(1):

(a): After performing the simulation, we can observe that when epsilon is 0, the total reward is always be zero, and the agent is stuck looping in an area near C (the delivery center). This happened because the agent followed a purly greedy policy ($\epsilon = 0$) that keeps it from exploring new routes by performing other actions. Thus, the agent will always follow "the loop" rather than trying to explore other possibilies.

(b): After performing the simulation, we can notice that when epsilon is 0.1, the agent found a route from the dilivery center to customer 1 and 2, with positive rewards. This happened because the agent started to take actions off-policy and thus exploring new and potentially better routes. (Sometimes, this will cause negative total rewards in a few episodes because the new route it explores might cause more negative rewards).

(c): After performing the simulation, we can notice that when epsilon is set to 0.5, the agent failed to find a route which has positive rewards. In fact, the total reward are negative in most episodes. This happened because the agent choose to take off-policy actions too often so that it failed to keep the route with largest total rewards, which will cause negative rewards as results.

(2):

(a): After performing the simulation, we can see that when epsilon is set to 0.05 and knows_thief is set to n, the total rewards are mostly negative with a few zeros. This happened because the agent failed to determine the action it should take at each step with respect to thief's location. In this way, the agent will always take huge penalties without knowing the reason (where the thief is) and will simply discard the route that causes the negative reward. Thus, the agent will rather be stuck in a loop (total reward = 0 cases) than taking a route with negative total reward.

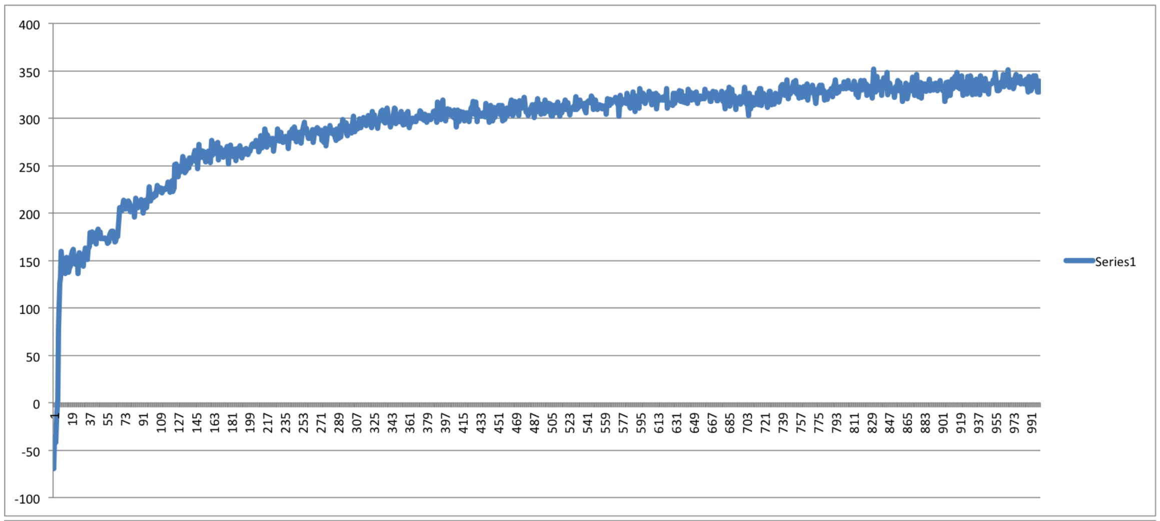
(b): After performing the simulation, we can see that when we set knows_thief to y, the total rewards will grow to a positive value around 30. This happened because the agent now can react with respect to thief's location. Specifically, if the agent is caught by the thief during one epsiode, it will know that the thief's location is this situation will cause high penalty. Thus, the agent will be able to avoid being caught by the thief in future episodes.

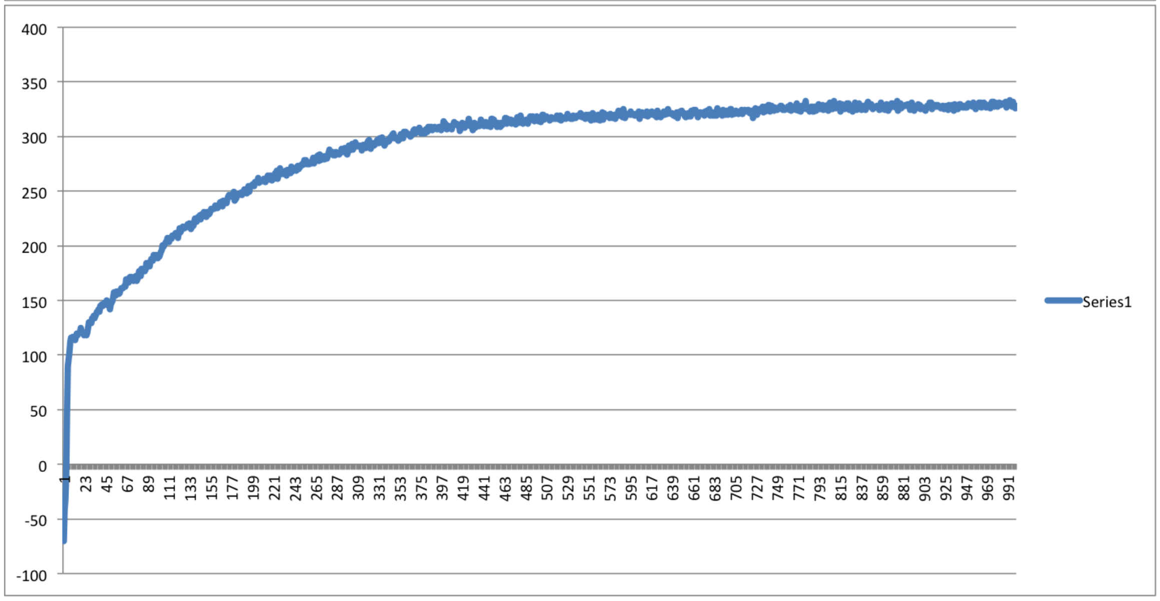
(c): The best learning rate I found is 0.05, and the best epsilon is 0.01. Firstly, I choose a fixed value of learning rate, and search for the best epsilon by gradually change the value from 0.5 to 0.01. I found out that the highest total reward will increase as the epsilon decreases to somewhere nearly 0.01. If the epsilon is too close to zero, the highest total reward we get will actually decrease. Secondly, I search for the best fixed value of learning rate by gradually decrease the learning rate from 0.5 to 0.01. I found out that the highest total reward will increase as the learning rate decreases to somewhere nearly 0.05. If the learning rate is too close to zero, the whole learning process will even fail.

(3):

I found out that the first few episodes have negative total rewards, and as the number of episodes it went through increases, the total rewards increases. And the total reward stops changing rapidly around 300 epsiodes.

The graph of the average of 10 trails shows similar pattern. However, the variance between episodes is much smaller. This happens because of the law of large numbers: as the number of trials we run increases, the smoother the learning curve will be.


400
350
300
250
200
150
100
50
0
-50
-100
Series1


400
350
300
250
200
150
100
50
0
-50
-100
Series1