***Project Report - Nikhith Nookala***

**Data and Analysis:**

**1) Meet with the TA to discuss your ideas.**

**2) Suppose the ship had four rooms, A, B, C, D, and the probability the mouse was in each of them was 0.4, 0.3, 0.2, 0.1. If you looked in room B, and the mouse was not there, what is the probability it is in room A?**

Room A: P(A) = 0.4

Room B: P(B) = 0.3

Room C: P(C) = 0.2

Room D: P(D) = 0.1

*P(A | !B) = ?*

Bayes Theorem: A and B are independent events so:

P(A | !B) = [P(A ∩ !B) / P(!B)

Checked Room B, no mouse: P(!B) = 1 - P(B) = 1 - 0.3 = 0.7

P(A ∩ !B) = 0.4

* (!B and A are independent events, so if the mouse is in room A, it is definitely not in room B)

[P(A ∩ !B) / P(!B)] = 0.4 / 0.7 = 0.5714 ≈ 0.6

**3) Given a probabilistic knowledge base of where the mouse is, how should you update your probabilistic knowledge base if the bot sensor gets a beep? Be precise and mathematical.**

* Let us denote that the probability that the mouse is at some position X is P(X). The probability of there being a beep can be denoted as P(Beep). If the bot sensor gets a beep, we are then solving for P(X | Beep). Using Bayes theorem, our probabilistic knowledge base then becomes [P(Beep | X) • P(X)] / P(Beep).

**4) Given a probabilistic knowledge base of where the mouse is, how should you update your probabilistic knowledge base if the bot sensor fails to beep? Be precise and mathematical.**

* Let us denote that the probability that the mouse is at some position X is P(X). The probability of there not being a beep can be denoted as P(!Beep). If the bot sensor gets a beep, we are then solving for P(X | !Beep). Using Bayes theorem, our probabilistic knowledge base then becomes [P(!Beep | X) • P(X)] / P(!Beep).

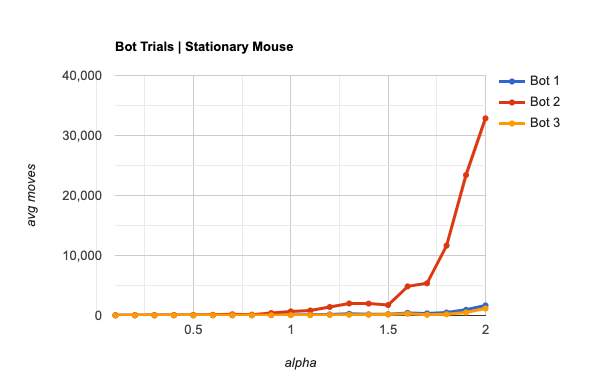
**5) Given a probabilistic knowledge base of where the mouse is, how should you update your probabilistic knowledge base if you know the mouse might have moved? Be precise and mathematical.**

* Let us denote that the probability that the mouse is at some position X is P(X). The probability of the mouse moving can be denoted P(Moved). This means that the probability the mouse did not move is P(!Moved). Ideally, if the mouse doesn't move, we are then solving for P(X | !Moved). Using Bayes theorem, our probabilistic knowledge base then becomes [P(!Moved | X) • P(X)] / P(!Moved).

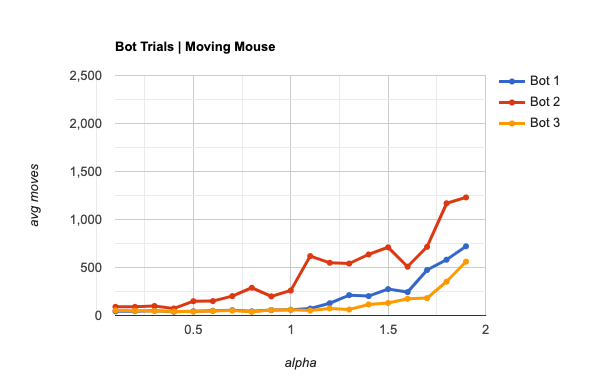
**6) How did you design your Bot 3? Be precise and thorough as to your choices.**

* I designed my Bot 3 with my theory that keeping a history of some x recent beeps might help the bot better decide when to move versus when to sense. I believed that by keeping a memory of some number of past beeps, the bot would be able to dynamically decide whether or not to move or sense each timestep, rather than a hard coded alternative that we find in bots 1 and 2.
* In my design, Bot 3 kept track of the last 5 sensing results which would allow it to calculate the likelihood that the bot was nearby.
* Strategy: Prioritize moving if the ratio between successful remembered beeps and total remembered beeps is less than 0.5 but greater than 0, which most likely means the bot is very far from the mouse.
  + Exception 1: if there have been no beeps, which means the bot still needs direction and therefore knows to sense again
  + Exception 2: If the bot gets within 25 units (manhattan distance) from its last successful beep’s location, it reverts back to Bot1’s strategy of sensing, moving all the way to the target, and then sensing again.

**7) Evaluate and compare Bots 1, 2, 3 in the case of a stationary mouse: plot their average moves to catch the mouse as a function of α. (How can you find a good range of α values?)**



**8) Evaluate and compare Bots 1, 2, 3 in the case of a moving mouse: plot their average moves to catch the mouse as a function of α. (How can you find a good range of α values? Does it change?)**



**9) How does the math change if there are two stationary mice? Two moving mice?**

Two Stationary Mice:

* The math changes because you have to take into account the beeps from two mice instead of one. It would change in the way that your probabilities would have to be weighted towards the closest mouse. This can be accomplished by using the product of the individual probabilities when sensing, or by sensing both mice and choosing the closest mouse to capture first, i.e the higher probability cells.

Two Moving Mice:

* The math changes here because you have to not only take into account two mice, but also their independent movements. The math itself would have to incorporate probabilities for both mice again in a similar or same fashion as it would be for two stationary mice, either using the product of both individual probabilities or choosing the closest mouse first. However, the model would have to account for the probability that the mice move as well, but this is less impactful on the math and more impactful on the algorithm/strategy that the bot is using.

**10) Evaluate and compare Bots 1, 2, 3 in the case of two moving mice: plot their average moves to catch the mice as a function of α. (You may assume that after the first mouse is caught, it is removed from the board - reducing the situation to the previous situation.)**

