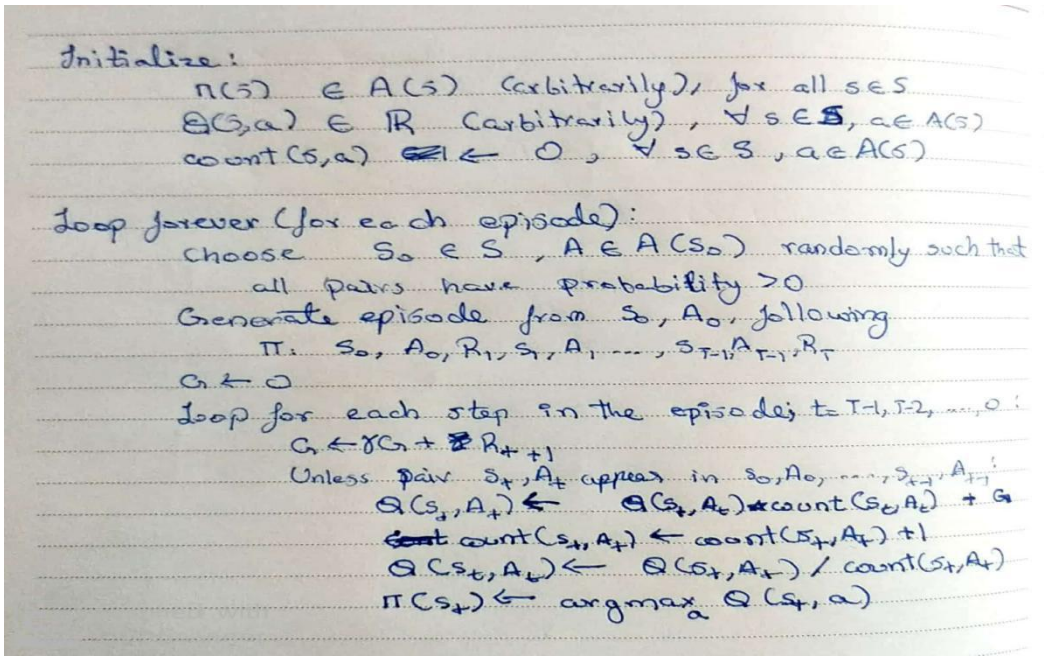
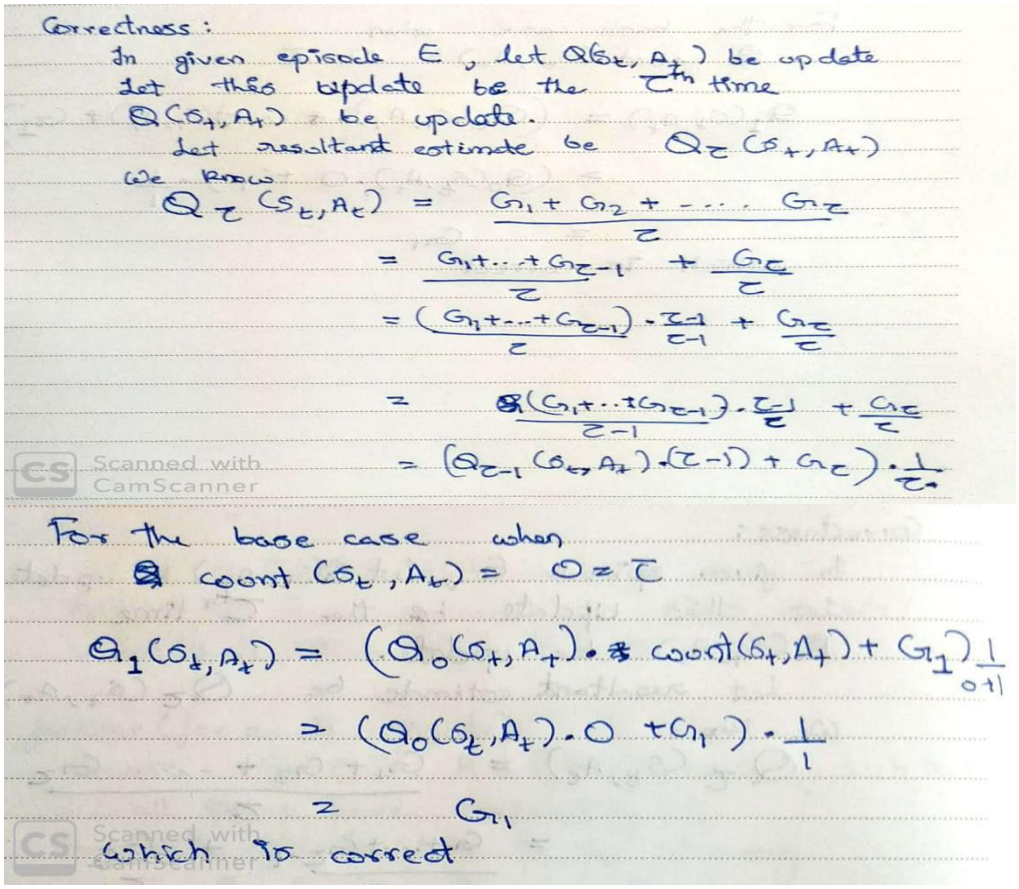
# Reinforcement Learning - Assignment 3

**Q1 -**
(Ex 5.4)
Pseudocode :




Justification :



In the $\tau^{th}$ episode before $Q(S_t, A_t)$ is updated,

$Q_{\tau-1}(S_t, A_t) \Rightarrow Q(S_t, A_t)$ ; $G_\tau = G_t$

$\tau - 1 = count(S_t, A_t)$

After update

$$Q_\tau(S_t, A_t) = \left(Q_{\tau-1}(S_t, A_t) * (\tau-1) + G_\tau\right) \cdot \frac{1}{\tau-1+1}$$

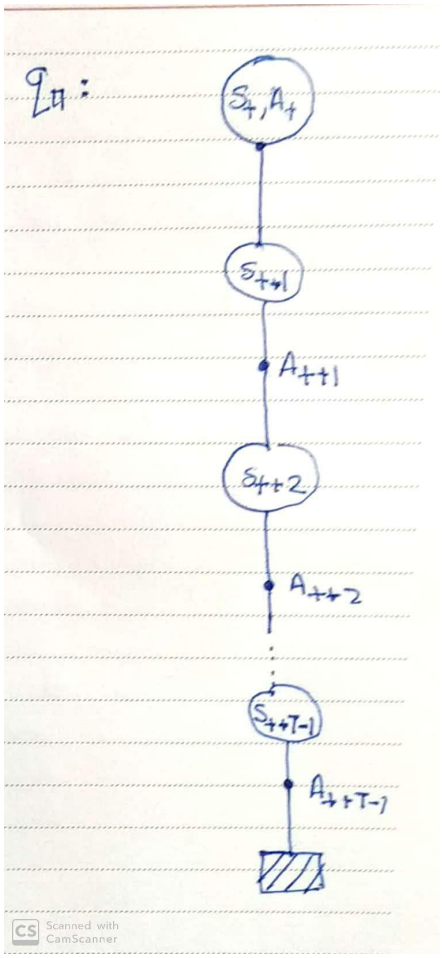
$$= \left(Q_{\tau-1}(S_t, A_t) * (\tau-1) + G_\tau\right) \cdot \frac{1}{\tau}$$

$count(S_t, A_t) = \tau$

Thus by induction, the pseudo code is correct.



**Q2 -**
(Ex 5.3)






**Q3 -**
(Ex 5.6)

$$Q(s,a) = \frac{\sum_{t \in \tau(s,a)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \tau(s,a)} \rho_{t:T(t)-1}}$$

$$= \frac{\sum_{t \in \tau(s,a)} \left(\prod_{k=t}^{T(t)-1} \frac{\pi(a_k|s_k)}{b(a_k|s_k)} \cdot G_t\right)}{\sum_{t \in \tau(s,a)} \prod_{k=t}^{T(t)-1} \frac{\pi(a_k|s_k)}{b(a_k|s_k)}}$$

But since we have picked $s_t$ and $a$ at the start

$\pi(A_t = a | S_t = s) = b(A_t = a | S_t = s) = 1$

$$\Rightarrow Q(s,a) = \frac{\sum_{t \in \tau(s,a)} \left(\prod_{k=t+1}^{T(t)-1} \frac{\pi(a_k|s_k)}{b(a_k|s_k)}\right) G_t}{\sum_{t \in \tau(s,a)} \prod_{k=t+1}^{T(t)-1} \frac{\pi(a_k|s_k)}{b(a_k|s_k)}}$$

$$\Rightarrow Q(s,a) = \frac{\sum_{t \in \tau(s,a)} \rho_{t+1:T(t)-1} G_t}{\sum_{t \in \tau(s,a)} \rho_{t+1:T(t)-1}}$$

**Q4 -**

Figure 5.1

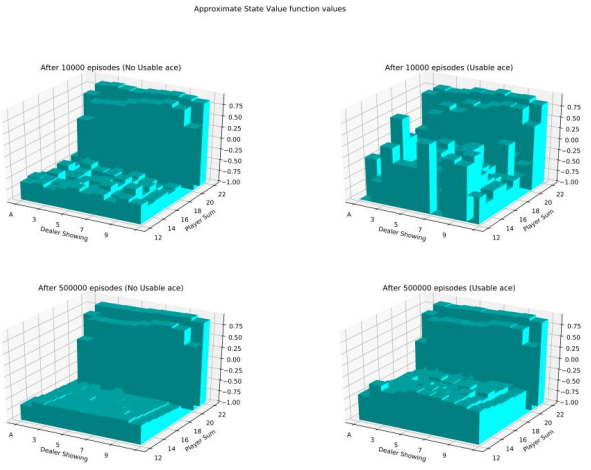
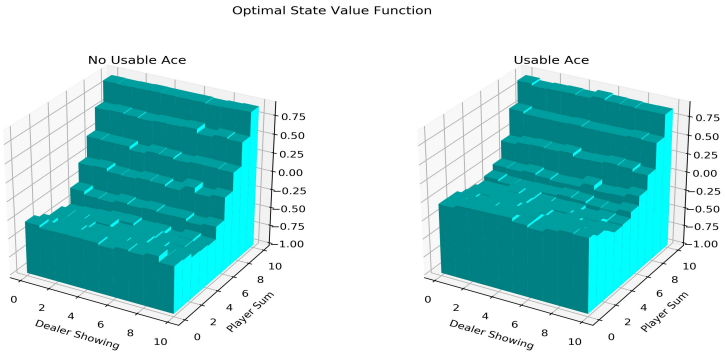



Figure 5.2 -

V* -->




Π* -->

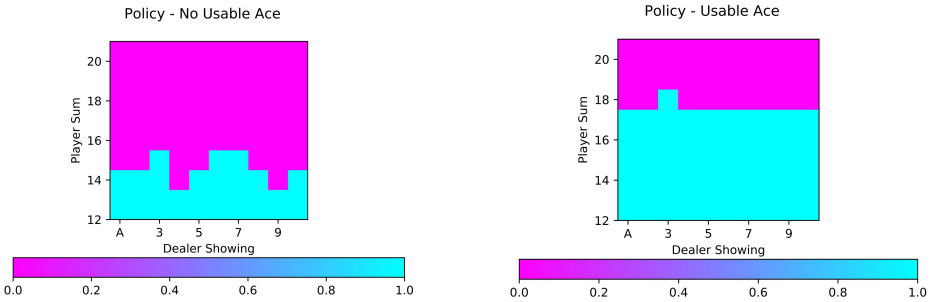
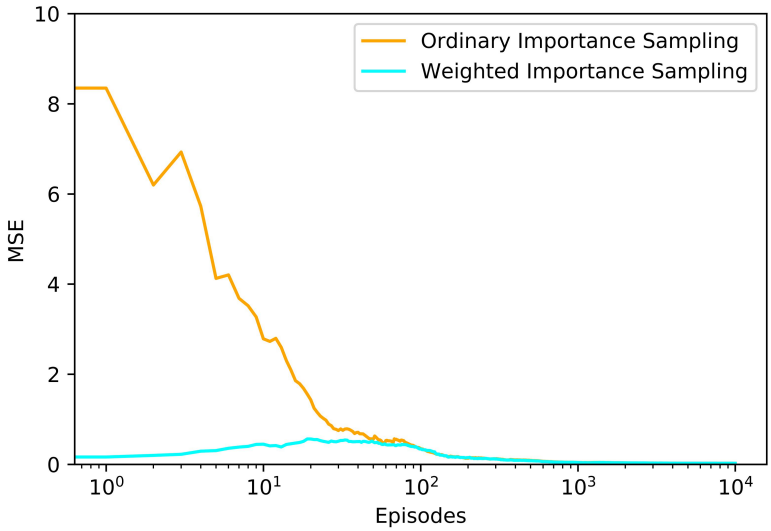


Figure 5.3 -



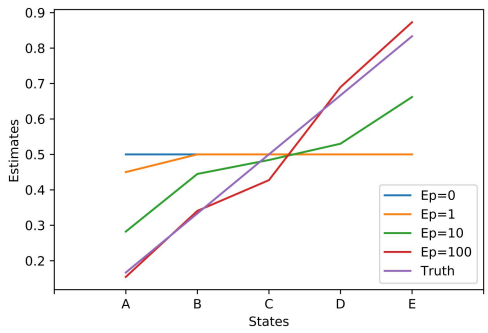

**Q5 -**

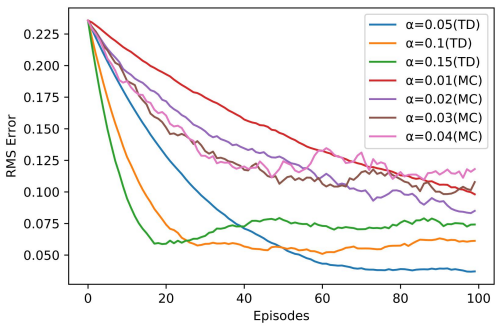
(Ex 6.2)
Consider the case a person has been driving home from work at location A for a long time. Thus the person has fairly accurate estimates of time to go from various positions in his journey home. Now he changes hos office location to B, such the route back home from B merges with route home from A at some point.
In this scenario using TD will help converge to accurate estimates faster for route from B since the common locations between the previous and current route have near optimal estimates. Thus the initial estimates of the new locations on route B will move towards a near stationary estimate of the known locations, enabling faster convergence. This is better compared to MC as the new locations will only end up learning from the end of episode returning, thus not harnessing the existing information present in the estimates of the common locations between the two routes

In the original scenario of the example in the book, we can see similar improvements while using TD if some of the states are initialized to near optimal estimates, enabling faster convergence as compared to MC.

**Q6 -**

(Example 6.2)



(Ex 6.3)
The initial estimate for each non terminal state is 0.5. From the plot we see that only V(A) changed[reduced].

Now analyzing the first episode -->
If an action is taken which changes the state from one non terminal state(X) to another non terminal state(Y), the change in the value function is as follows:
V(X) = V(X) + alpha*(R+gamma*V(Y)-V(X))
Here alpha = 0.1 ; gamma =1; V(Y) =V(X)
Thus the expression reduces to V(X) = V(X) + alpha*R
Now R = 0, thus V(X) remains unchanged

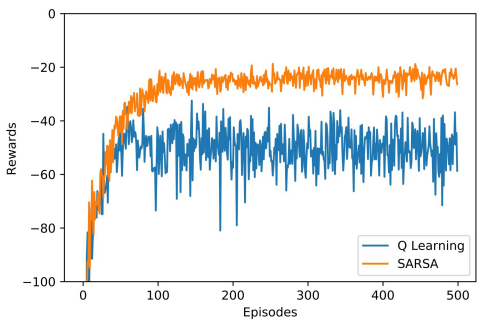
If an action is taken which changes the state from a non terminal state(X) to terminal state(T), the change in the value function is as follows:
V(X) = V(X) + alpha*(R+gamma*V(T)-V(X))
Here alpha = 0.1 ; gamma =1; V(T) = 0
Thus V(X) = V(X) + alpha*(R-V(X))
Now for V(X) to reduce after 1 episode R-V(X) must be negative, thus the reward from the given MRP must be 0.
Thus the final transition was from A --> adjoining terminal state (Reward = 0)
Thus V(A) <----- (1-alpha)*V(A) = 0.9*V(A)

Thus only V(A) value decreased as episode terminated on the left terminal state wrt C.

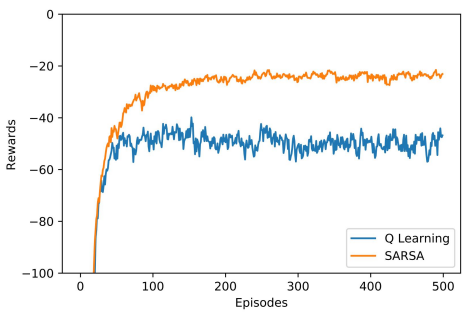
(Ex 6.4)
Altering the alpha value in either algorithms will only alter the importance given to the change required in the estimate at each step. A higher alpha value will cause more randomness as each update to the value function will be highly sensitive to the error observed(td error / mc error).
A smaller value will have a lower sensitivity to errors at each time step, resulting in a smoothly decreasing loss curve(vs episodes) albeit at a slower rate. Altering the value of alpha does not effect the inherent effectiveness of either algorithms it just helps balance between the sensitivity to noise and rate of improvement of the estimates.

(Ex 6.5)
The alpha value is a possible cause of the increasing TD error. After the estimates come sufficiently close to the optimal values, a high alpha could cause a change in the current value estimate which causes it to overshoot its optimal value. This continues to occur causing the estimate to diverge from the optimal value and leading to an increased RMS error. It can be observed that for smaller alpha this phenomenon occurs at a lower RMS error, which can be attributed to being closer to the optimal value. Thus in case of a constant alpha the smaller the alpha the closer it allows the estimate to reach towards the optimal value. This occurs always but as the alpha decreasing it becomes less perceivable.

**Q7 -**
Original Rewards -




Smoothed Rewards -



**Q8 -**

In Q-Learning if action selection also made greedy then both the target and behaviour policy become greedy. In the case of SARSA both these policies are epsilon greedy. The two variants are not the same because this new variant of Q-Learning will not have any exploration and thus wont explore every state. SARSA will not always choose the same action as this algorithm since it has epsilon greedy policies and will choose non greedy actions with a non zero policy. The same reasoning goes for weight updation. Thus this modified Q Learning algorithm is not the same as SARSA. It does not even guarantee convergence as its greedy policies can prevent it from exploring the entire state space, which is prerequisite for converging to an optimal value.