PROJECT 2 REPORT

1) For the hard sensor bot, how are you representing the knowledge base? How are you updating the knowledge base based on a positive signal or a negative signal?

- A collection of possible cells (self.possible_cells) where the rat might be serves as the hard sensor bot's knowledge base. We start wit all the open cells on the ship. The signal that the bot receives each time it senses is either positive (the rat is within k Manhattan distance) or negative (the rat is farther than k). If the signal is positive, the bot only keeps cells within a =range of k and if it is negative, it removes those cells and only keeps cells outside of that range. This gradually reduces the number of legitimate rat locations.
- We set our k value to equal 6 after numerous trials, specifically after testing with values between 1 and 20.

2) For the soft sensor bot, how are you representing the probabilistic knowledge base? How are you updating the knowledge base based on a beep or no beep? Give the update formula you're using. Please note: there is a correct formula for how to update the probabilities!

- Based on the Soft Sensor The bot's probabilistic knowledge base is shown as a 2D belief grid, where each cell holds the probability that the space rat is in it. When a beep is hard by the bot, it will use Bayes' rule to revise the belief. $P'(i,j) = \frac{P(i,j)*e^{-\alpha(d-1)}}{z}$, else if there is no beep heard, it will use $P'(i,j) = \frac{P(i,j)*(1-e^{-\alpha(d-1)})}{Z}$

3) Explain the design and algorithm for your improved Hard Sensor Strategy, being as specific as possible as to what your bot is actually doing.

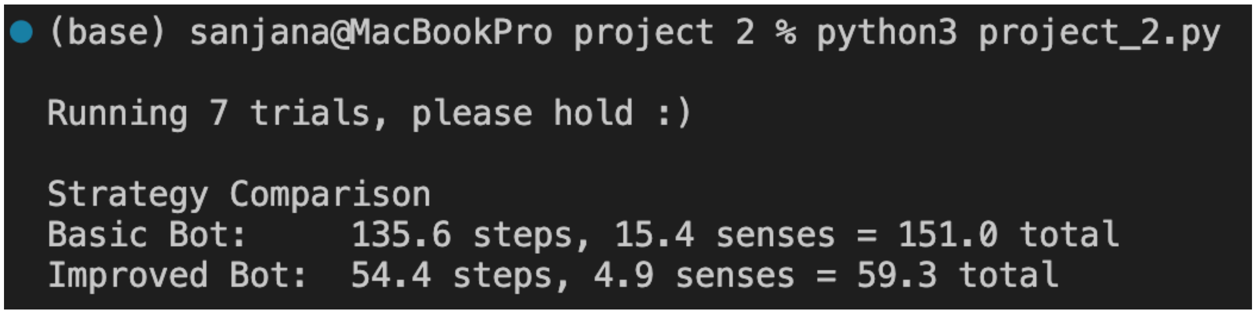
- The improved hard sensor bot selects its next sensing location using an the following strategy:
- At each step, it assesses every potential rat location that hasn't been sensed yet at each stage and calculates how much information it could learn by sensing there.
- Specifically, it estimates how much information can be learned by multiplying the number of cells that, depending on a potential signal, would be ruled in (within distance k) or ruled out (outside distance k).
- It also favors closer targets by dividing the gain by the distance to the target.
- After that, the bot navigates to the cell with the greatest anticipated information gain, senses it, and modifies its knowledge base appropriately.
- This continues until the bot finds the rat.

4) Explain the design and algorithm for your improved Soft Sensor Strategy, being as specific as possible as to what your bot is actually doing.

- According to the enhanced Soft Sensor technique, the bot maintains a belief grid, a 2D array with the likelihood that the space rat is present in each cell. Bots perceive faint beeps with each step. Perceiving a beep from a cell is based on its Manhattan distance from the bot, using the formula: P(beep∣d)=e−α(d−1), where α regulates the signal decay rate with distance. The bot changes its belief grid using Bayes' rule. It increases the odds of neighboring cells proportionately to their The system increases the chances of generating a beep when one is detected and decreases them when none is found. To normalize probabilities, multiply each cell's probability by $e^{-\alpha(d-1)}$ for a beep and by $1 - e^{-\alpha(d-1)}$ for no beep. After updating the belief, the bot goes toward the cell with the highest probability, making smart decisions and zooming in on the rat's position. This probabilistic technique simulates uncertainty instead of utilizing a binary threshold, improving performance.

5) As in the previous project, I want to do comparisons between the strategies by comparing statistics. You'll run multiple experiments to get data to perform the comparison.
– For the hard sensor strategies, create three plots. As a function of k (the sensor radius), plot a) the average number of moves it took each strategy to find the space rat, b) the average number of sense actions it took each strategy to find the space rat, and c) the average number of total actions (sense + move) it took each strategy to find the space rat

- At k = 6, here is the comparison between strategies:



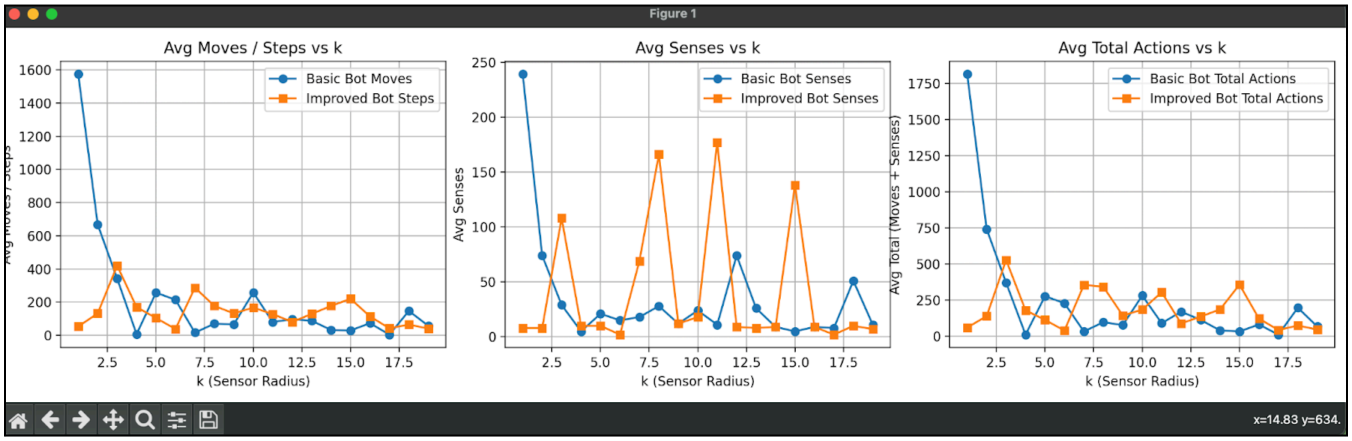
```
(base) sanjana@MacBookPro project 2 % python3 project_2.py

Running 7 trials, please hold :)

Strategy Comparison
Basic Bot:     135.6 steps, 15.4 senses = 151.0 total
Improved Bot:  54.4 steps, 4.9 senses = 59.3 total
```

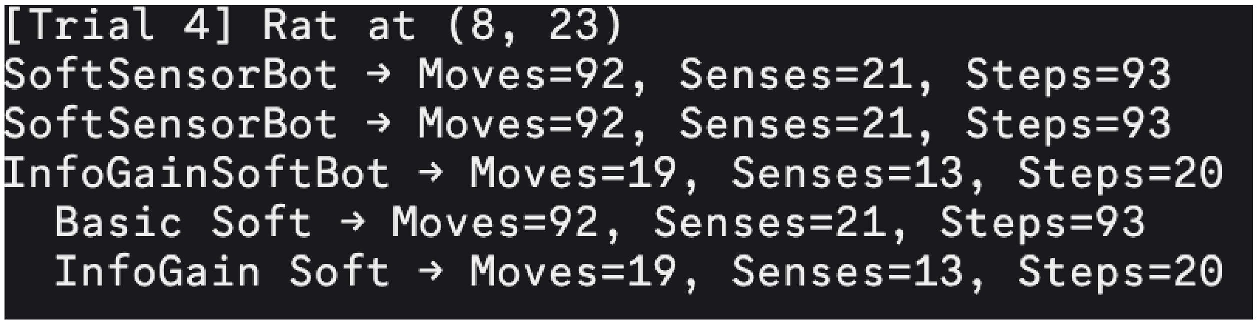
- Here are the plots for for the function of k (Range: 1-20):

-



- **Analysis:** Both bots performances differed greatly depending on the sensor radius value. Extreme peaks (such as more than 1800 actions at k = 1) and irregular dips were among the Basic bot's high action fluctuation. The improved bot, on the other hand, showed us considerably more consistent behavior, consistently exhibiting lower or comparable total actions across a wide range of k values. The improved bot maintained moderate sensing counts while keeping its step counts efficient across most values. The total action curve for the improved bot was smoother and more controlled than that of the basic bot, indicating better adaptability and decision making. Overall, this demonstrates the improved strategy's dependability across a range of sensor ranges. Of those, k = 6 stood out as one of the values where the improved bot achieved both low steps and low senses, demonstrating a successful balance between movement and information gathering.

– For the soft sensor strategies, create three plots. As a function of α (the sensor sensitivity), plot a) the average number of moves it took each strategy to find the space rat, b) the average number of sense actions it took each strategy to find the space rat, and c) the average number of total actions (sense + move) it took each strategy to find the space rat. Note, I recommend a range of α between 0 and 0.5 maximum.

- In trial 4, for example, the soft sensor parameter when set on 0.5 alpha, the InfoGainSoftBot excelled from the baseline than SoftSensorBot because of the use of probabilistic reasoning and information theoretic planning.Out of the 93 things that SoftSensorBot had to do for which it took 92 moves and 21 sensing actions. While the InfoGainSoftBot did it much better and took 19 steps and 13 sensing actions, for a total of 20steps. Using entropy based sensing and Bayesian belief updates, the InfoGainSoftBot is superior, it chooses action that will provide it with most information based on the sensor model P(beep | *k)* = $e^{-a(k-1)}$. *It* is much easier to find your way around and find targets in places where you do not know what is going on by using probabilistic model with active sensing is what the results show.
- 

```
[Trial 4] Rat at (8, 23)
SoftSensorBot → Moves=92, Senses=21, Steps=93
SoftSensorBot → Moves=92, Senses=21, Steps=93
InfoGainSoftBot → Moves=19, Senses=13, Steps=20
  Basic Soft → Moves=92, Senses=21, Steps=93
  InfoGain Soft → Moves=19, Senses=13, Steps=20
```

- Plot from a different range of trials (7 trials):

- Analysis: In conclusion, the InfoGain Bot consistently utilized a lower total number of actions (and fewer steps than the basic bot, achieving the maximum number of sensing actions in nearly all of the 7 trials conducted. This indicates that the InfoGain Bot relies significantly on advanced sensing techniques (driven by information gain and probabilistic reasoning), enhancing its efficiency and minimizing unnecessary movements. The Basic Bot exhibited variable performance with less effective movement and sensing patterns, resulting in higher total action counts in the majority of trials.

6) Speculate on the optimal strategy for both sensing modalities.

- Hard Sensor: Combining an adaptive movement efficiency with an information gain strategy like the one I implemented would probably be the best approach for the hard sensing modality. To maximize info gain, cells that divide the remaining potential rat locations approximately in half should be given priority by the bot early in the search. The bot could switch to greedy targeting as the search becomes more focused and there are fewer options left, moving straight to positions that are most likely to be occupied by rats in order to reduce movement.

- Soft Sensor: Make a probability belief map that changes with each random data point. This is a good way to use soft sensing. The bot would be able to find places where the rat is most likely to be. To get the most information at first, the bot should pick places where there is a lot of doubt. Then, as the chance distribution gets more precise, it should switch to a more flexible approach, moving toward places with the highest trust to cut down on useless movement.

Acknowledgments:

- **Sanjana Suresh worked on the hard sensor bot implementations, code integration, and the report.**
- **Krish Patel worked on the soft sensor bot implementations, and the report.**