Jason Teng
Programming Assignment 4 Writeup

1. Q-learning Analysis

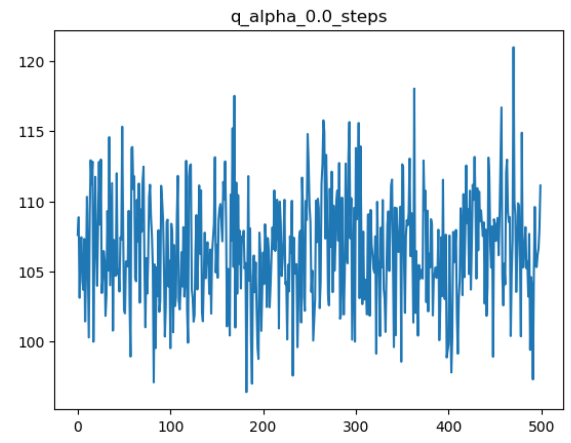
1. Implementation

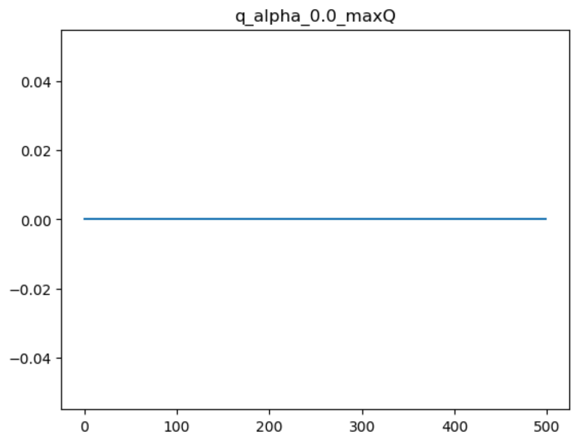
Overall, I found the implementation of Q-learning to be fairly straightforward, more or less directly implementing the pseudocode from the lecture slides (lecture 20, slide 76). My program works fully as intended, although full runs with 500 experiments and 500 episodes each take around 5 minutes to complete on my system. A multi-process implementation would likely drastically reduce the time required for each run.

I modified the Gridworld class implementation so that rather than receiving the reward of +5.0 after making an action from the terminal state, the actor would receive the +5.0 reward upon reaching the terminal state.

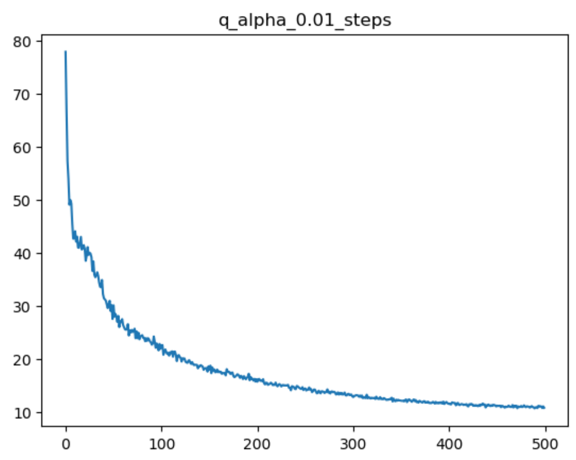
2. Analysis

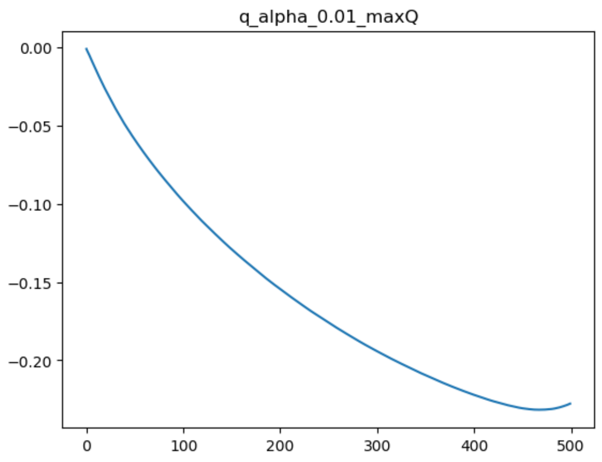
Below are the graphs using Q-learning with various values for α (all other values at default):



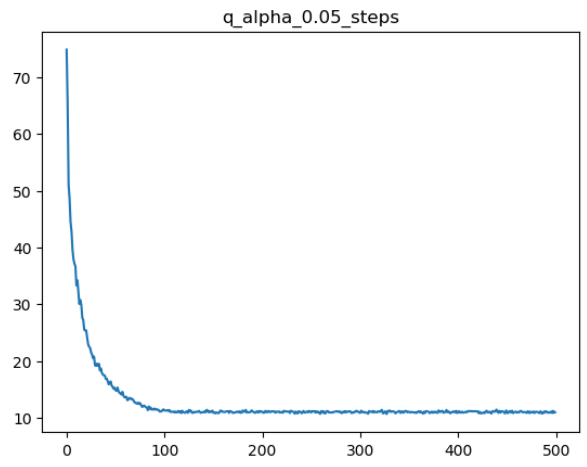


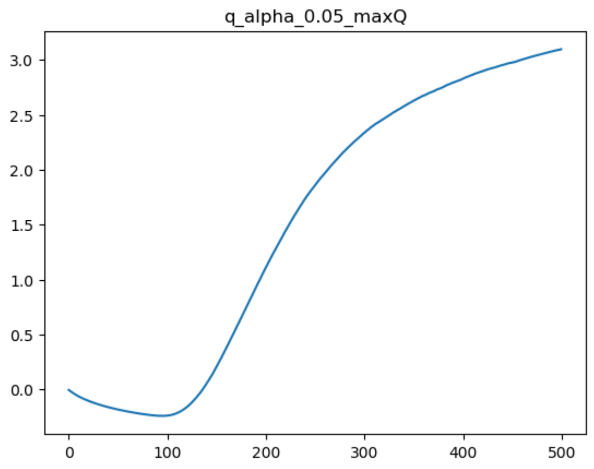
With α=0, we see that there is no modification of the Q-values from their starting values of 0, and the average number of steps required to reach the terminal state does not converge at all. This is to be expected, as a learning rate of 0 corresponds to no learning of the optimal policy.



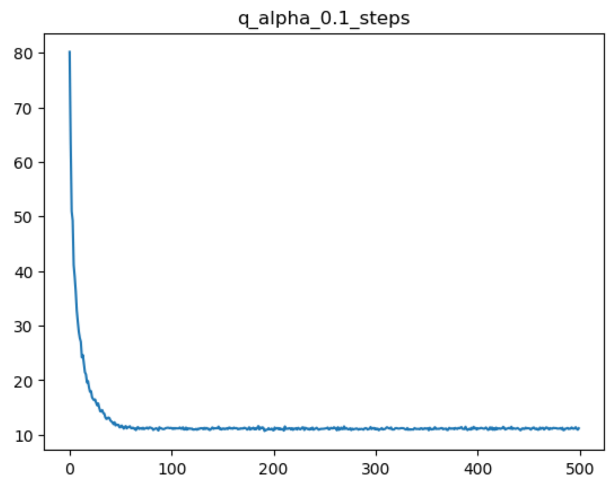


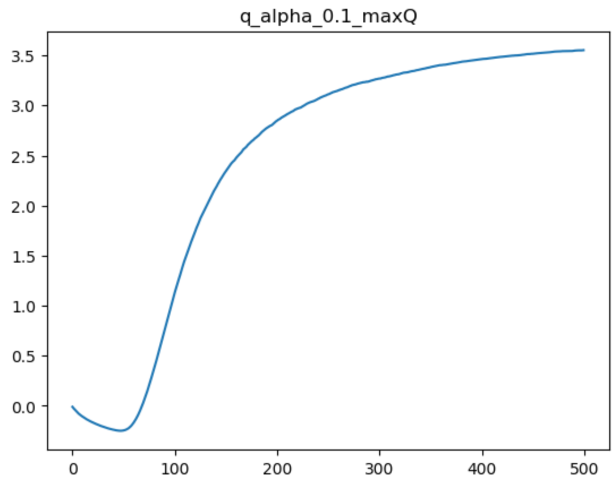
With α=0.01, there is immediately an improvement to the average number of steps to reach the terminal state over time. The max-Q values, however, paradoxically decrease, although they do not appear to converge within 500 episodes. The graph shows a clear uptick in the line, indicating that the max-Q values begin increasing past 500 episodes with α=0.01, and might converge to a larger value with more episodes or a larger α.



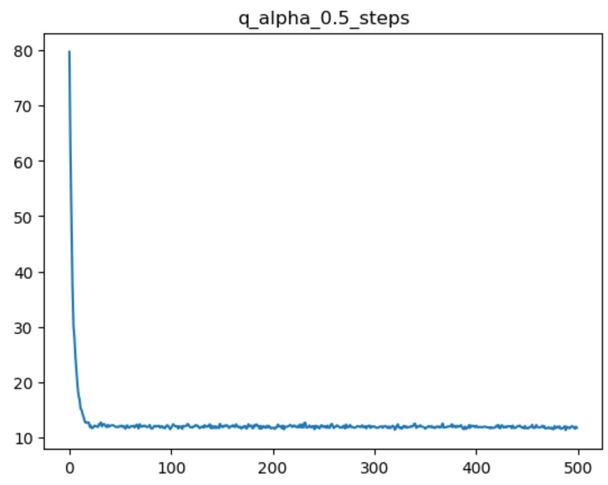


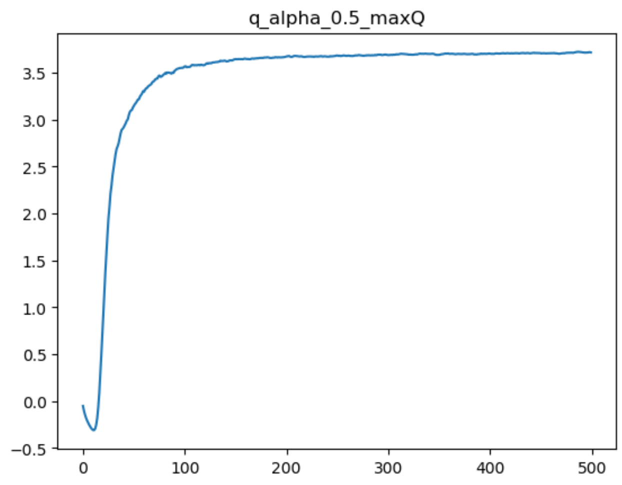
At α=0.05, the average number of steps drops faster than before, and shows clearly a convergence at approximately 10 steps. The max-Q value graph confirms the suspicions of a positive turn from the previous set of experiments, but still do not appear to have converged to a single Q-value.



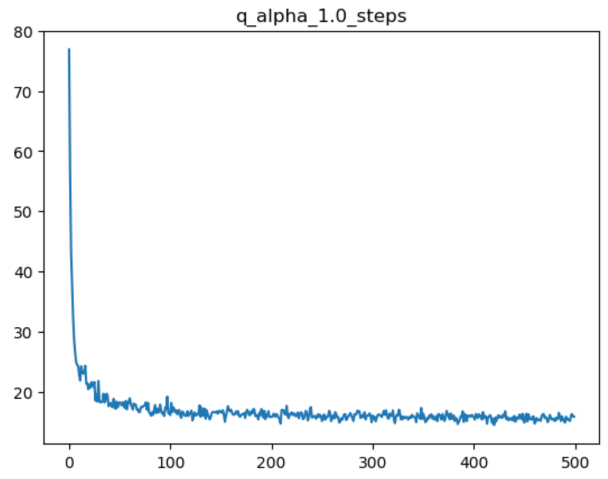


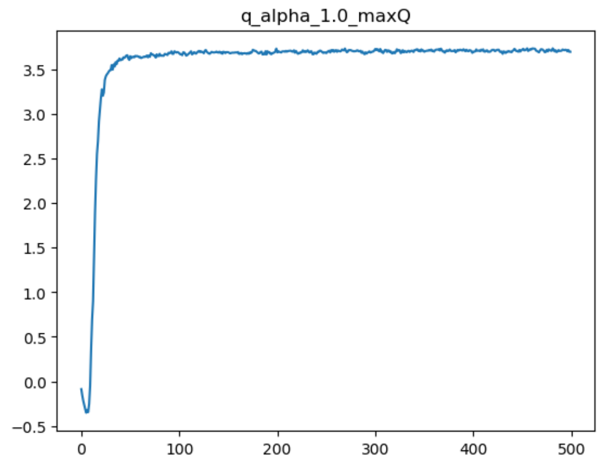
With α=0.1, the number of steps converges slightly faster than before, as well as the max-Q values, although it is still impossible to determine precisely where the Q-values are converging.





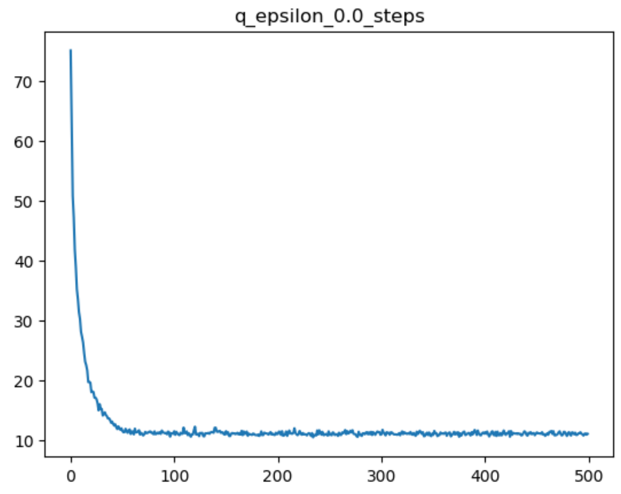
At α=0.5, the number of steps converges extremely quickly, and the max-Q values finally appear to show convergence as well, at a value of roughly 3.6.

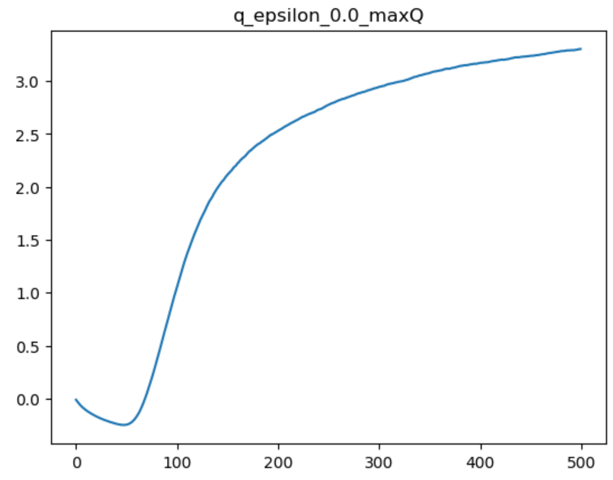




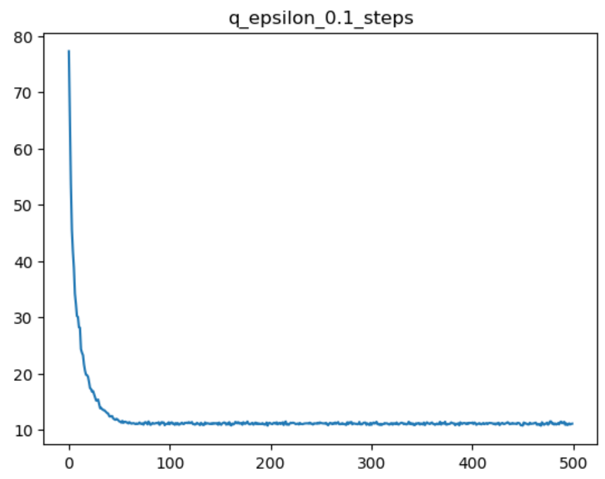
At α=1.0, we begin to see the effects of stochasticity, where although each plot roughly converges to the same values as before, there is significant noise in both graphs due to the stochastic nature of the ε-greedy policy. This would indicate that setting α=1.0 would in general be a poor choice in running Q-learning.

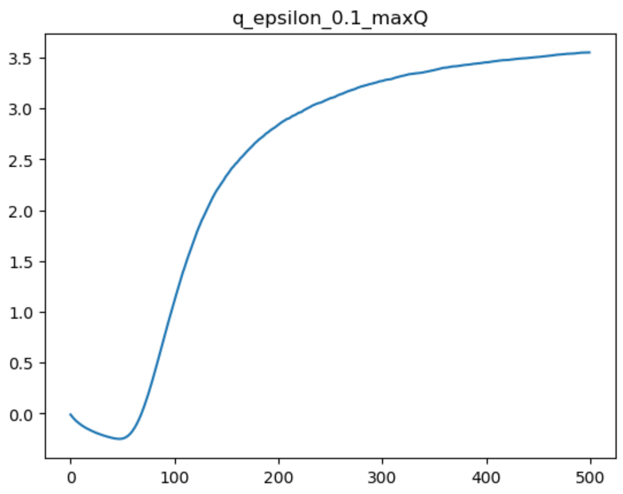
Below are the graphs obtained using Q-learning with various values of ε:



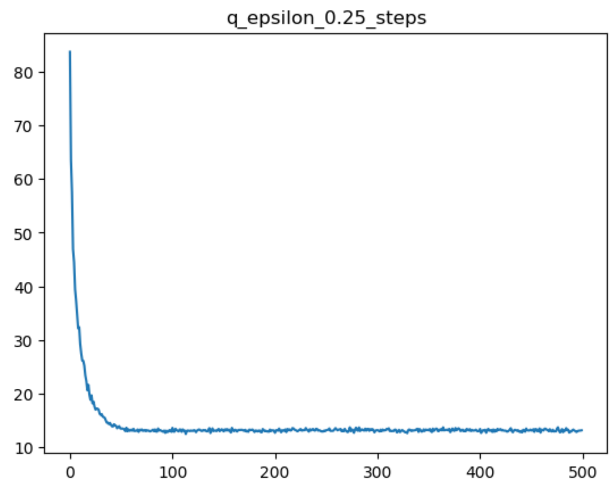


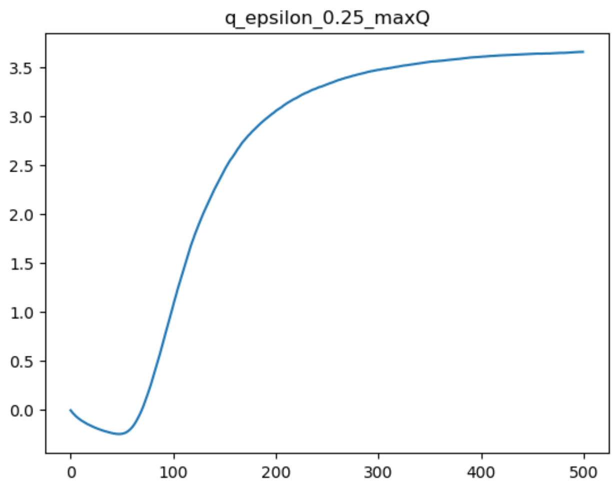
With ε=0, the average number of steps converges relatively quickly, but the same cannot be said for the max-Q value, which fails to converge to the previously achieved value of 3.6 within 500 episodes. I believe this is due to the fact that with ε=0, no “exploratory” actions will ever be taken; therefore suboptimal “local” routes that happen to reach the terminal state will not be deviated from, except for the 0.2 random movement for each action.



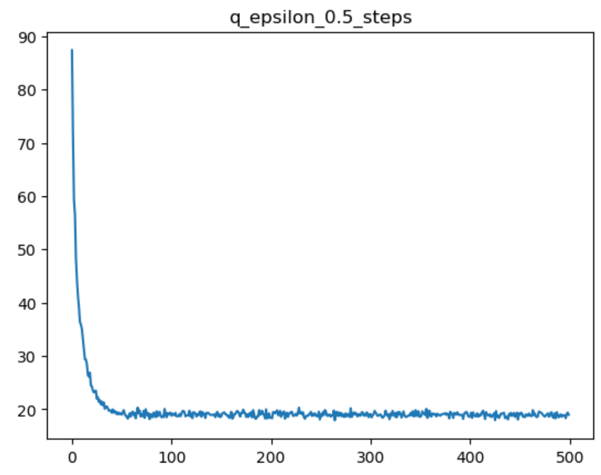


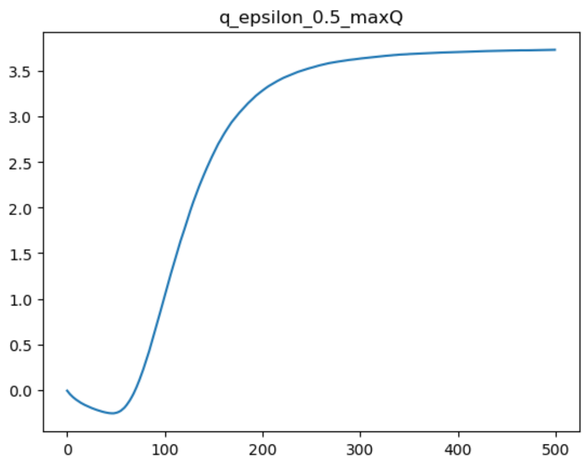
Increasing ε to 0.1 has no noticeable impact on the speed at which the average number of steps converge, but does improve the learning speed for the max-Q values, although it still does not quite fully reach the expected value of 3.6.



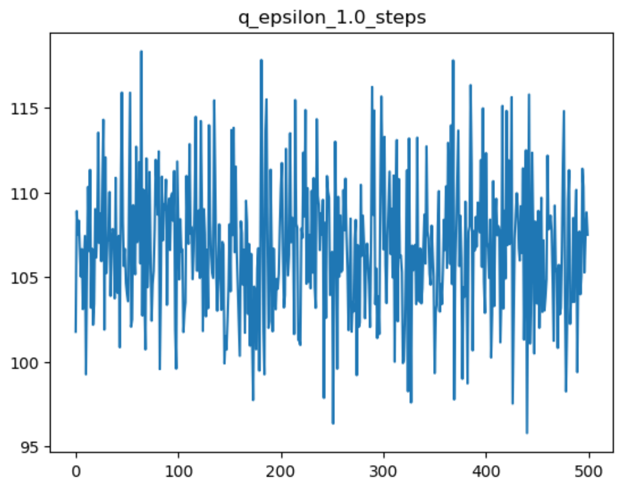


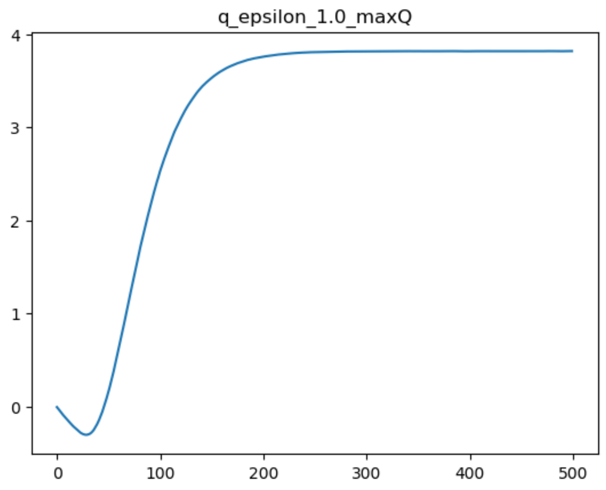
At ε=0.25, the max-Q values finally begin to converge towards 3.6.





At ε=0.5, there appears to be slightly more noise in the average number of steps, although the max-Q values also converge slightly faster.





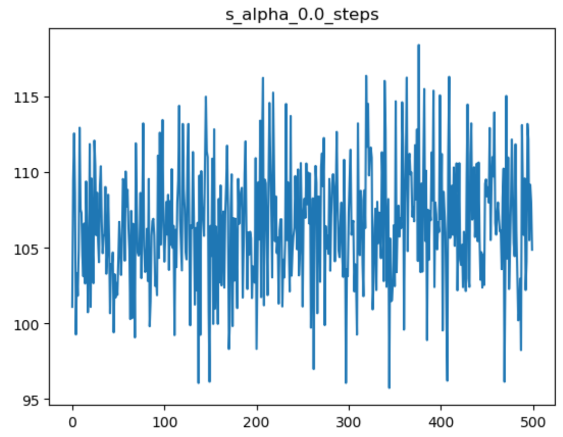
Finally, at ε=1.0, the average number of steps no longer converges, and remains completely random, just like with α=0, but this time the max-Q values converge to an even higher value than seen in previous plots, reaching approximately 3.8. I believe the randomness of the steps is due to the fact that with ε=1, every action taken is completely random, but learning/reward information is still obtained, and therefore Q-learning is still able to quickly converge to optimal values.

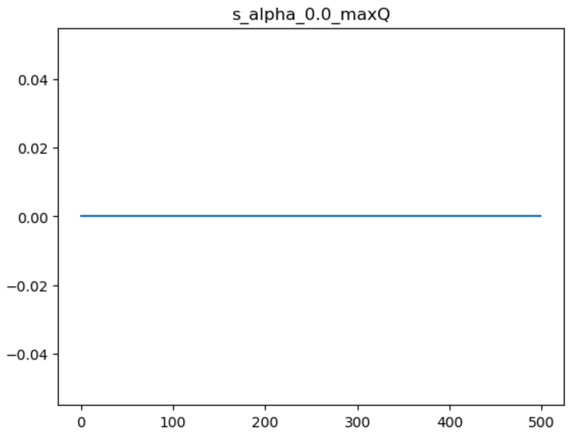
2. Sarsa(λ):

1. For Sarsa(λ), I implemented the algorithm described in http://incompleteideas.net/book/ebook/node77.html, which was given by Professor Amato in Piazza post #244. One potential issue I found is with the runtime of my algorithm; due to the additional nested loop in my Sarsa(λ), each run of 500 experiments of 500 episodes takes almost 20 minutes.

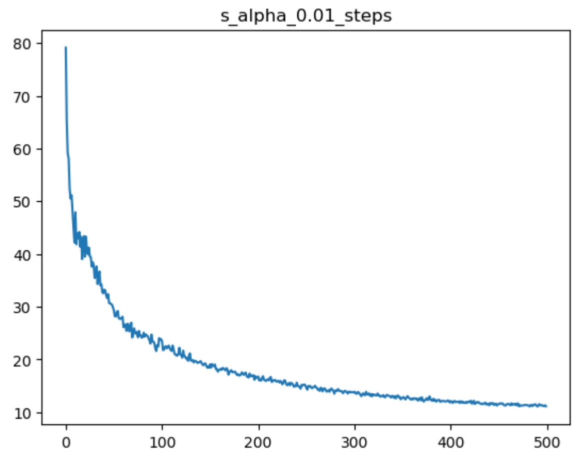
2. Plot Analysis

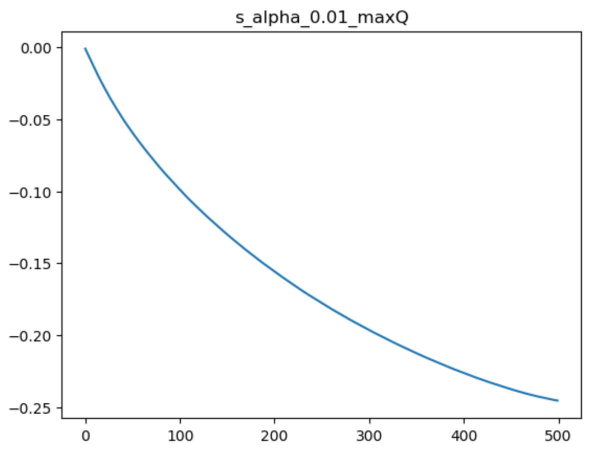
Below are graphs using Sarsa(λ) with various values of α:



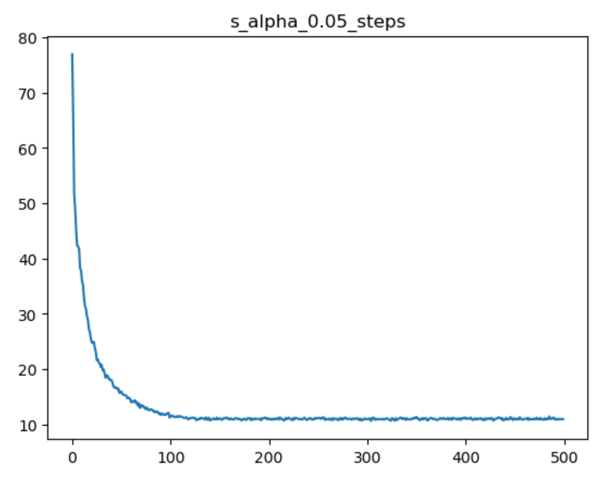


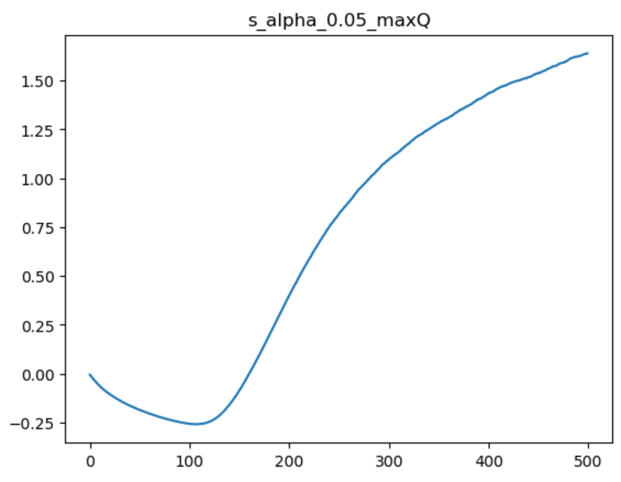
With α=0, the results are practically identical to those from Q-learning, for much the same reasons.



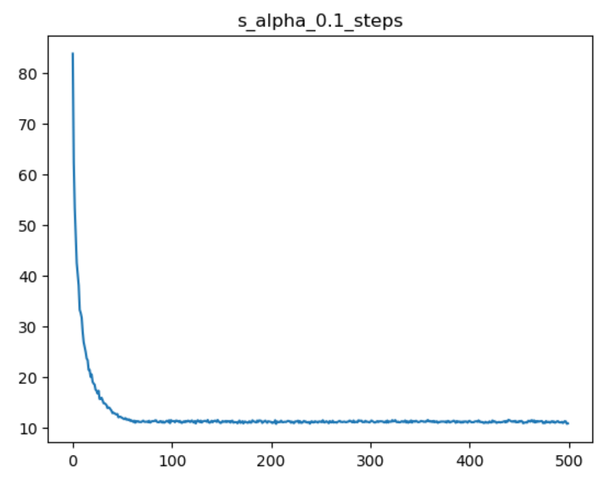


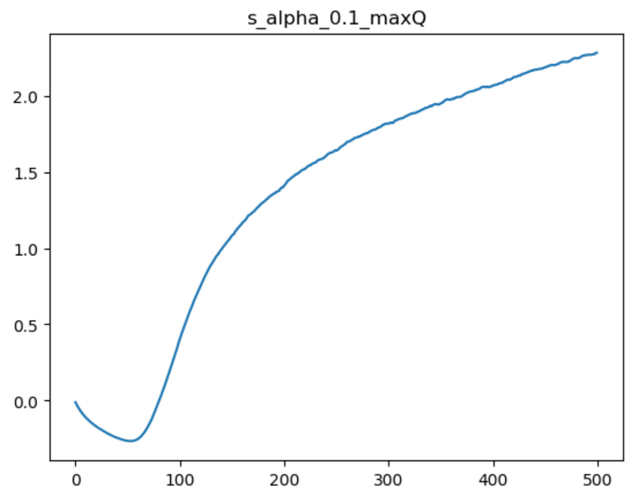
Increasing α to 0.01 shows a similar immediate improvement to the results of both the steps and the max-Q, but both seem to converge even more slowly than with Q-learning, as the max-Q plot does not even show the same "uptick" as the equivalent Q-learning plot.



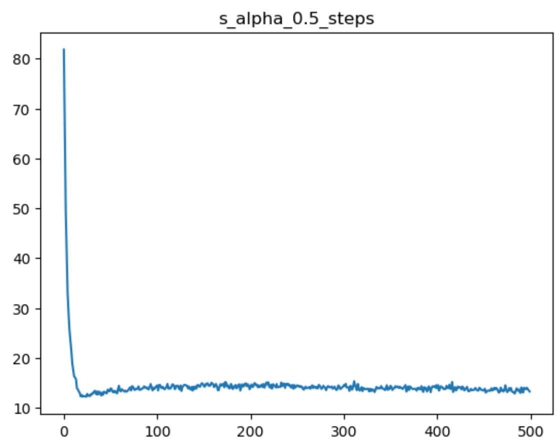


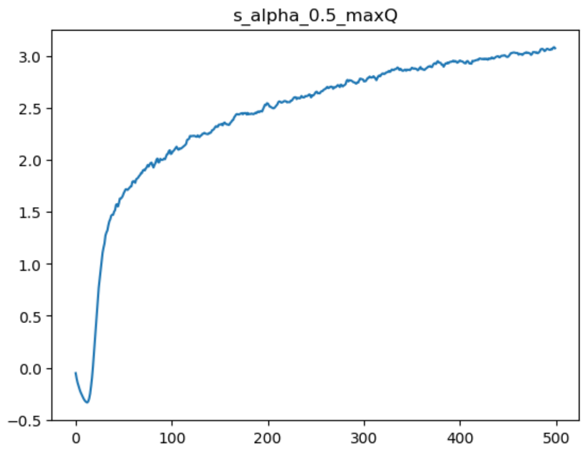
At α=0.05, we see that the max-Q plot begins to show the upwards trend, but is nowhere near approaching any sort of convergence. The average steps plot also takes around 150 episodes to converge to its final value of around 10 steps.



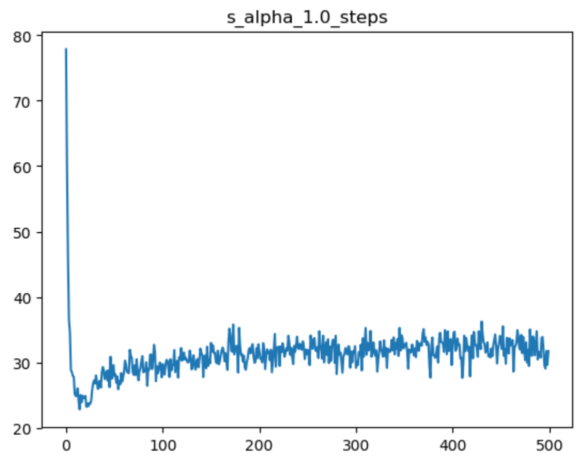


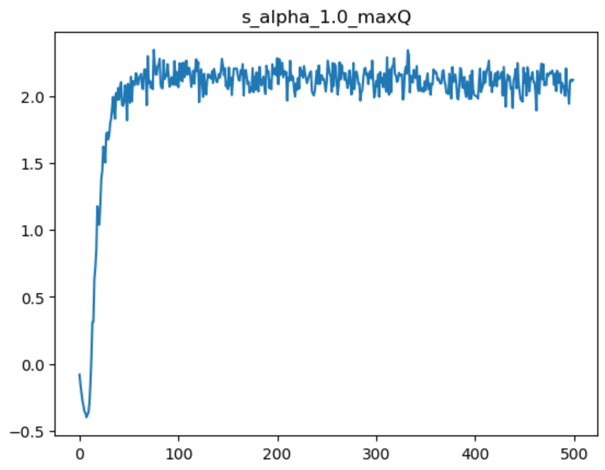
As α increases further to 0.1, the trends continue, as the average steps converges at closer to 70 episodes, and by 500 episodes the max-Q value has reached around 2.3.





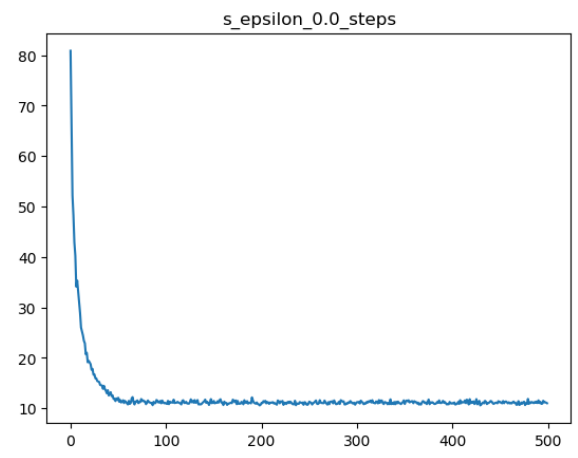
At α=0.5, the plots display somewhat odd behavior. The average steps quickly drops to around 12, but then bounces between 13 and 15, without approaching the previously determined value of 10 steps. The max-Q plot shows similar jitter, although it continues to get closer to converging, this time reaching around 3.0 by 500 episodes.

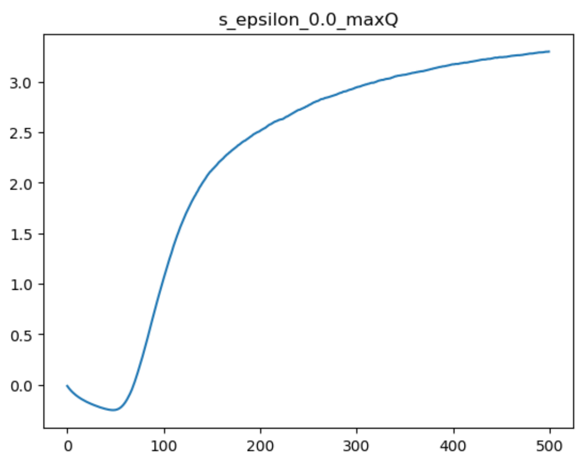




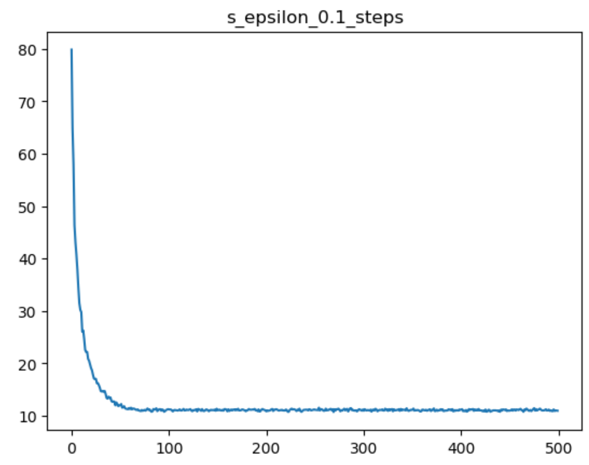
At α=1, the odd behavior that began to occur at α=0.5 further intensifies, with the average steps plot showing further variance, this time between 25 and 35, and the max-Q plot seems to settle somewhere around 2.0. I hypothesize that these strange behaviors at α=0.5 and α=1 occur due to the fact that Sarsa(λ) is an on-policy method, and therefore is more susceptible to local optima.

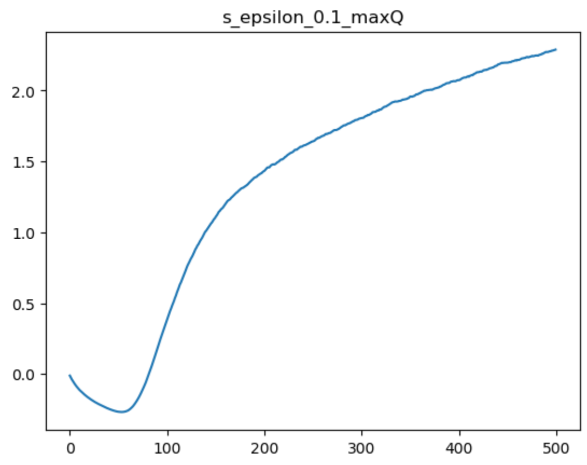
Below are graphs using Sarsa(λ) with various values of ε:



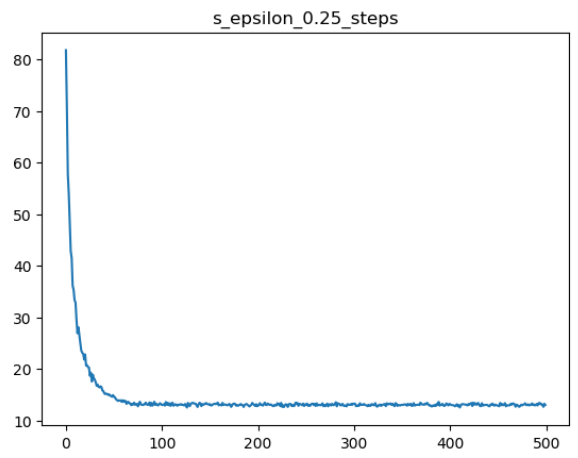


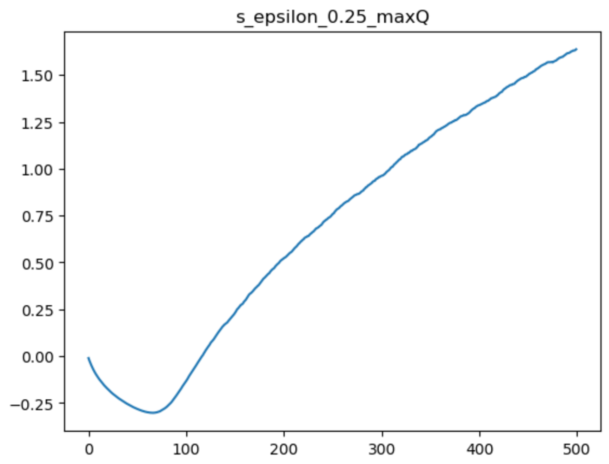
At ε=0, the plots show generally decent performance, although not as strong as with Q-learning. Notably, the max-Q value still fails to fully converge to a final value.



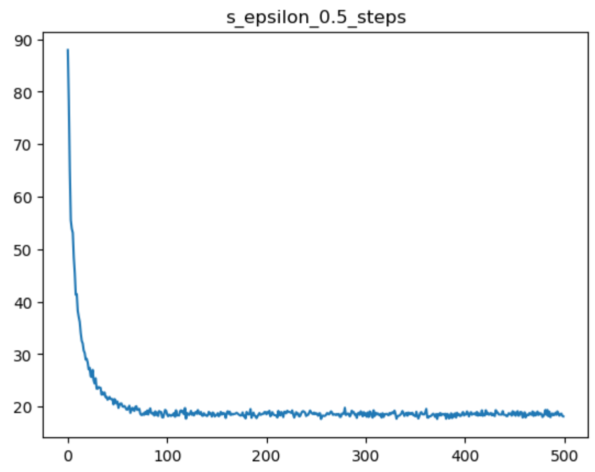


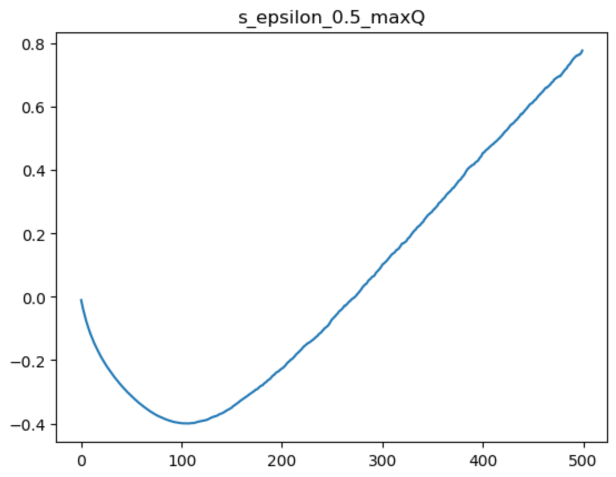
Increasing ε to 0.1 paradoxically reduces the performance of the max-Q values, as we see that by 500 episodes the max-Q values have only reached around 2.5.



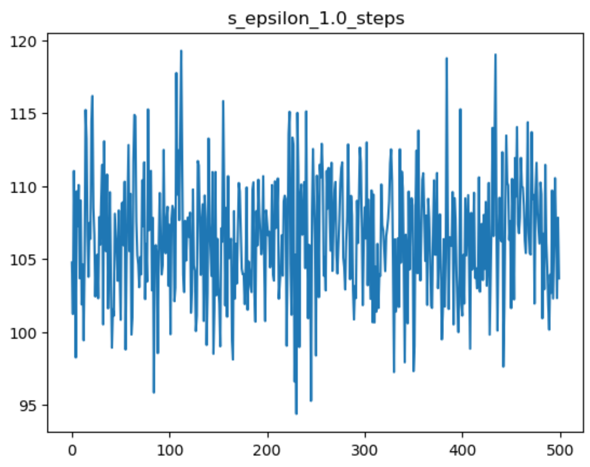


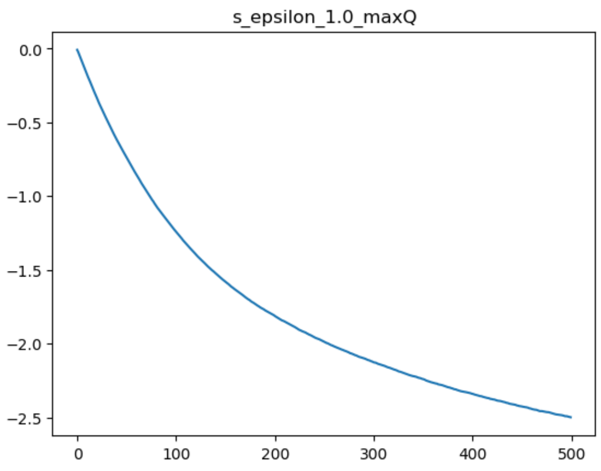
At ε=0.25, the trend continues: now the max-Q values only reach 1.6 by 500 episodes.





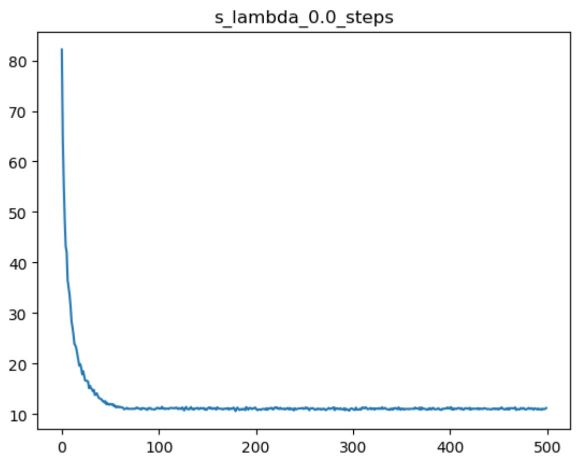
With ε=0.5, the max-Q values now only reach 0.8 by 500 episodes.

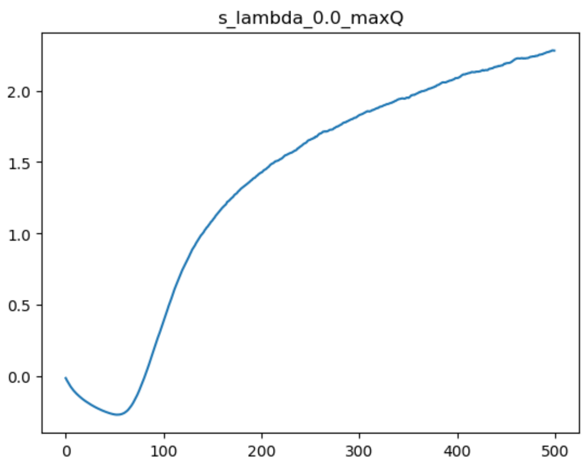




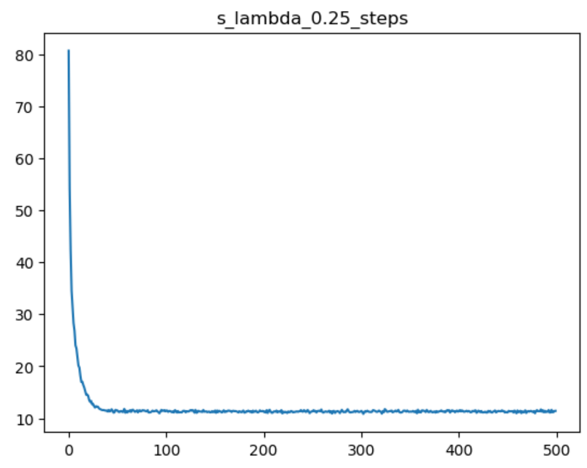
Finally, at ε=1, we see a steps graph similar to when α=0, but unlike with α, the max-Q plot looks similar to when α=0.01, although it drops to -2.5 rather than -0.25. I believe that the reason why the performance decreased as ε increased is due to the fact that increasing stochasticity via ε actually prevents Sarsa(λ) from utilizing the policy it is attempting to optimize properly.

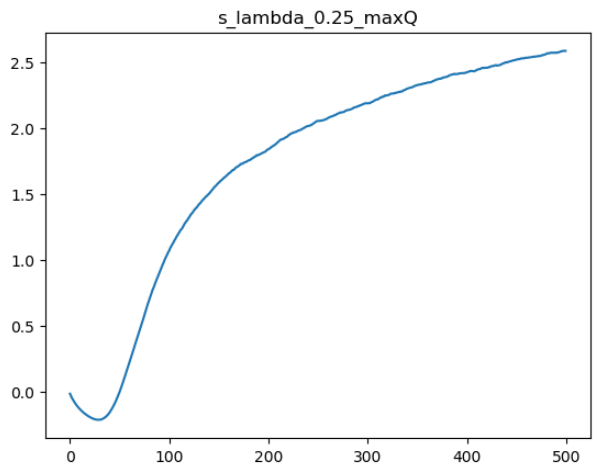
Below are graphs using Sarsa(λ) with various values of λ:



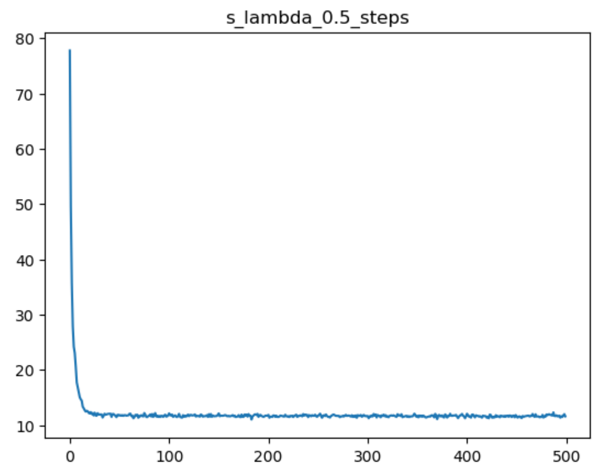


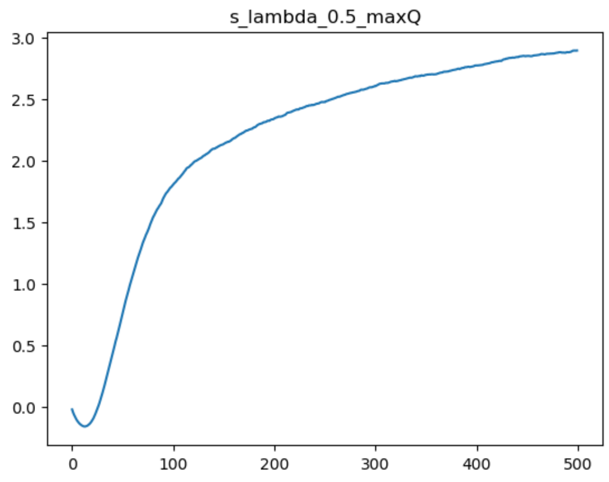
At λ=0, we see decent performance of the steps graph, although the max-Q values only reach around 2.3.



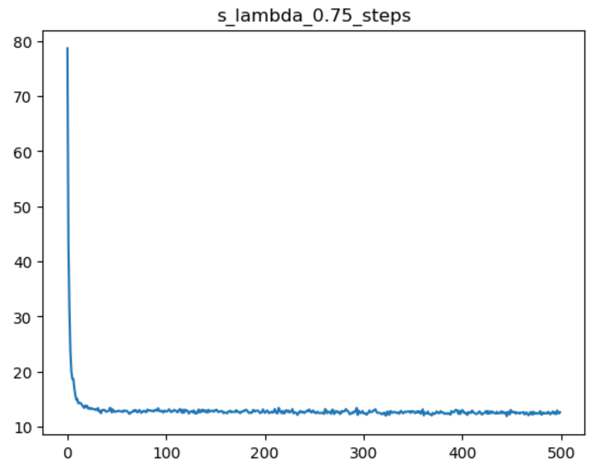


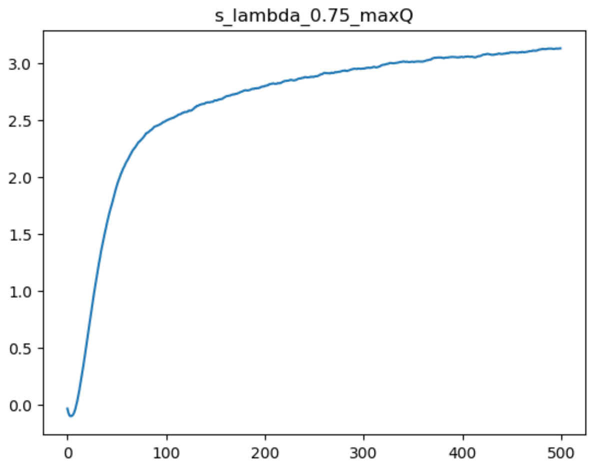
At λ=0.25, the number of steps converges slightly faster, and the max-Q value reaches 2.5 by 500 episodes.



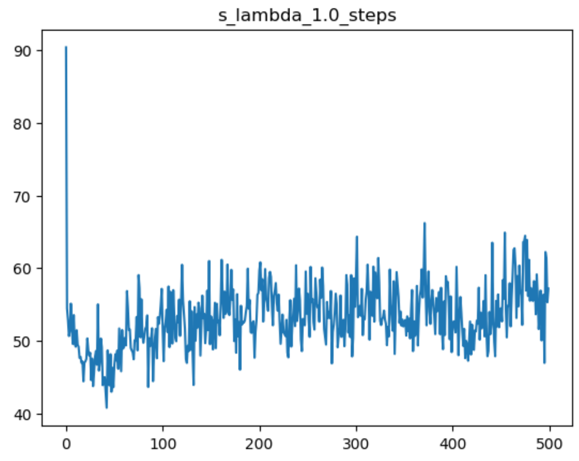


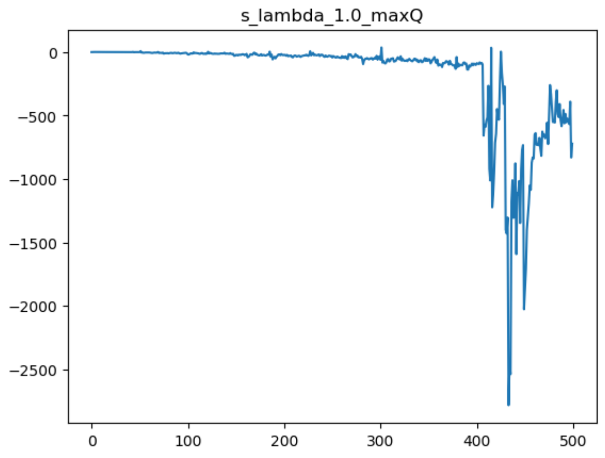
As λ increases to 0.5, performance further increases: the steps converge at around 20 episodes, and the max-Q value approaches almost 3.0 by 500 episodes.





At λ=0.75, it is difficult to see a significant improvement in the step convergence, and max-Q reaches around 3.1 by 500 episodes.



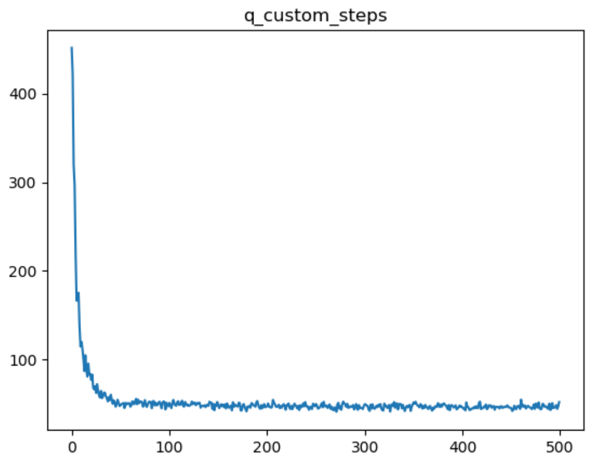


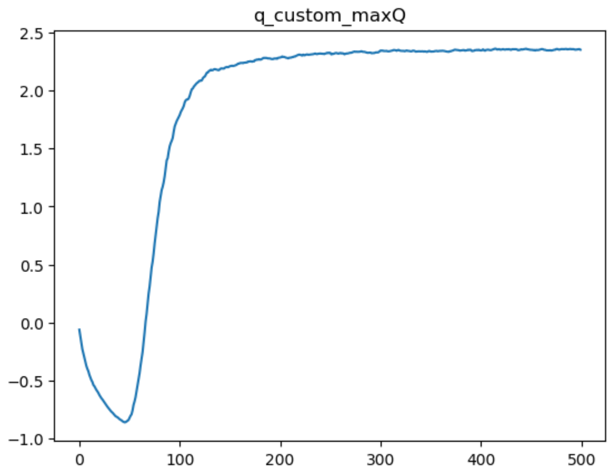
When λ=1, both the average steps and max-Q plots break down; the steps plot varies wildly between 45 and 65, and the max-Q plot decreases somewhat dramatically.

3. Custom plots on 10x10 grid

1. Q-learning

For Q-learning, I opted to use α=0.5 and ε=0.5, which yielded the following plots:



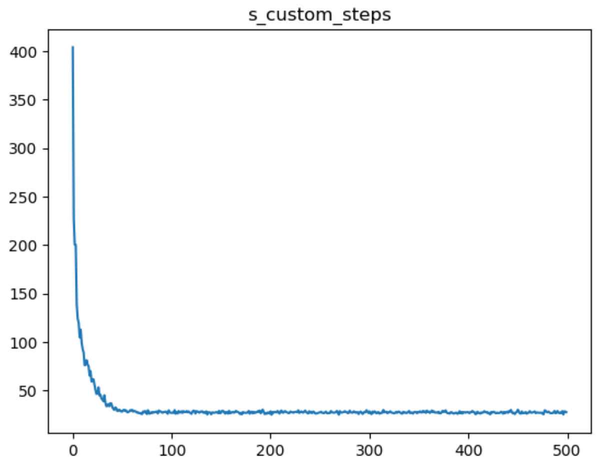


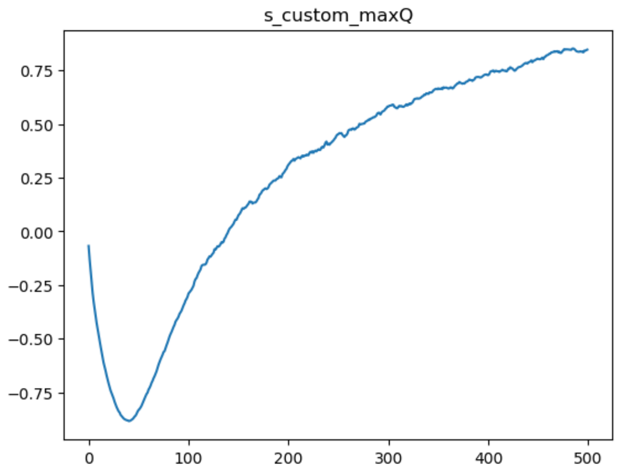
Based on the steps plot, it appears to take roughly 50 episodes to converge to the optimal policy at around 30 steps, although it takes much longer for the max-Q values to converge.

The reason why I chose these values for α and ε was that, in looking at each set of graphs where we varied the respective parameters independently of all other parameters, these values corresponded to the fastest convergence in number of episodes for the step count, while also having good performance in terms of converging on the max-Q value.

2. Sarsa(λ)

For Sarsa(λ), I opted to use α=0.1 ε=0.1, and λ=0.75, which yielded the following plots:





Here we see the number of steps converges at roughly the same number of episodes as in Q-learning, roughly 50 episodes, but the max-Q values converge much more slowly.

I chose these values for α, ε, and λ similarly as with Q-learning. The reason why α and ε are different than in the case of Q-learning, is that higher values of α and ε seem to decrease the rate of convergence for the Q-values, and therefore a lower value for these two parameters yields better performance in terms of finding the Q-values.

3. Choosing α, ε, and λ:
Choosing values for α, ε, and λ depends on whether the learning algorithm is Q-learning or Sarsa(λ). For Q-learning, larger values for α and ε are preferred (but not equal to 1), while for Sarsa(λ) smaller values for α and ε (but not too close to 0) yield better results, though a higher value for λ is also better.