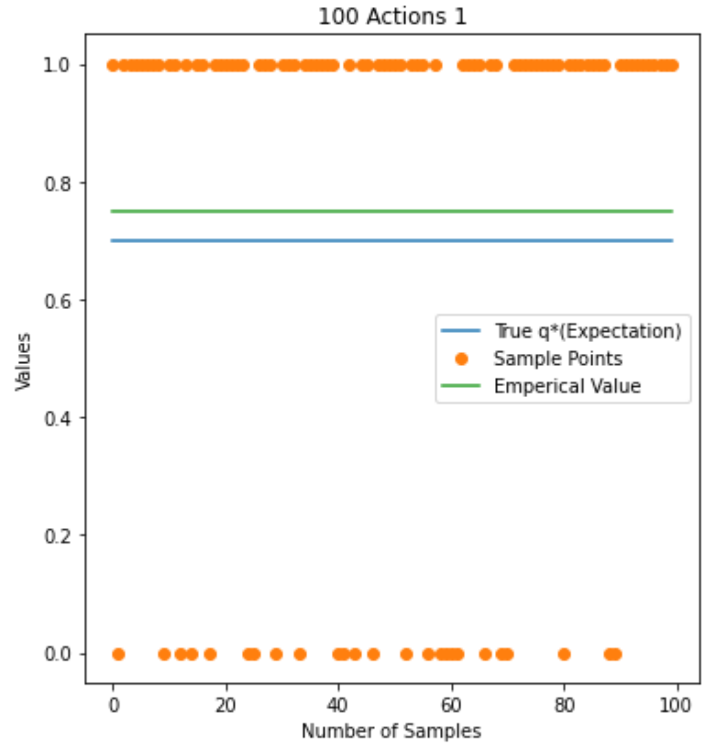
Comp579

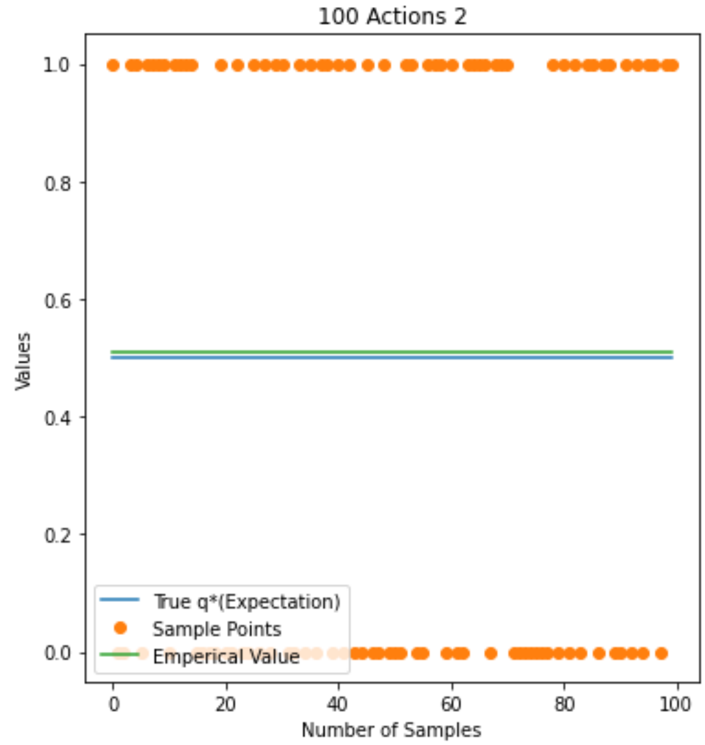
ASSIGNMENT1

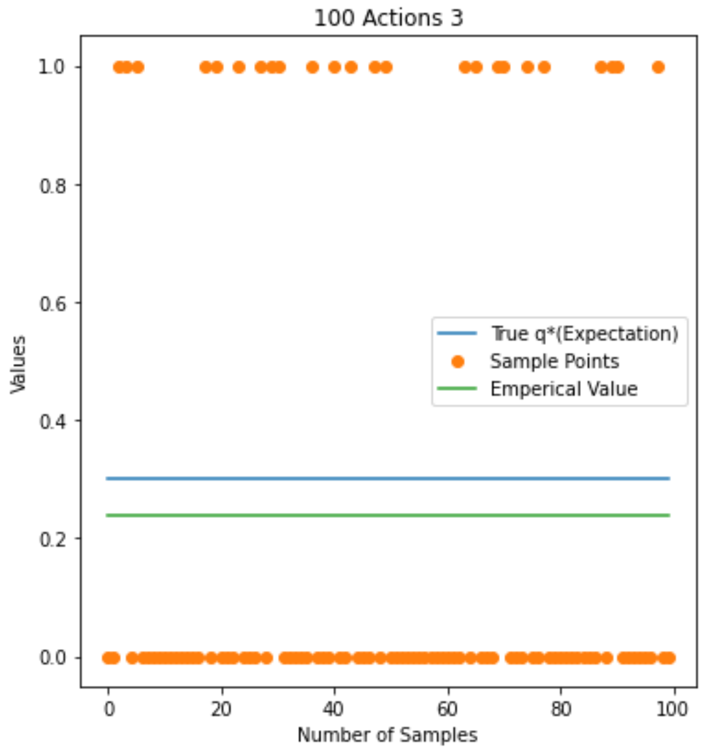
YE YUAN 260921269

The link for code:
https://colab.research.google.com/drive/15QxFxMXNvVEUkI0JmVunY5V5vl7Veg_m?usp=sharing

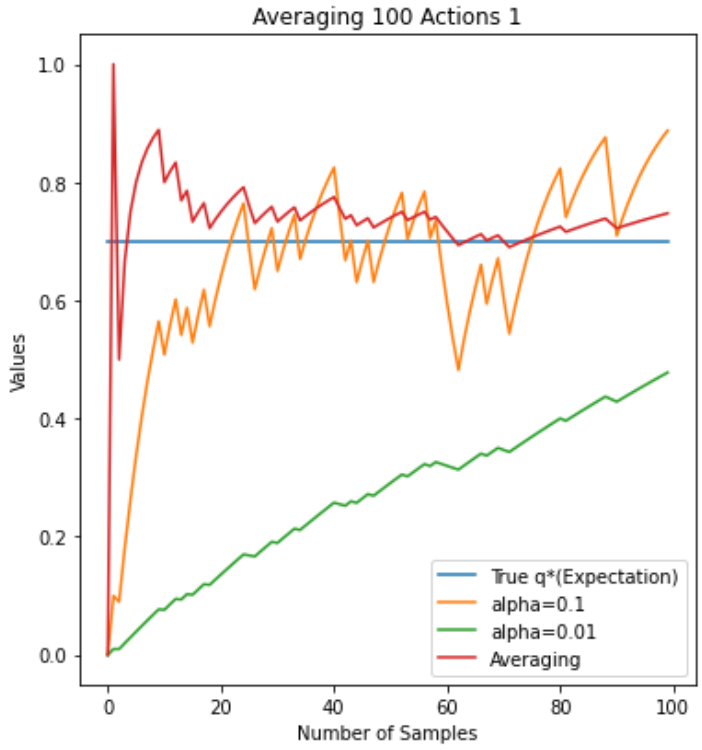
Q2:

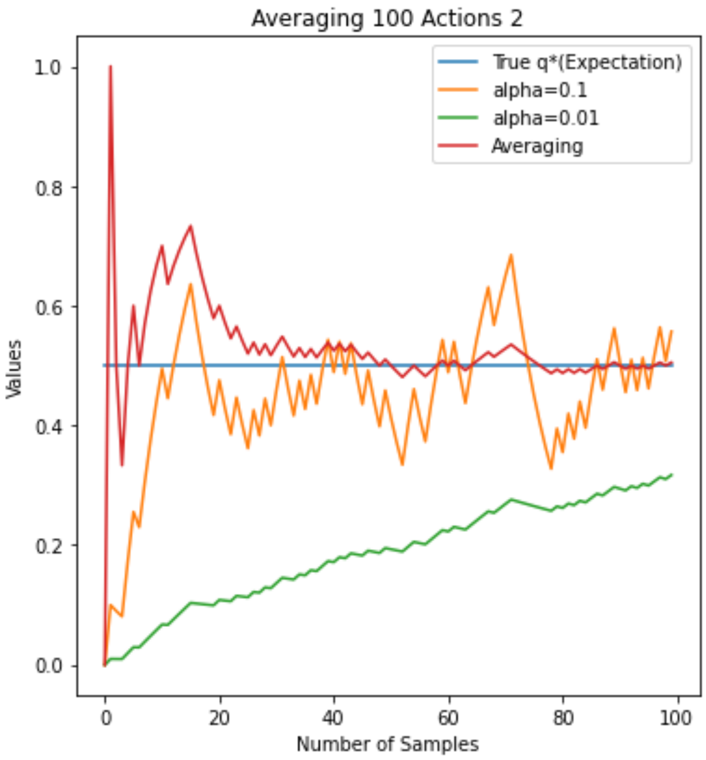


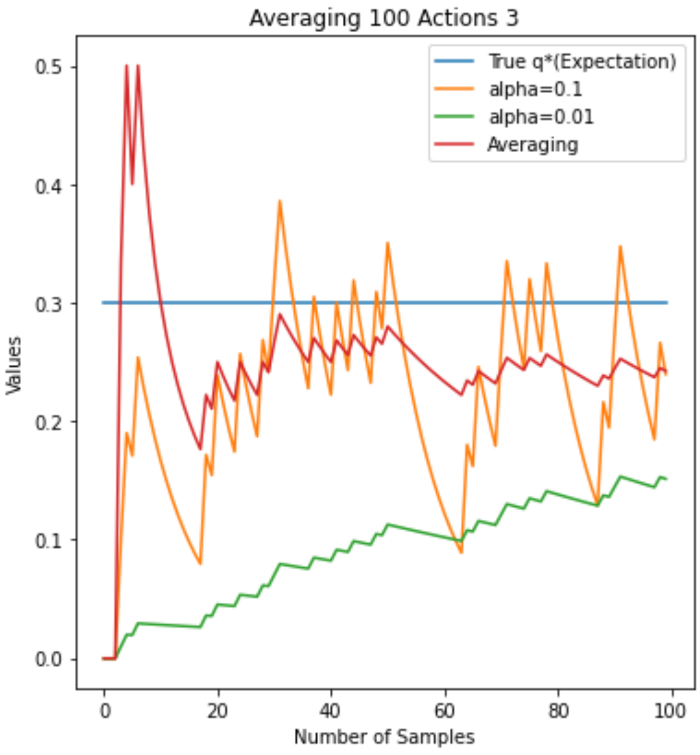



100 Actions 3
Values
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
Number of Samples
True q*(Expectation)
Sample Points
Emperical Value

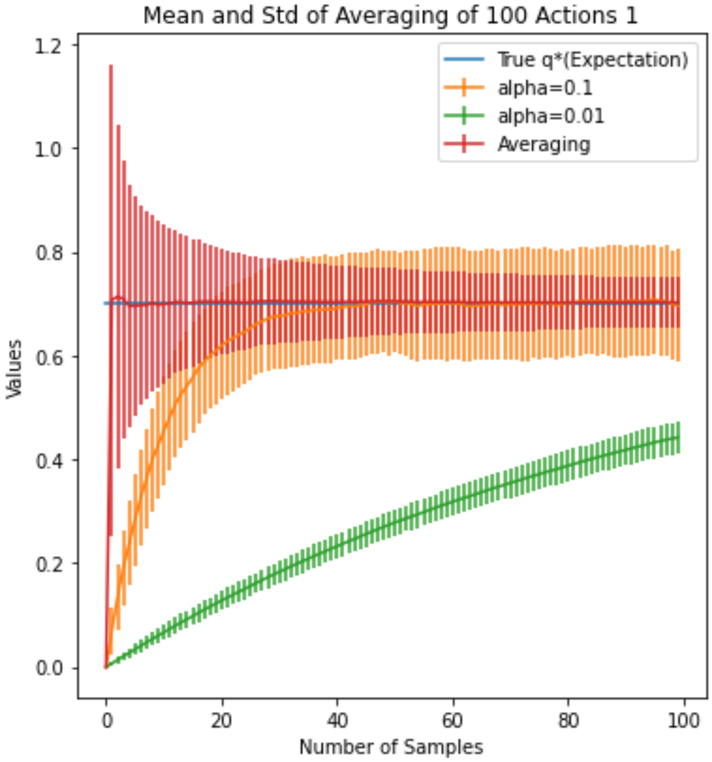
Q3:

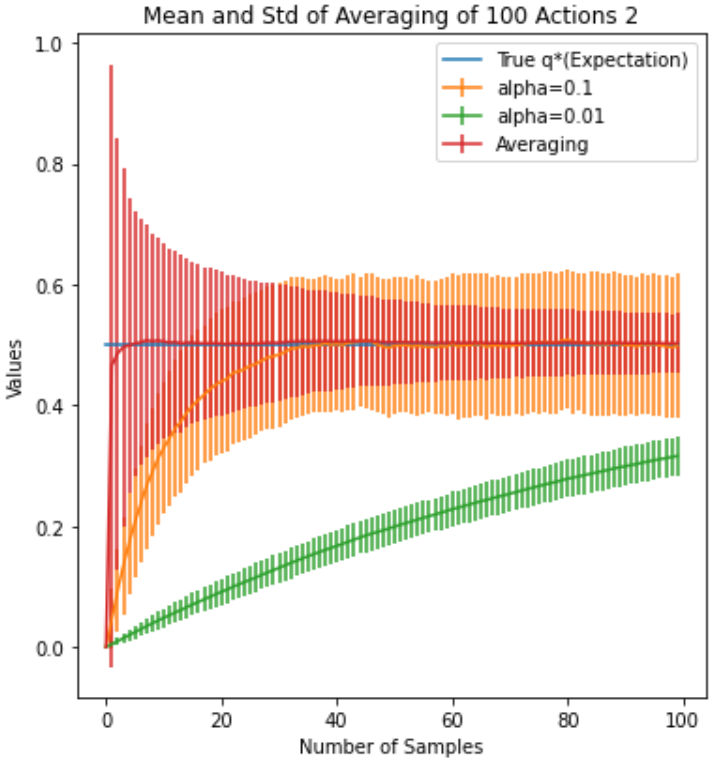

Averaging 100 Actions 1
Values
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
Number of Samples
True q*(Expectation)
alpha=0.1
alpha=0.01
Averaging

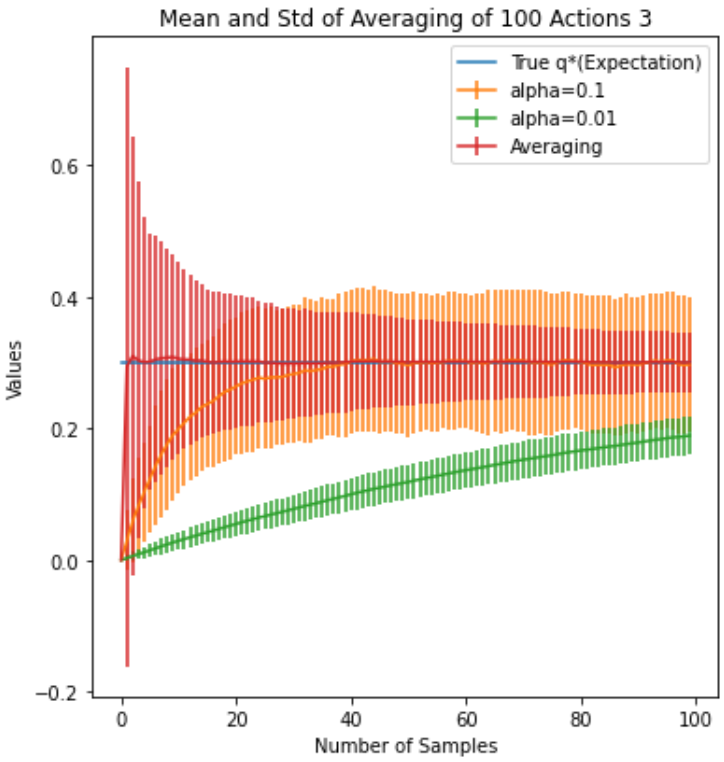

Averaging 100 Actions 2
True q*(Expectation)
alpha=0.1
alpha=0.01
Averaging
Values
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
Number of Samples


Averaging 100 Actions 3
True q*(Expectation)
alpha=0.1
alpha=0.01
Averaging
Values
0.5
0.4
0.3
0.2
0.1
0.0
0
20
40
60
80
100
Number of Samples

Q4:

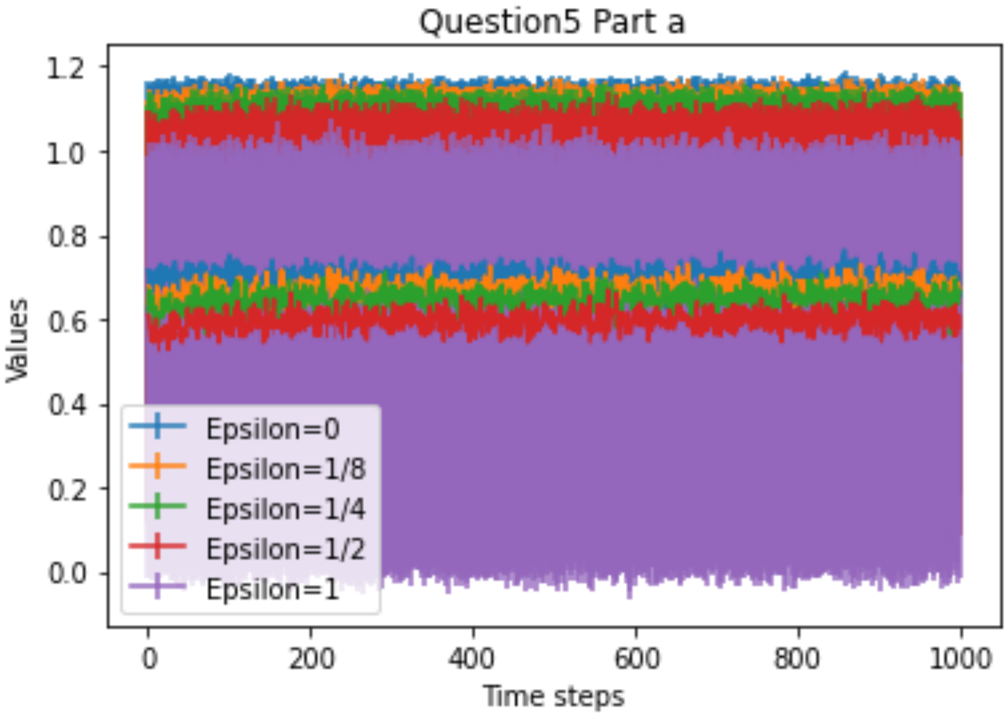

Mean and Std of Averaging of 100 Actions 1
True q*(Expectation)
alpha=0.1
alpha=0.01
Averaging
Values
Number of Samples
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
20
40
60
80
100

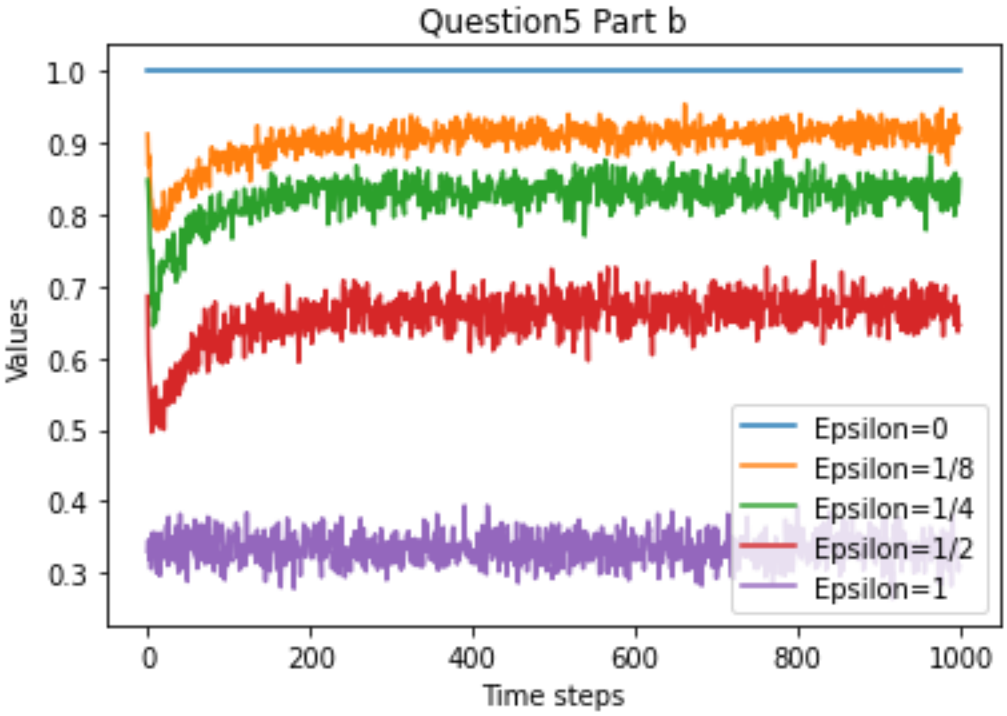

Mean and Std of Averaging of 100 Actions 2
True q*(Expectation)
alpha=0.1
alpha=0.01
Averaging
Values
Number of Samples
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

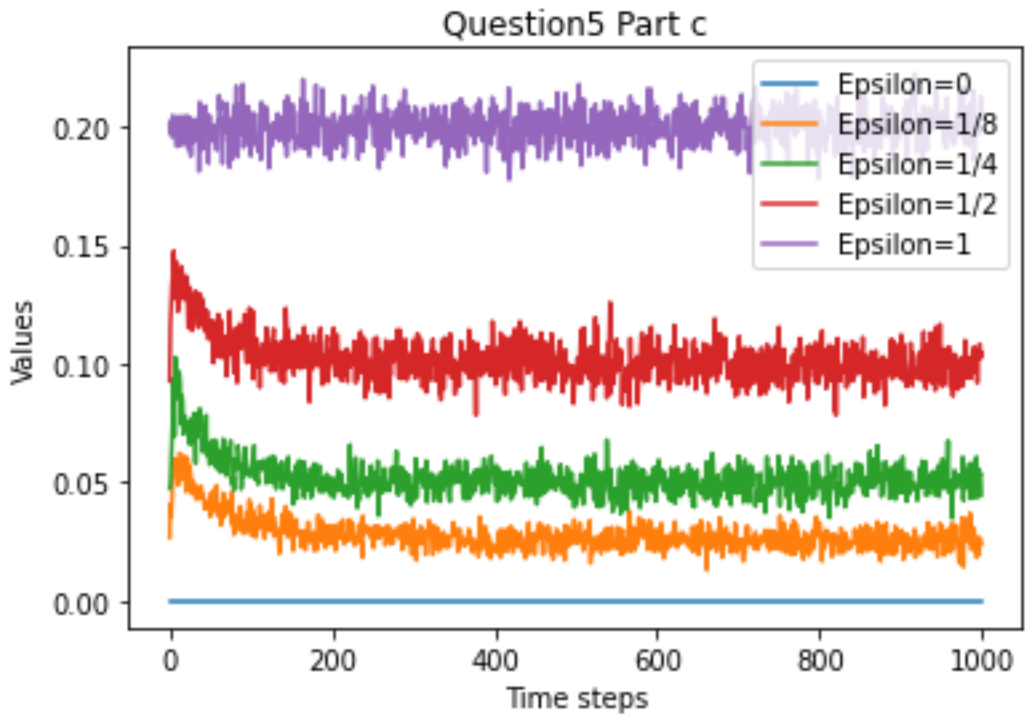


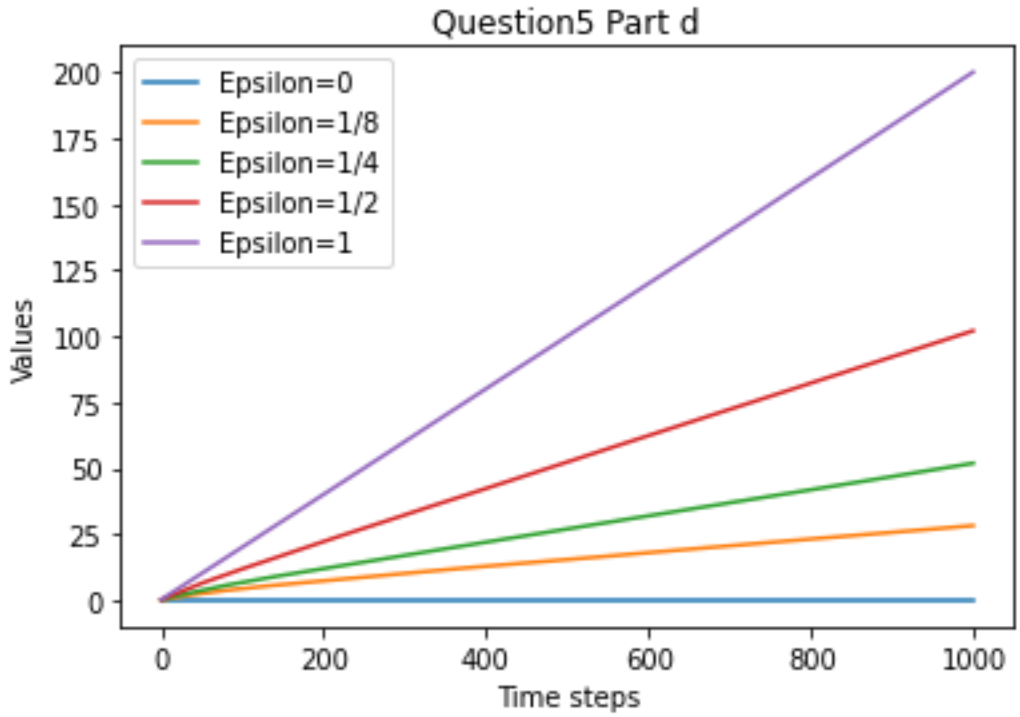
In the graphs above, I observed that the performance is better when alpha=0.1 since it's closer to the true value. When the value of alpha is large, the more recent values will be taken into the estimation. The larger value of alpha results in a faster convergence speed. As shown in the graphs, the curves of alpha=0.1 converge to the true value faster. However, the larger value of alpha results in a larger standard deviation, which means the results are not very stable. To balance between the stability of the results and the convergence speed, we should look for a better value of alpha in the range 0.01 to 0.1, such as 0.04 or 0.05, which will converge to the true value faster than alpha=0.01 and has a smaller standard deviation than alpha=0.1.

Q5:


Question5 Part a
Values
Time steps
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
Epsilon=0
Epsilon=1/8
Epsilon=1/4
Epsilon=1/2
Epsilon=1


Question5 Part b
Values
Time steps
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
200
400
600
800
1000
Epsilon=0
Epsilon=1/8
Epsilon=1/4
Epsilon=1/2
Epsilon=1

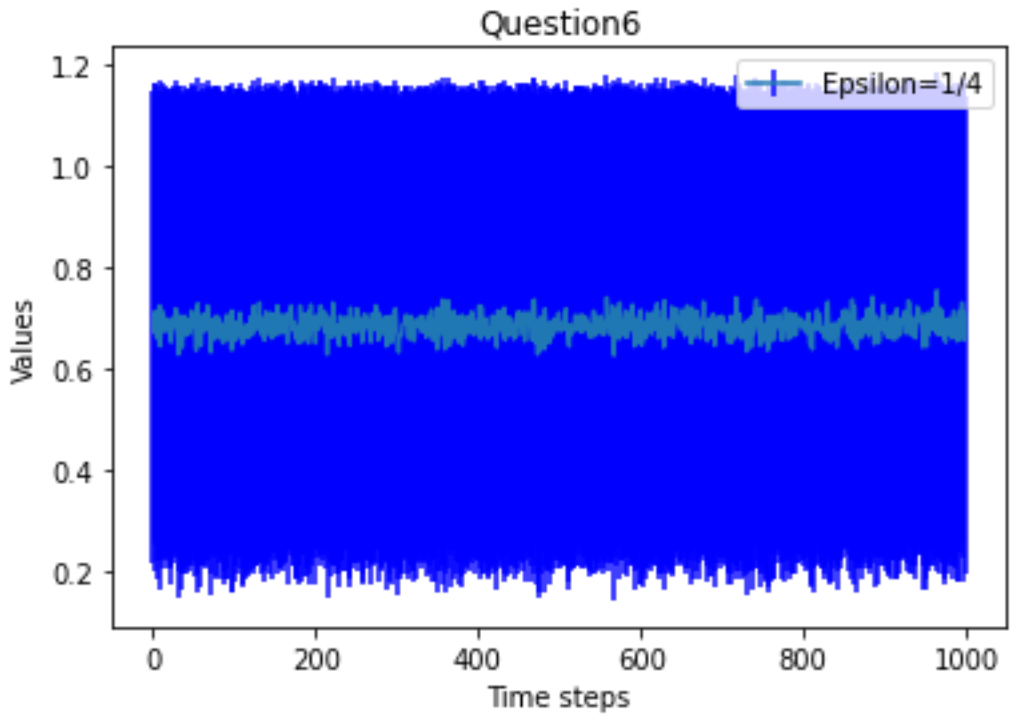


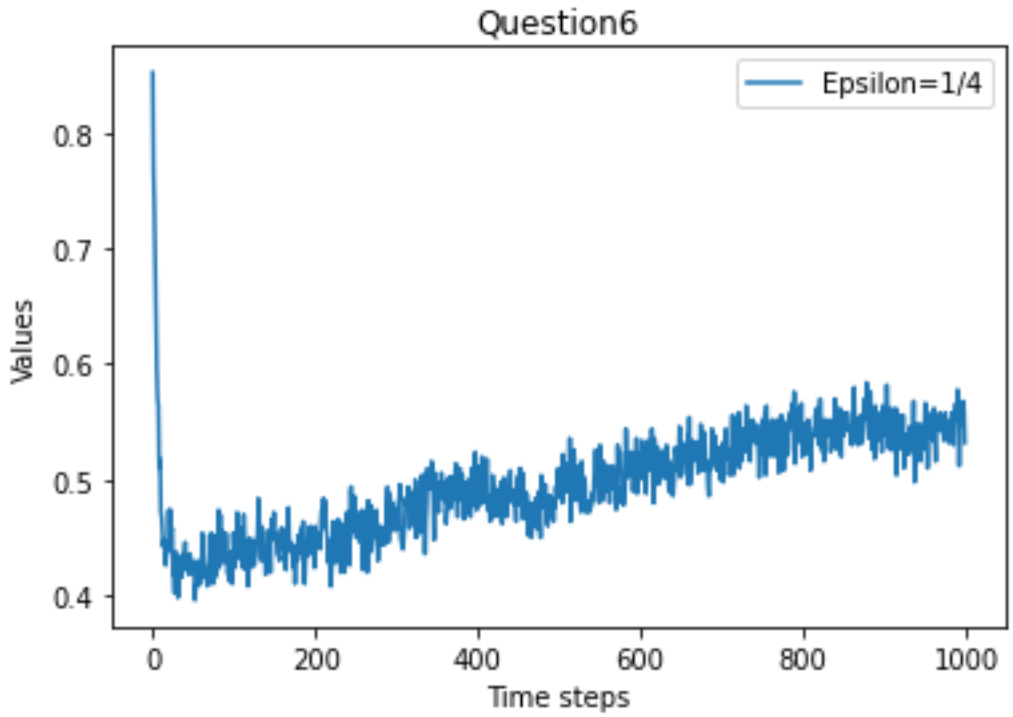


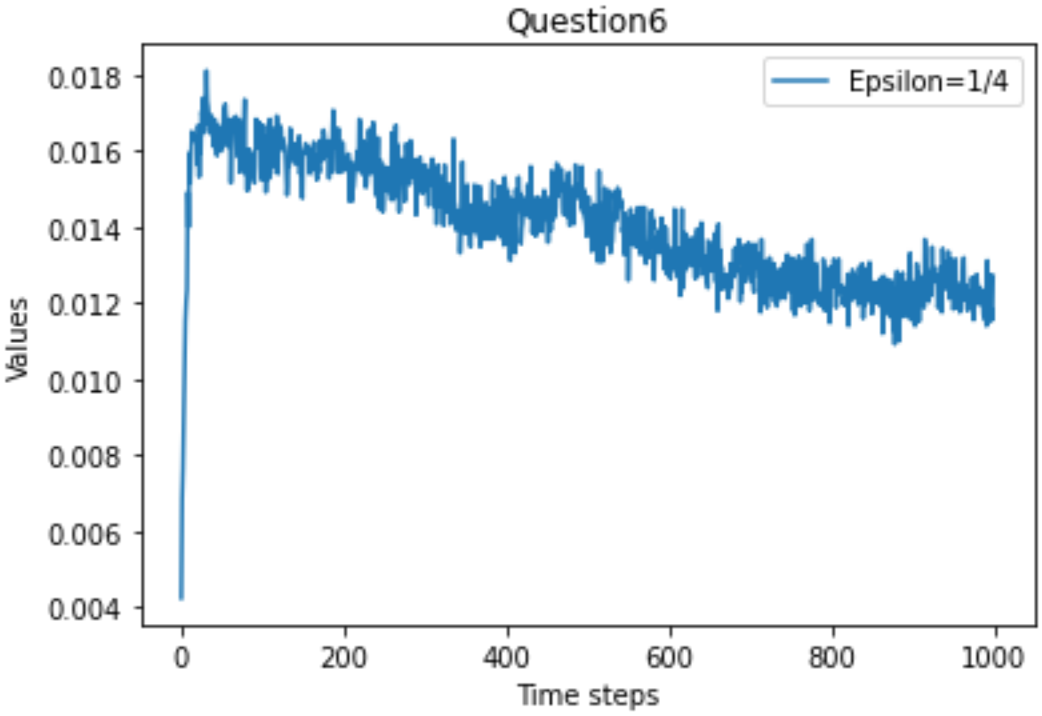
From the graphs above, I noticed that the average reward and standard deviation decrease as the value of epsilon increases, the fraction of correct estimation decreases as the value of epsilon increases, and the instantaneous regret and total regret increase as the value of epsilon increases. The reason is that when the value of epsilon is larger, we explore more for the actions. The decrement of reward is because we more randomly choose the actions rather than picking the one with the highest value, and so the standard deviation decreases as well.

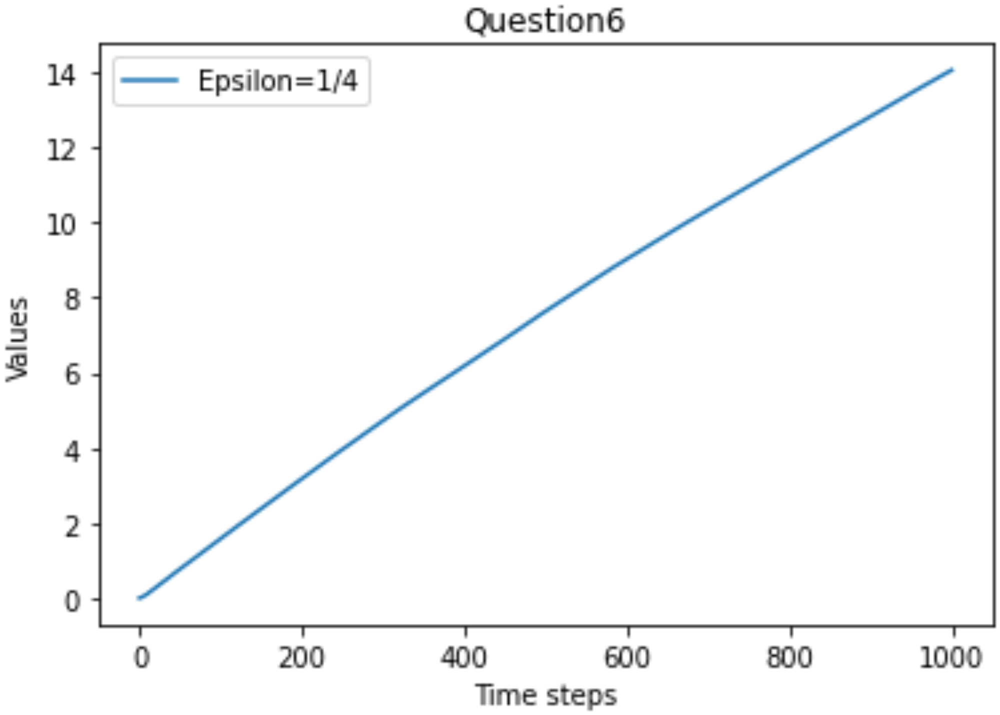
Similarly, the fraction we picked the correct action would decrease as well since we choose action more randomly, and the regret will increase. Personally speaking, the best epsilon is 1/8. It will help us to have a chance to explore all actions and have a greater probability to choose the currently highest-valued action, which means we don't pay too much effort to explore new actions.

Q6:



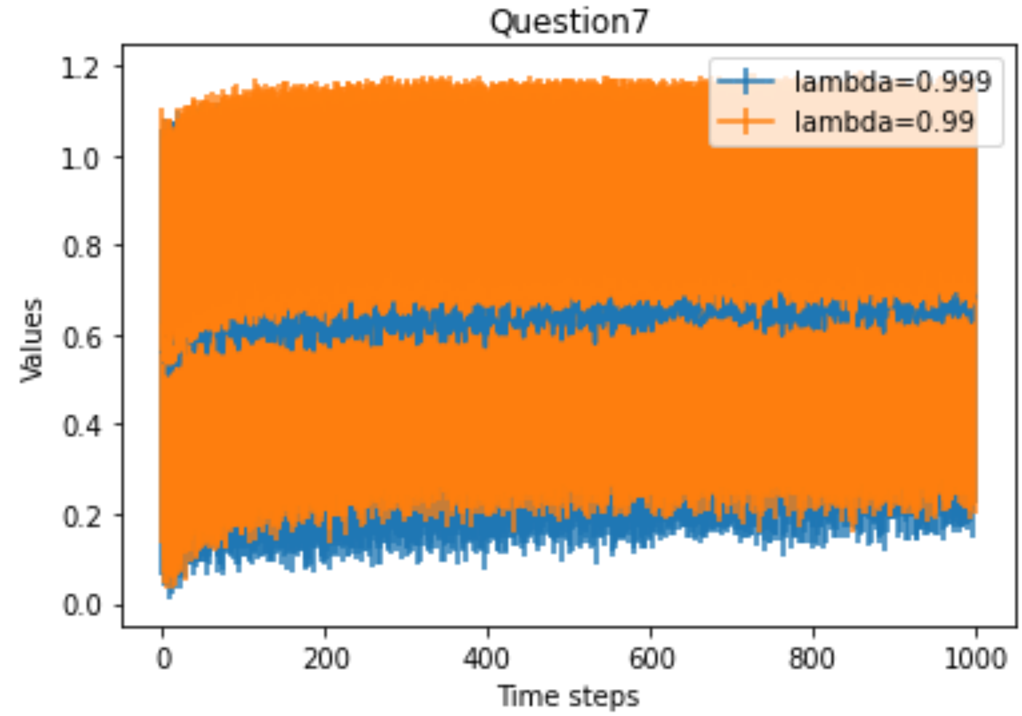


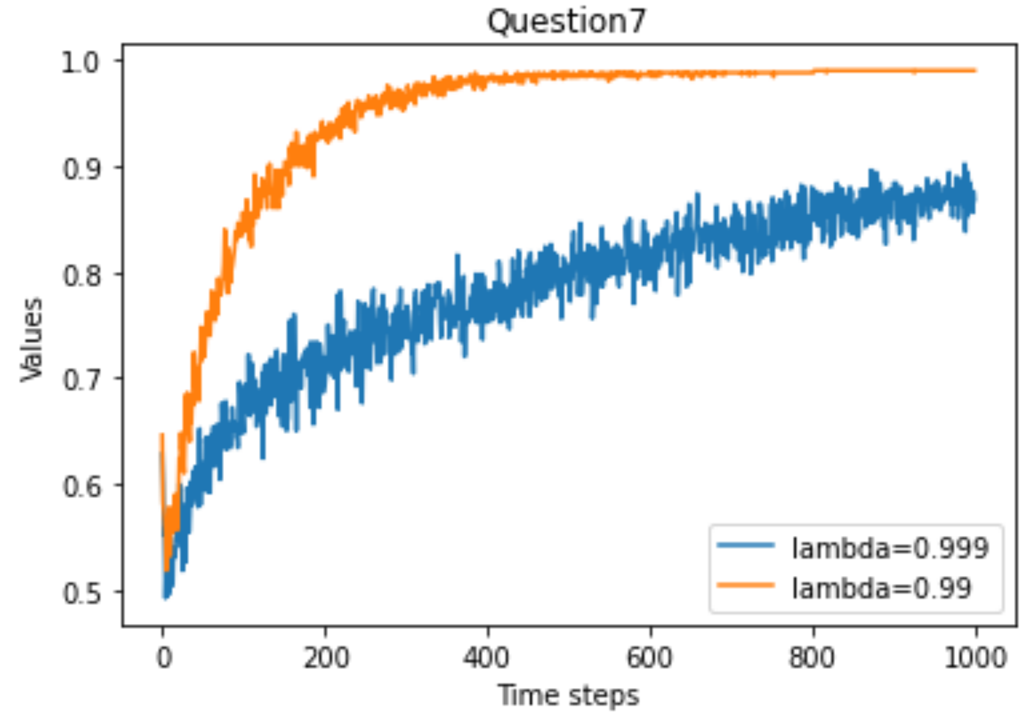


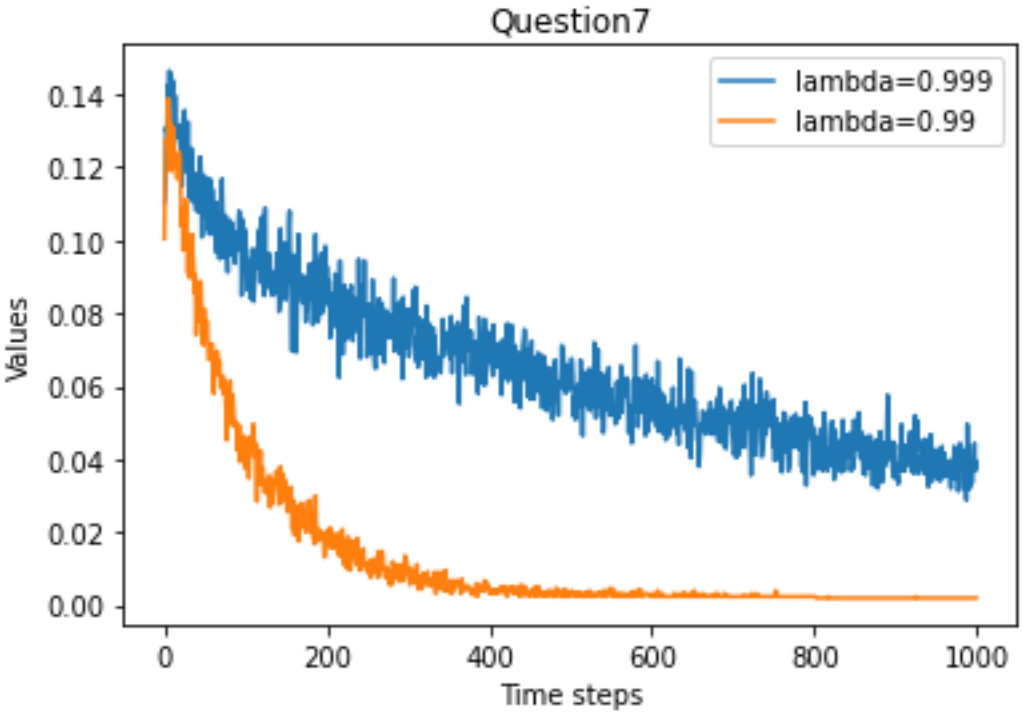


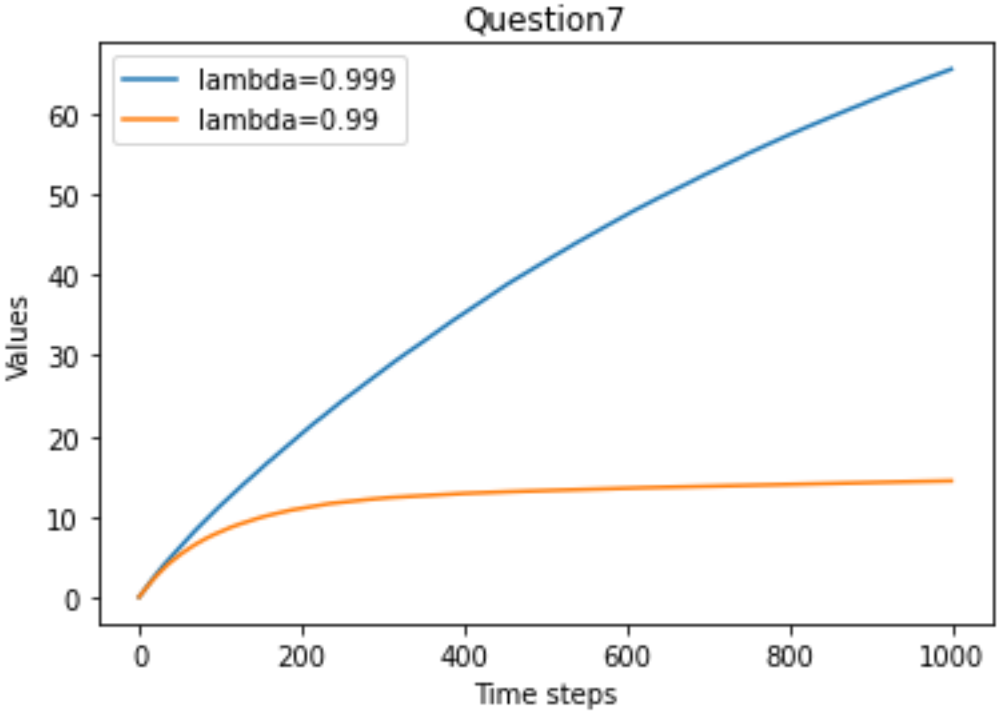
Personally speaking, delta=0.02 results in a higher average reward, lower standard deviation, lower fraction of correct estimates, a lower instantaneous regret and total regret. The potential reason for these facts is that these three actions are comparable. The rewards of these three actions are almost same. As a result, we can achieve higher rewards. And since the difference between these three actions is small, we also have lower regrets. However, since these three actions are almost same, the fraction of correct estimates will decrease. (hard to distinguish)

Q7:


Question7
lambda=0.999
lambda=0.99
Values
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
200
400
600
800
1000
Time steps


Question7
Values
0.5
0.6
0.7
0.8
0.9
1.0
lambda=0.999
lambda=0.99
0
200
400
600
800
1000
Time steps

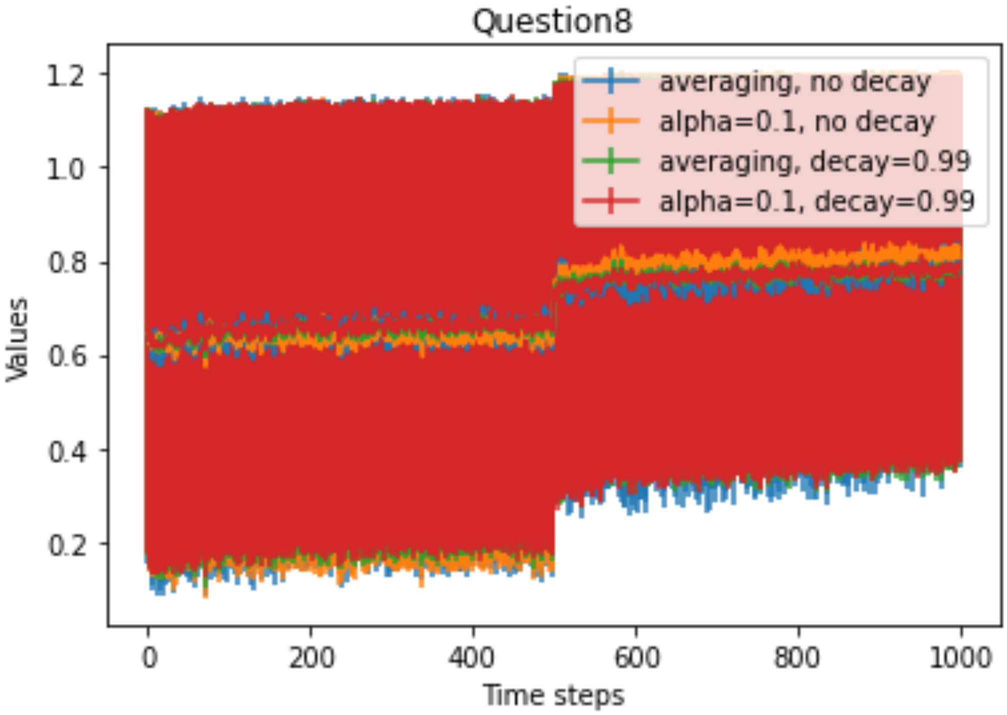




As shown in the above graphs, the rewards achieved by lambda =0.99 are greater than lambda =0.999. And lambda =0.99 also has a higher fraction of optimal choice, lower instantaneous and total regrets. The reason is that when the value of lambda is larger, we will explore more, that is we will randomly choose an action. However, when we have explored for a while, we will be more confident in the estimation of each action. Compared to exploring, it's better to exploit the best action based on our estimation.

Compared to the previous experiment, both lambda=0.99 and lambda=0.999 achieve higher rewards, higher faction of correct estimation, lower instantaneous regrets and total regrets. The reason is similar to the above. When we have explored for a while, we have a better estimation of the value of each action. As a consequence, we could explore less and exploit more, so adding a decay factor to the epsilon will achieve better performance.

Q8:

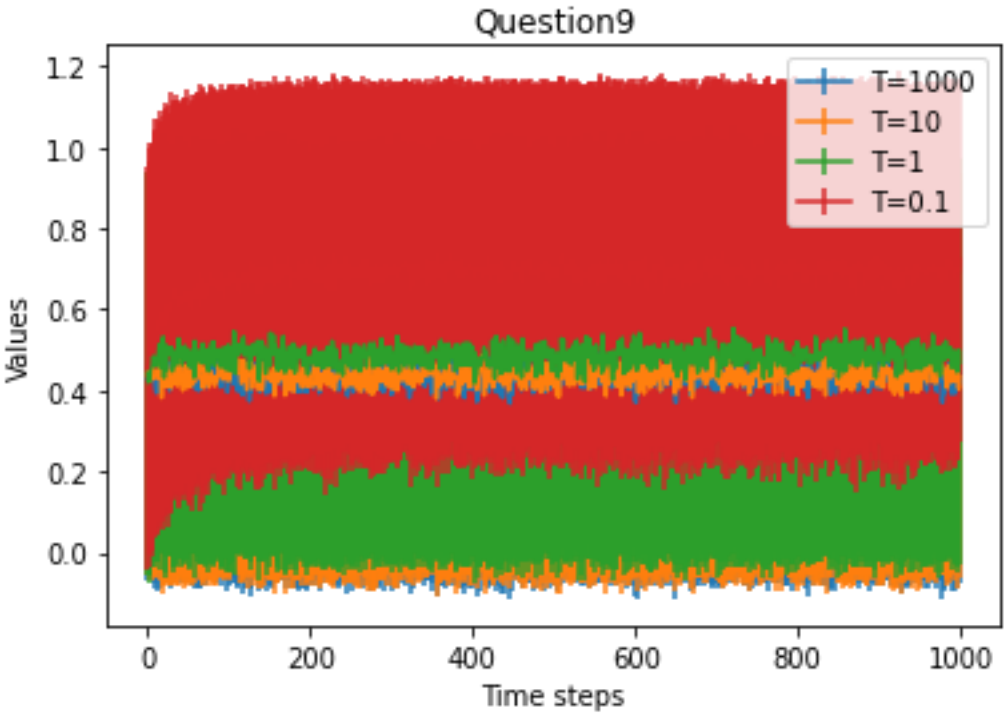


In this experiment, I chose the value of epsilon to be 1/2, the value of alpha to be 0.1, and the value of lambda to be 0.99 for the following reasons. In question4, I concluded that choosing alpha=0.1 will be closer to the true value. In the above question7, I concluded that choosing lambda=0.99 will get higher rewards since it will explore less gradually. Meanwhile, in this question8, we consider the non-stationary condition. So, I choose epsilon=1/2 because it will make us explore more. Considering this is a non-stationary condition, exploring more will be helpful to get a more accurate estimation of the value of each action.

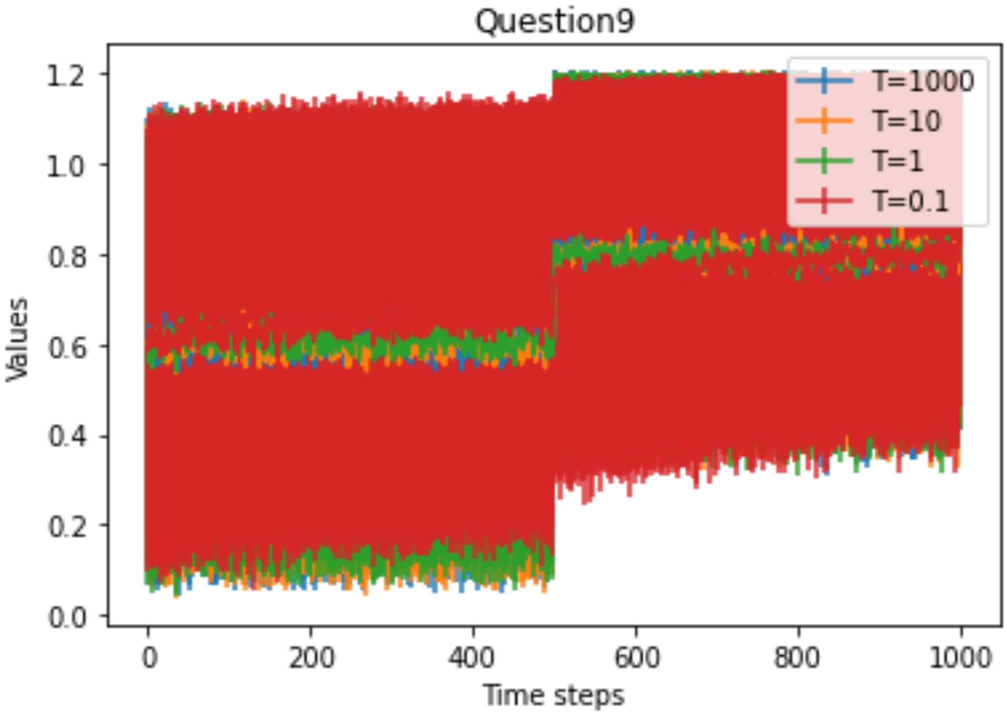
As shown in the above graph, before the time step=500, decaying epsilon has better performance for the same reason in question7. It reduces the rate of exploration gradually, so we will have more optimal selections. After changing the probability of action2 and action3, no decaying epsilon performs better than others. The reason is that we will keep the rate to explore action, so it will notice that action3 becomes the best action to choose. Moreover, alpha=0.1 will help a lot compared to the averaging case, since we don’t pay too much attention to the old data and only focus on the recent data.

In a summary, considering the performance before time step=500 and after time step=500, I think the most suitable configuration is decaying epsilon with a fixed value of alpha, which will achieve the best performance.

Q9:
Stationary:



Non-Stationary:



As shown in the graphs above, when temperature=0.1, we have the largest rewards. As we discussed in class, when the temperature gets to zero, the softmax exploration acts similarly to the ε-greedy algorithm. In other words, it prefers exploitation. When the temperature gets to infinity, it acts like the uniform selection, which means it prefers exploration.

The similarity between the ε-greedy algorithm and softmax exploration is that when the temperature gets to zero, the softmax exploration is the same as the ε-greedy algorithm. Another same point is that they all use hyper-parameter to tune the relationship between exploration and exploitation.

The difference between these two methods is that the ε-greedy algorithm will randomly choose action when exploring. However, the softmax exploration will choose an action with probability proportional to its current value(that is the preference to an action).