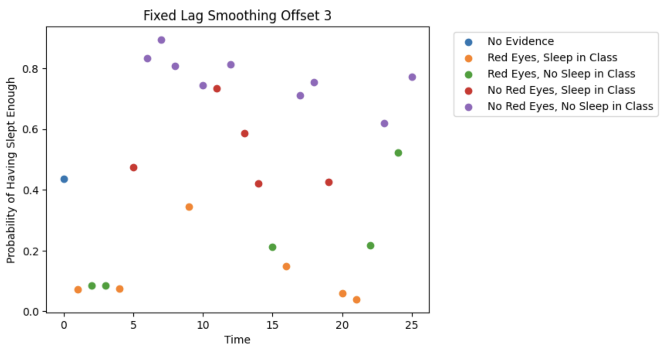


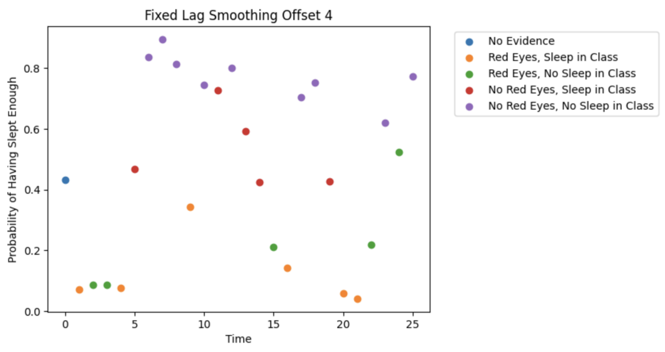


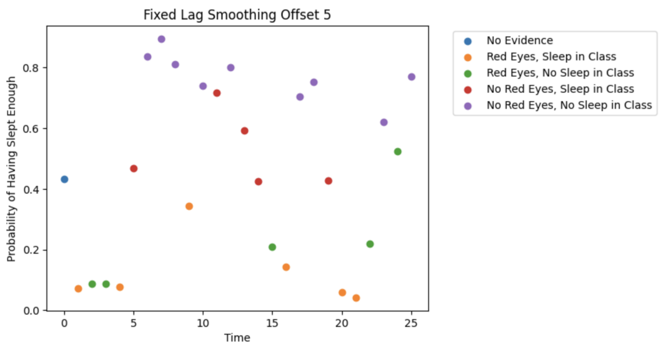


Clearly the initial probability of sleeping well before any observations arrive is not 0.7 in any of the cases, verifying that smoothing has a noticeable effect on the various hidden state probabilities.
The Fixed Lag Smoothing Algorithm only modifies state probabilities based off of the next *offset* future evidence values. Examining an observation sequence of size 25, with various fixed lag offsets, I interestingly saw little difference in each state probability given the various offsets, although the difference *is* non-zero:

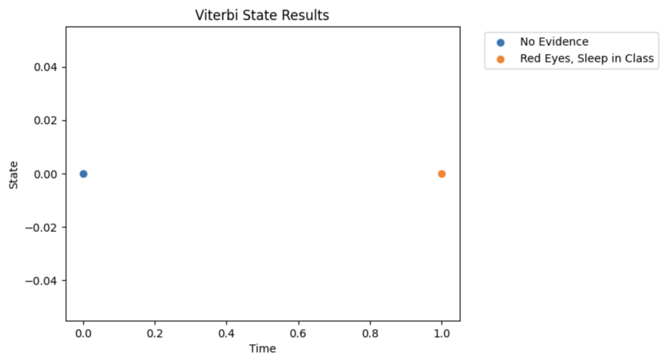


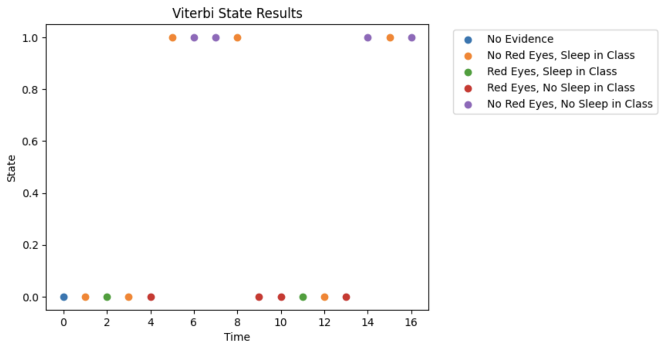


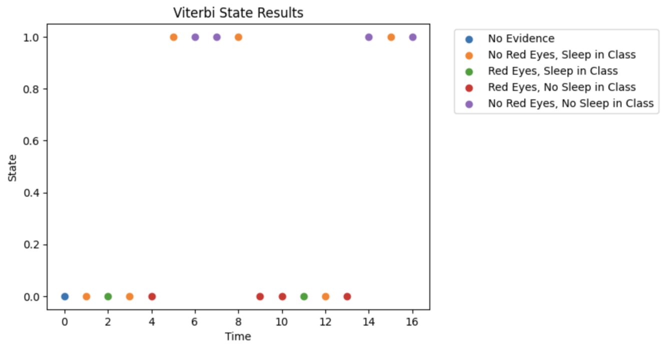


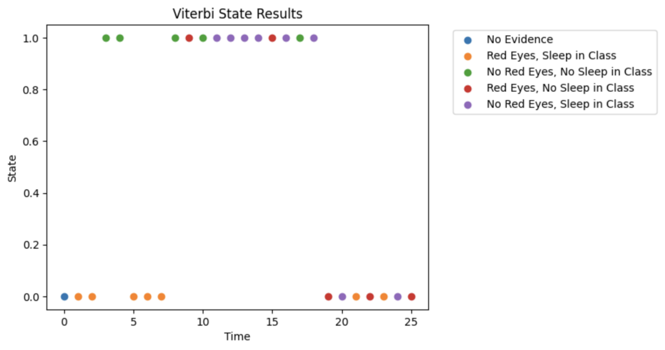


While the difference is slight, one can note that the states at times 1 through 4 (orange, green, green, orange) clearly have slightly different probabilities as the fixed lag offset changes - this is most noticeable when comparing offset 2 with offset 5.
Finally, I implemented the Viterbi Algorithm to predict the most likely sequence of hidden states to explain a sequence of observations. As expected, observations more or less attributed with getting enough sleep had the effect on the most likely sequence:





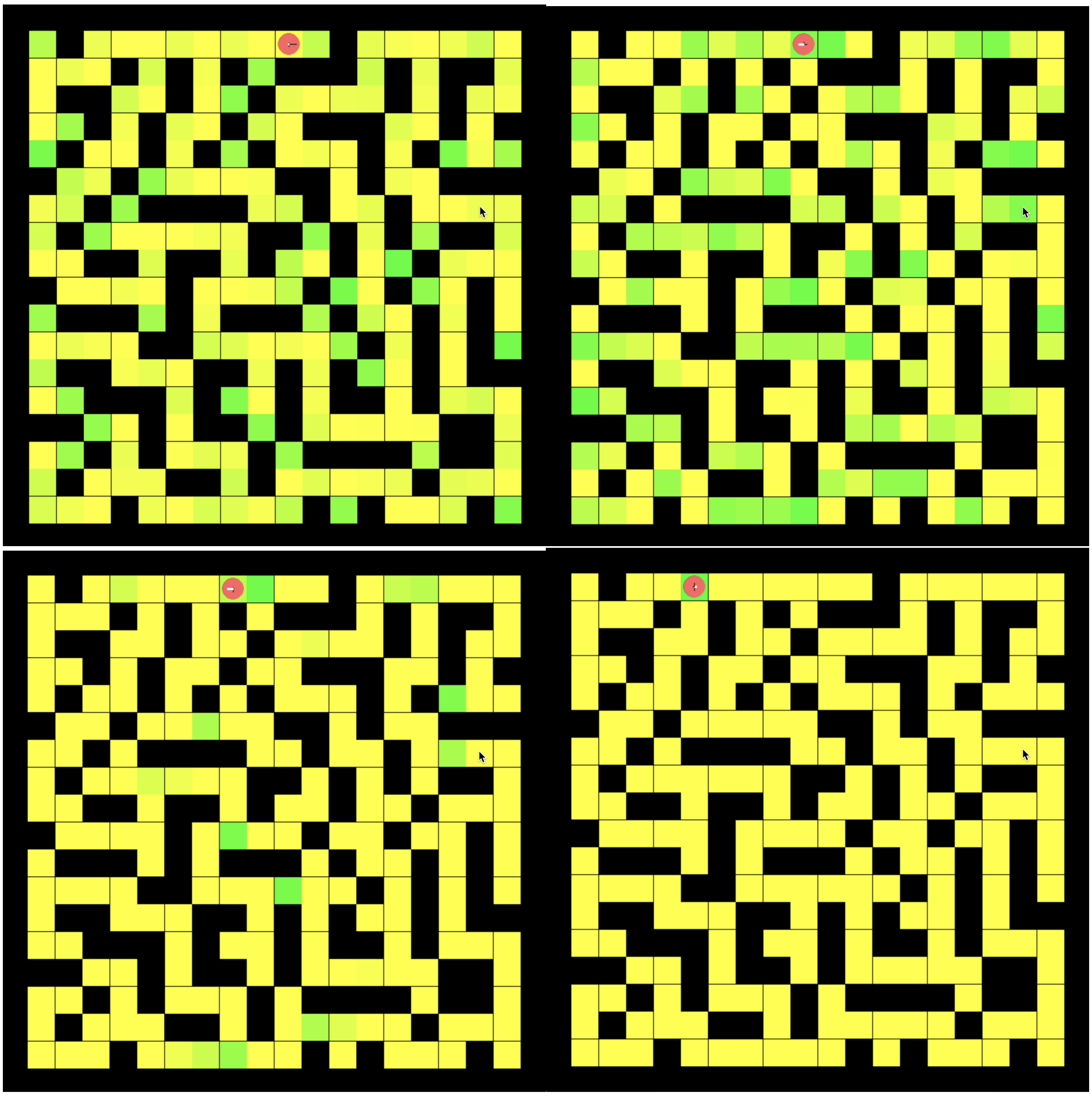




Thus concludes the section of the project that covers the implementation and results of a Hidden Markov Model in the given context.

## 2 Robot Location

Combining multiple discrete observations into a single discrete variable is only practical for a small number of states. In the case of robot location, the number of states numbers in the hundreds, and exponentiating 2 to that power is simply infeasible, which is what would be necessary to combine all such states into one discrete variable.
As such, this project had us implement the Particle Filtering Algorithm to have a blind robot estimate its location in a maze (whose shape it *does* know - it just has no idea where in this maze it is). My implementation of the algorithm runs by throwing 10000 particles randomly into the different (open) cells of the maze. Then, each particle moves according to how the robot moves, and that particle yields a new heading. The weight of the (moved) particle is determined by the probability of the robot generating said new heading from *its* old heading (which that much, at least, it knows). The moved particles are then sampled from according to their weight to create 10000 more particles, and the process repeats. Cells with more particles in them have a higher probability of being the robot's location. Ultimately, this achieved general success in the robot finding its bearings:



As the robot moved by the time step, I recorded the most likely position, as well as the probability the robot would be in said position. I also included the actual robot position.

```
Most Probable Position(s): [(14, 9)]; Probability: 0.0139; Robot Position: (10, 1)
Most Probable Position(s): [(17, 5)]; Probability: 0.0141; Robot Position: (9, 1)
Most Probable Position(s): [(9, 1)]; Probability: 0.0229; Robot Position: (8, 1)
Most Probable Position(s): [(8, 1)]; Probability: 0.0342; Robot Position: (7, 1)
Most Probable Position(s): [(8, 12)]; Probability: 0.0498; Robot Position: (6, 1)
Most Probable Position(s): [(6, 1)]; Probability: 0.0735; Robot Position: (5, 1)
Most Probable Position(s): [(5, 1)]; Probability: 0.1197; Robot Position: (4, 1)
Most Probable Position(s): [(4, 8)]; Probability: 0.0958; Robot Position: (3, 1)
Most Probable Position(s): [(4, 1)]; Probability: 0.1044; Robot Position: (4, 1)
Most Probable Position(s): [(5, 1)]; Probability: 0.1077; Robot Position: (4, 1)
Most Probable Position(s): [(6, 1)]; Probability: 0.1121; Robot Position: (5, 1)
Most Probable Position(s): [(7, 1)]; Probability: 0.1284; Robot Position: (6, 1)
Most Probable Position(s): [(8, 1)]; Probability: 0.1692; Robot Position: (7, 1)
Most Probable Position(s): [(9, 1)]; Probability: 0.2232; Robot Position: (8, 1)
Most Probable Position(s): [(10, 1)]; Probability: 0.2753; Robot Position: (9, 1)
Most Probable Position(s): [(11, 1)]; Probability: 0.3359; Robot Position: (10, 1)
Most Probable Position(s): [(10, 1)]; Probability: 0.4175; Robot Position: (11, 1)
Most Probable Position(s): [(9, 1)]; Probability: 0.4012; Robot Position: (10, 1)
Most Probable Position(s): [(8, 1)]; Probability: 0.3876; Robot Position: (9, 1)
Most Probable Position(s): [(8, 1)]; Probability: 0.3931; Robot Position: (8, 1)
Most Probable Position(s): [(7, 1)]; Probability: 0.399; Robot Position: (7, 1)
Most Probable Position(s): [(6, 1)]; Probability: 0.4109; Robot Position: (6, 1)
Most Probable Position(s): [(5, 1)]; Probability: 0.4052; Robot Position: (6, 1)
Most Probable Position(s): [(4, 1)]; Probability: 0.4772; Robot Position: (5, 1)
Most Probable Position(s): [(4, 1)]; Probability: 0.3951; Robot Position: (4, 1)
Most Probable Position(s): [(3, 1)]; Probability: 0.3913; Robot Position: (3, 1)
Most Probable Position(s): [(3, 2)]; Probability: 0.3909; Robot Position: (3, 2)
Most Probable Position(s): [(3, 1)]; Probability: 0.3114; Robot Position: (3, 1)
Most Probable Position(s): [(4, 1)]; Probability: 0.3403; Robot Position: (4, 1)
Most Probable Position(s): [(5, 1)]; Probability: 0.3319; Robot Position: (5, 1)
Most Probable Position(s): [(6, 1)]; Probability: 0.3311; Robot Position: (6, 1)
Most Probable Position(s): [(7, 1)]; Probability: 0.3113; Robot Position: (7, 1)
Most Probable Position(s): [(8, 1)]; Probability: 0.3101; Robot Position: (8, 1)
Most Probable Position(s): [(9, 1)]; Probability: 0.3515; Robot Position: (9, 1)
Most Probable Position(s): [(10, 1)]; Probability: 0.4092; Robot Position: (10, 1)
Most Probable Position(s): [(11, 1)]; Probability: 0.6135; Robot Position: (11, 1)
Most Probable Position(s): [(10, 1)]; Probability: 0.5928; Robot Position: (10, 1)
Most Probable Position(s): [(9, 1)]; Probability: 0.5598; Robot Position: (9, 1)
Most Probable Position(s): [(8, 1)]; Probability: 0.5272; Robot Position: (8, 1)
Most Probable Position(s): [(7, 1)]; Probability: 0.5001; Robot Position: (7, 1)
Most Probable Position(s): [(6, 1)]; Probability: 0.4847; Robot Position: (6, 1)
Most Probable Position(s): [(5, 1)]; Probability: 0.4649; Robot Position: (5, 1)
Most Probable Position(s): [(4, 1)]; Probability: 0.4927; Robot Position: (4, 1)
Most Probable Position(s): [(3, 1)]; Probability: 0.6813; Robot Position: (3, 1)
Most Probable Position(s): [(4, 1)]; Probability: 0.7049; Robot Position: (4, 1)
Most Probable Position(s): [(5, 1)]; Probability: 0.6691; Robot Position: (5, 1)
Most Probable Position(s): [(6, 1)]; Probability: 0.6481; Robot Position: (6, 1)
Most Probable Position(s): [(7, 1)]; Probability: 0.5946; Robot Position: (7, 1)
Most Probable Position(s): [(8, 1)]; Probability: 0.574; Robot Position: (7, 1)
Most Probable Position(s): [(9, 1)]; Probability: 0.5284; Robot Position: (8, 1)
Most Probable Position(s): [(10, 1)]; Probability: 0.4919; Robot Position: (9, 1)
Most Probable Position(s): [(11, 1)]; Probability: 0.6095; Robot Position: (9, 2)
Most Probable Position(s): [(10, 1)]; Probability: 0.6316; Robot Position: (9, 1)
Most Probable Position(s): [(9, 1)]; Probability: 0.57; Robot Position: (8, 1)
Most Probable Position(s): [(8, 1)]; Probability: 0.5239; Robot Position: (7, 1)
Most Probable Position(s): [(7, 1)]; Probability: 0.4835; Robot Position: (6, 1)
Most Probable Position(s): [(6, 1)]; Probability: 0.4534; Robot Position: (5, 1)
Most Probable Position(s): [(5, 1)]; Probability: 0.4324; Robot Position: (4, 1)
Most Probable Position(s): [(3, 1)]; Probability: 0.383; Robot Position: (3, 1)
Most Probable Position(s): [(4, 1)]; Probability: 0.5221; Robot Position: (4, 1)
Most Probable Position(s): [(5, 1)]; Probability: 0.5552; Robot Position: (5, 1)
Most Probable Position(s): [(6, 1)]; Probability: 0.5771; Robot Position: (6, 1)
Most Probable Position(s): [(7, 1)]; Probability: 0.5674; Robot Position: (7, 1)
Most Probable Position(s): [(8, 1)]; Probability: 0.5643; Robot Position: (8, 1)
Most Probable Position(s): [(9, 1)]; Probability: 0.5439; Robot Position: (9, 1)
Most Probable Position(s): [(10, 1)]; Probability: 0.5069; Robot Position: (10, 1)
Most Probable Position(s): [(11, 1)]; Probability: 0.8481; Robot Position: (11, 1)
Most Probable Position(s): [(10, 1)]; Probability: 0.7932; Robot Position: (10, 1)
Most Probable Position(s): [(9, 1)]; Probability: 0.7251; Robot Position: (9, 1)
Most Probable Position(s): [(8, 1)]; Probability: 0.6684; Robot Position: (8, 1)
Most Probable Position(s): [(7, 1)]; Probability: 0.624; Robot Position: (7, 1)
Most Probable Position(s): [(6, 1)]; Probability: 0.5925; Robot Position: (6, 1)
Most Probable Position(s): [(5, 1)]; Probability: 0.557; Robot Position: (5, 1)
Most Probable Position(s): [(4, 1)]; Probability: 0.5387; Robot Position: (4, 1)
Most Probable Position(s): [(3, 1)]; Probability: 0.5836; Robot Position: (3, 1)
Most Probable Position(s): [(3, 2)]; Probability: 0.5428; Robot Position: (3, 2)
Most Probable Position(s): [(3, 1)]; Probability: 0.4556; Robot Position: (3, 1)
Most Probable Position(s): [(4, 1)]; Probability: 0.5178; Robot Position: (4, 1)
Most Probable Position(s): [(5, 1)]; Probability: 0.501; Robot Position: (5, 1)
Most Probable Position(s): [(6, 1)]; Probability: 0.5705; Robot Position: (5, 2)
Most Probable Position(s): [(8, 1)]; Probability: 0.4308; Robot Position: (5, 1)
Most Probable Position(s): [(5, 2)]; Probability: 0.2196; Robot Position: (5, 2)
Most Probable Position(s): [(5, 3)]; Probability: 0.2604; Robot Position: (5, 2)
Most Probable Position(s): [(9, 1)]; Probability: 0.2653; Robot Position: (5, 3)
Most Probable Position(s): [(8, 1)]; Probability: 0.2455; Robot Position: (4, 3)
Most Probable Position(s): [(4, 4)]; Probability: 0.1653; Robot Position: (4, 4)
Most Probable Position(s): [(4, 5)]; Probability: 0.172; Robot Position: (4, 5)
Most Probable Position(s): [(4, 4)]; Probability: 0.2289; Robot Position: (4, 4)
Most Probable Position(s): [(4, 3)]; Probability: 0.2958; Robot Position: (4, 3)
Most Probable Position(s): [(4, 4)]; Probability: 0.3477; Robot Position: (4, 4)
Most Probable Position(s): [(4, 5)]; Probability: 0.3725; Robot Position: (4, 5)
Most Probable Position(s): [(4, 4)]; Probability: 0.4052; Robot Position: (4, 4)
Most Probable Position(s): [(4, 3)]; Probability: 0.4006; Robot Position: (4, 3)
```

```
    Most Probable Position(s): [(5, 3)]; Probability: 0.4419; Robot Position: (5, 3)
    Most Probable Position(s): [(4, 3)]; Probability: 0.462; Robot Position: (4, 3)
    Most Probable Position(s): [(5, 3)]; Probability: 0.427; Robot Position: (5, 3)
    Most Probable Position(s): [(5, 2)]; Probability: 0.5046; Robot Position: (5, 2)
    Most Probable Position(s): [(5, 1)]; Probability: 0.2857; Robot Position: (5, 2)
    Most Probable Position(s): [(6, 1)]; Probability: 0.2092; Robot Position: (5, 1)
    Most Probable Position(s): [(6, 1)]; Probability: 0.1874; Robot Position: (5, 2)
    Most Probable Position(s): [(5, 2)]; Probability: 0.2816; Robot Position: (5, 2)
    Most Probable Position(s): [(5, 1)]; Probability: 0.2628; Robot Position: (5, 1)
    Most Probable Position(s): [(5, 2)]; Probability: 0.3586; Robot Position: (5, 2)
    Most Probable Position(s): [(10, 1)]; Probability: 0.337; Robot Position: (5, 3)
```

Interestingly, the robot was able to narrow its location down to nearby cells, and eventually track down the exact correct cell. It often stayed correct once it found its right cell, but notably it could lose its way again as seen above. However, encouragingly, with each loss, it eventually found its location again and remained correct for multiple time steps. I am confident that despite the last slipup seen in the report above, the robot would have found itself again had I allowed the simulation more runtime.