

**Originality Statement:** I confirm that all work submitted as part of this document is my own. I used sources such as GeeksForGeeks, Python documentation, and ChatGPT for things like zip(), sort(), plotting and other Python nitpicky things

### **Parameters Used**

Learning rate: 0.1

Discount factor: 0.9

Epsilon: 0.1

Episodes: 500

Time steps: 20

### **1 - Implement SARSA to solve this problem. How did the algorithms perform? Include learning curves and plots of the learned value tables.**

Plots are included at the end of the writeup. SARSA appears to be much slower and takes a lot of missteps to reach the final solution. This is probably because it uses the q value and action of the next state instead of doing a greedy selection like Q-learning does. The heatmap shows this as well where it's not clear where the path it took is.

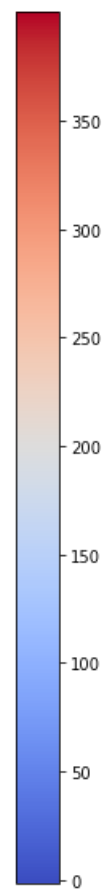
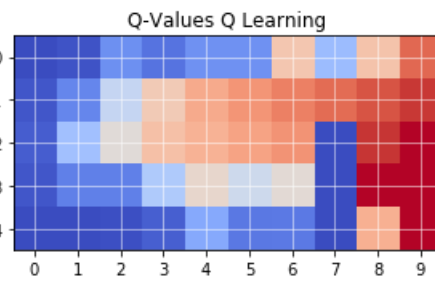
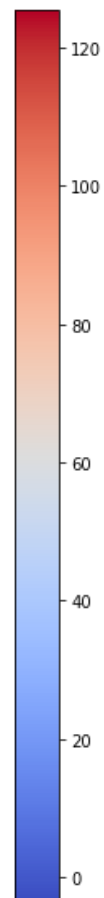
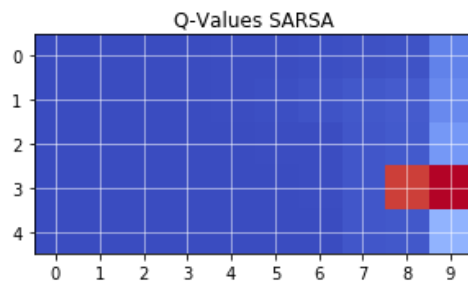
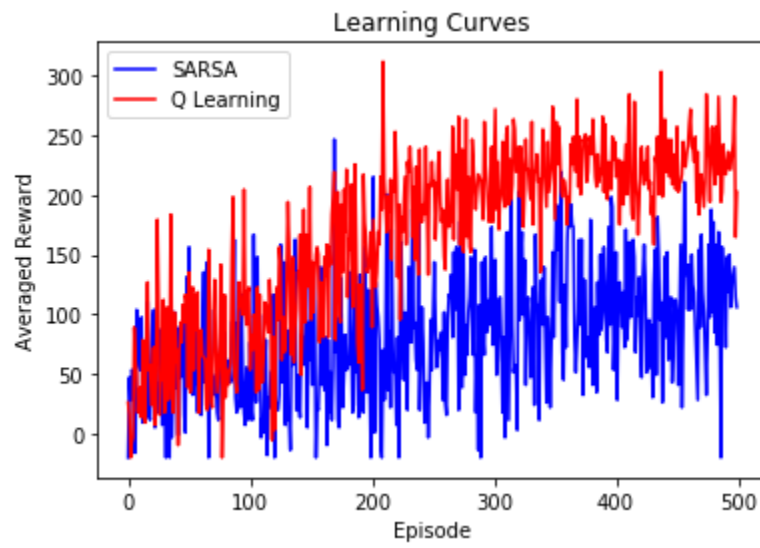
### **2 - Implement a Q-learning algorithm to solve this problem. How did the algorithms perform? How did solution compare to the SARSA solution? Discuss the implications of your results.**

Q-learning outperformed the SARSA solution. Q-Learning appears to learn the policy much faster and as a result converges to a solution much faster than SARSA does. This is probably because of the greedy action selection which picks the best q value to follow. This is also visible in the heatmap where you can clearly see the path taken by the robot.

### **3 - Now consider the environment where the red door moves randomly by 1 cell every time step. Keep the initial starting location of the door the same as before. Use the EXACT same algorithms from problems 1 and 2 to solve this problem. How does the performance of the agent compare to problems 1 and 2? Does the agent learn a good policy? Describe your results and hypothesize why your agent performs the way it does. Speculate on how you may improve the performance of the agent. Again, plot learning curves and value tables.**

Both the SARSA and Q-Learning algorithms performed significantly worse when the goal moves. This is possibly because each episode assumes that the goal is stationary so it tries going the same way. SARSA doesn't appear to learn a good policy with its averaged reward being negative more often than not. Q-learning does learn a good policy. Q-learning seems to outperform slightly and this is possibly because its greedy nature is more likely to try random actions. You can see this in the heat map where Q-learning seems to move more erratically but SARSA takes the same-ish path every time.

## Learning Curves and Heatmaps for STATIONARY Goal



## Learning Curves and Heatmaps for RANDOMLY MOVING Goal

