REPORT

PHD17008
Aradhya Neeraj Mathur

Q1.

NOTE: the pseudocode has been used to implement the question 4 5.1 , 5.2.

The pseudocode for question 4 is as follows

G += R(t+1)

Q(s, a) = (Q(s, a) * count(S,a) + G) / (count(S,a) + 1)

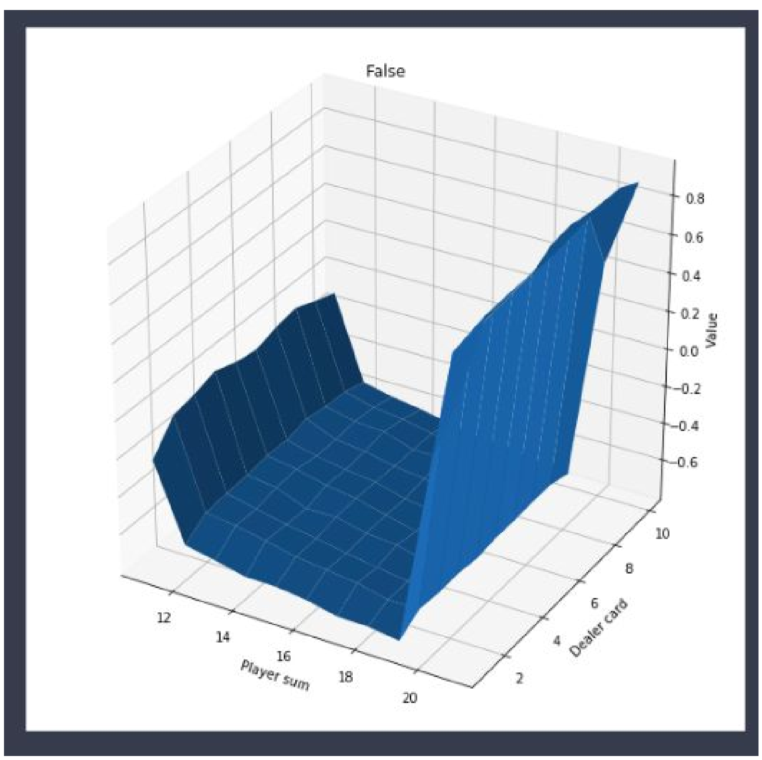
While the pseudocode in book maintains a list of the returns and then calculates the average my method keeps track of the count of each state action pair and the multiplies the older Q value with the old count and adds the current updated return and then divides by the new count which is essentially equivalent to the mean.

Q4 Value diagrams

Fig 5.1

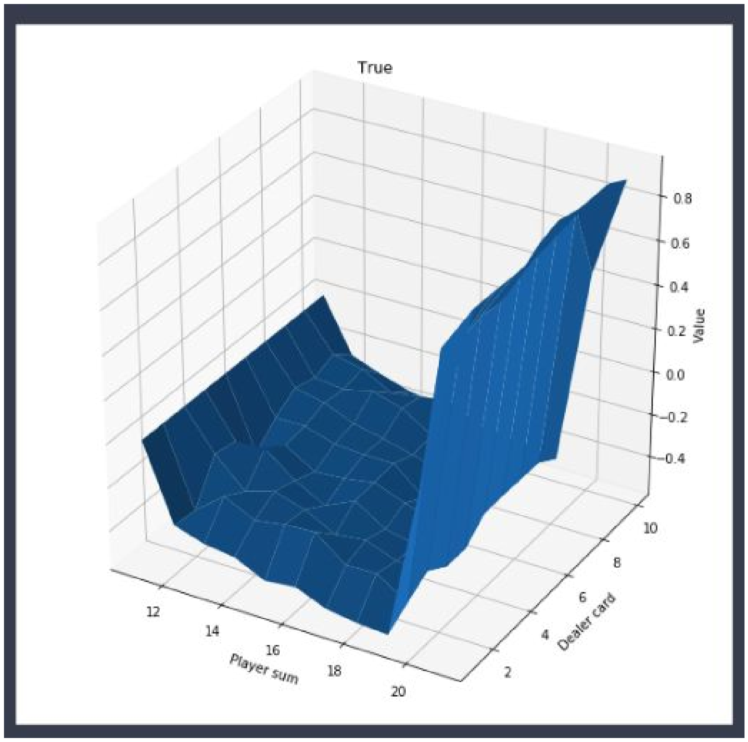
For 500000 episodes

Non usable Ace



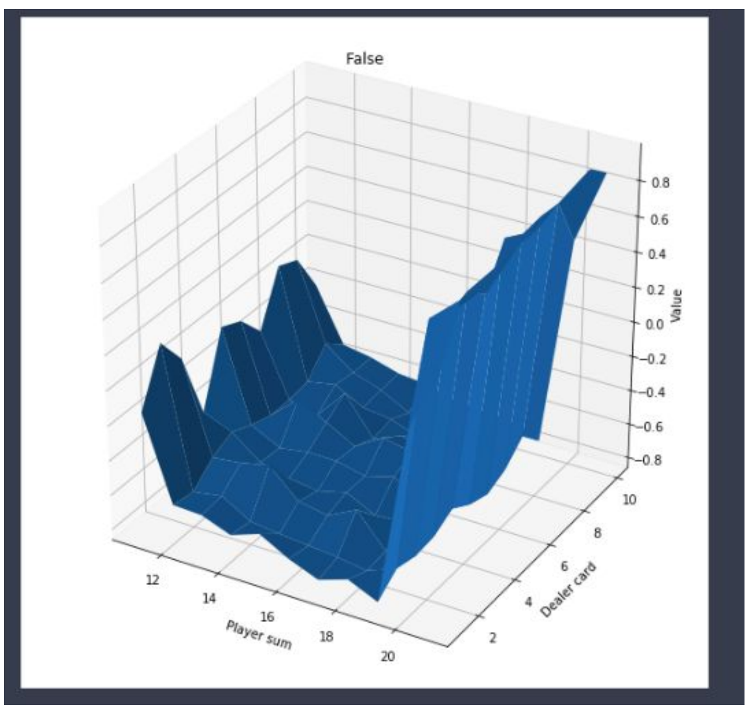
The code has been explained in the notebook

Usable Ace



For 10000 episodes

Non usable Ace



Usable Ace

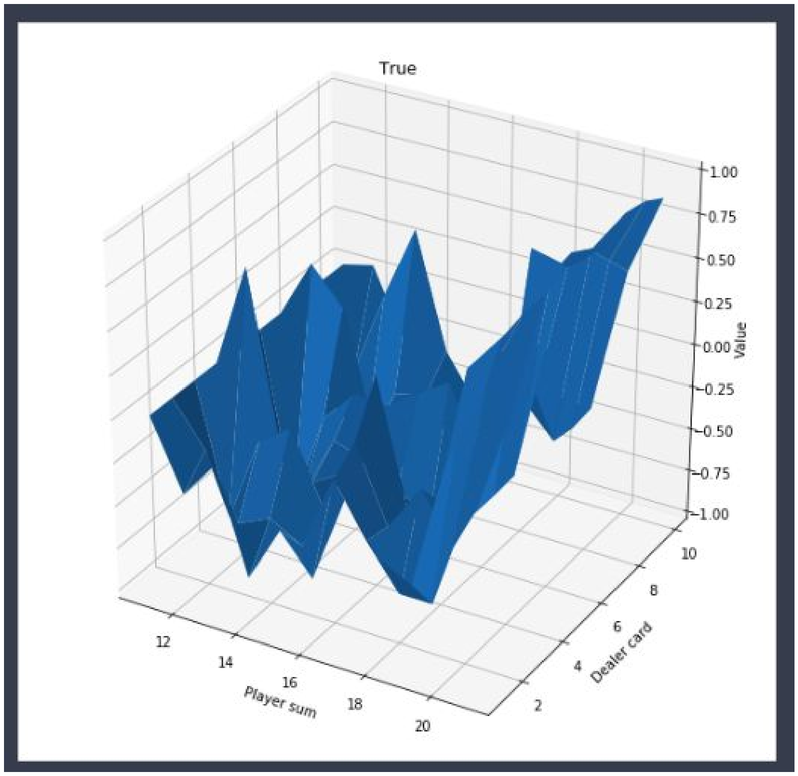
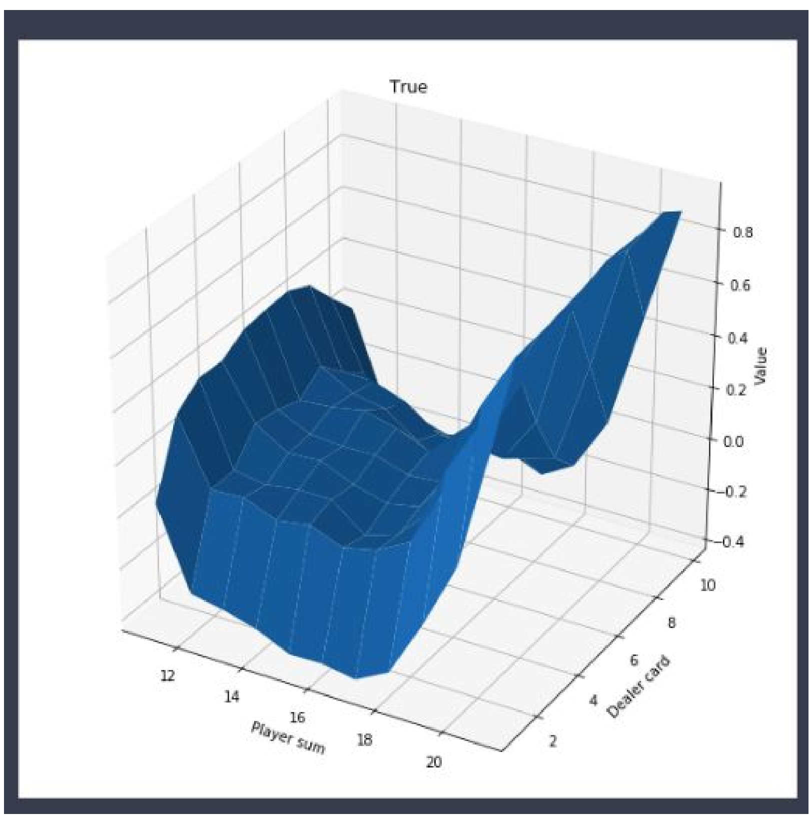
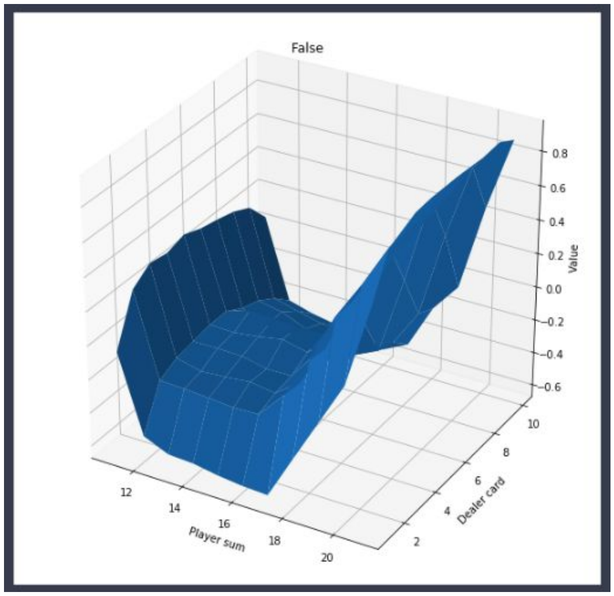


Fig 5.2

Usable Ace

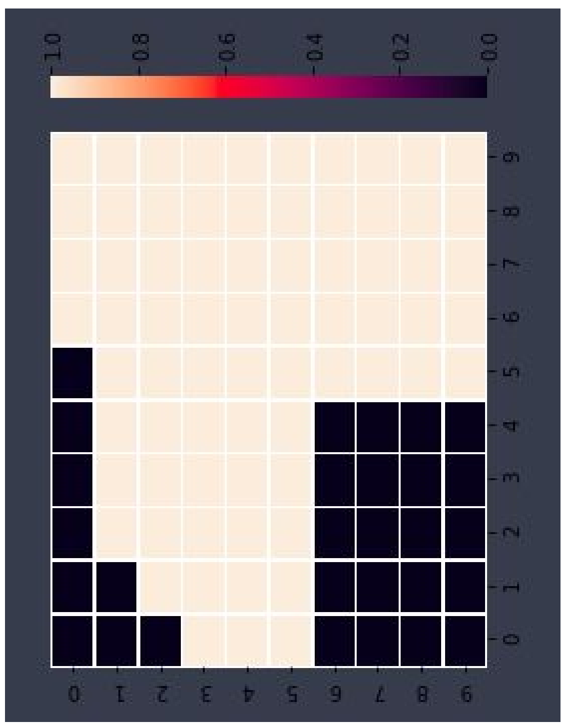


Non Usable Ace

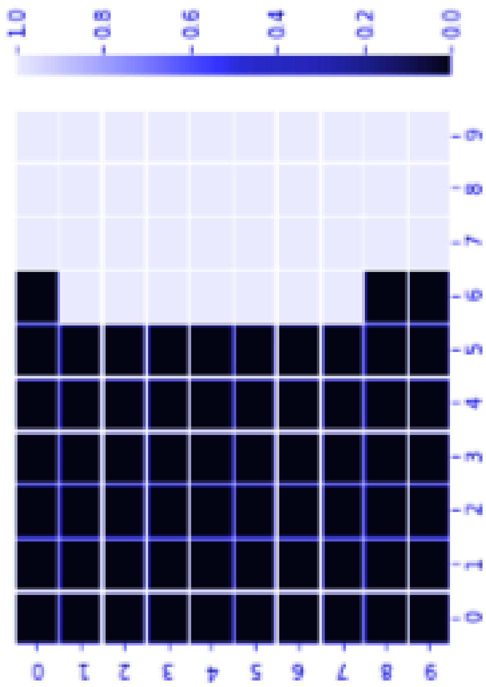


Policy diagram

Non Usable Ace



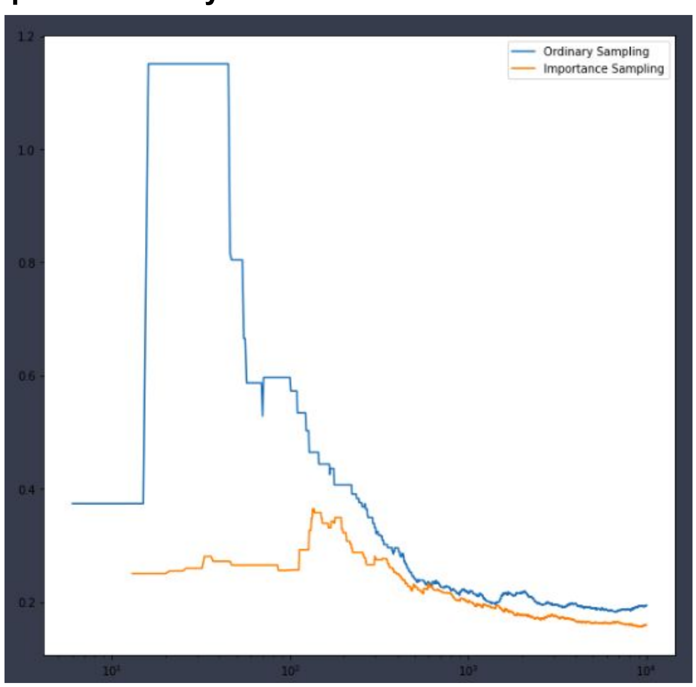
Non Usable Ace Policy Hit:0, Stick:1



Usable Ace Policy Hit:0, Stick:1

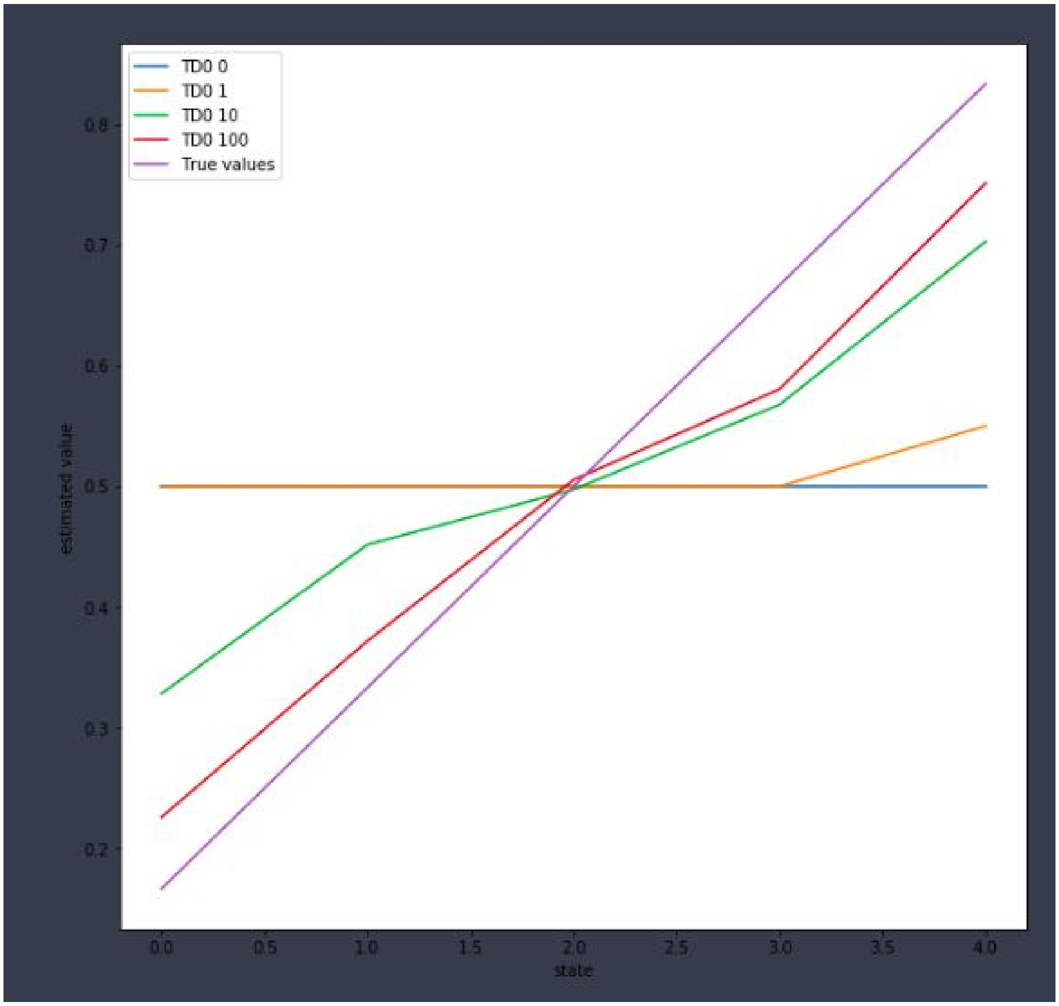
Fig 5.3

X axis : number of episodes y axis: MSE error in value

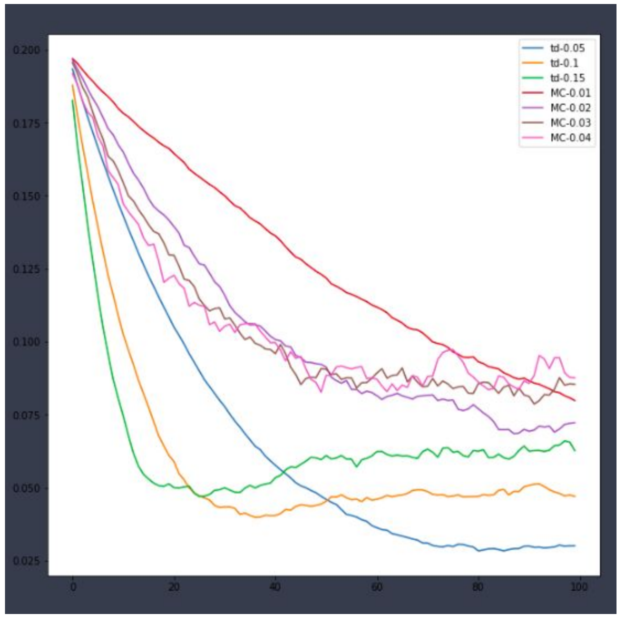


It can be observed from the graphs that the error of ordinary sampling is more than importance sampling.

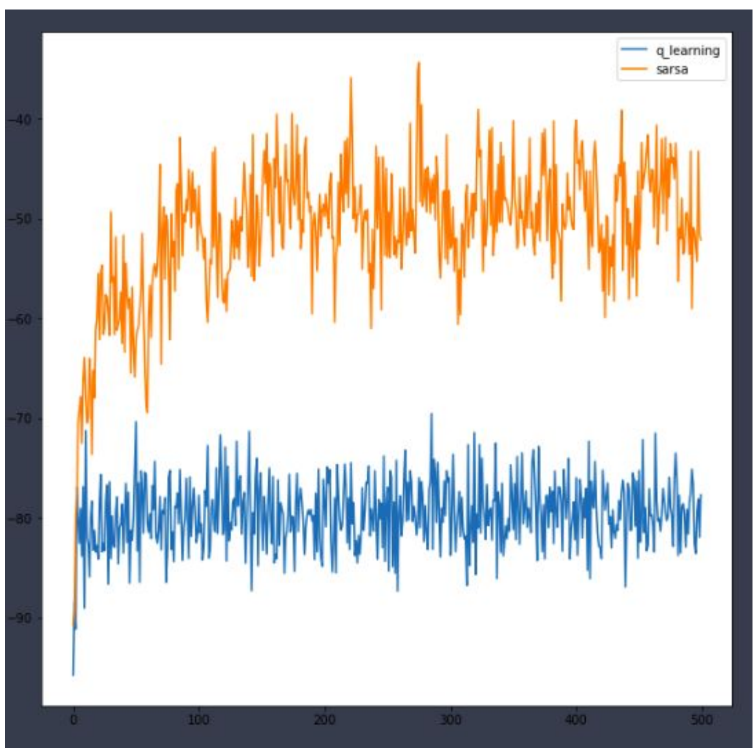
Q6.



Empirical RMS error,
averaged over states for TD0 and MC for different alpha values



Q7.



Q Learning Vs sarsa for sum of rewards y axis:sum of rewards x axis:episodes

Q3:
Ex. 5.6

Equation 5.6 calculates weighted importance sampling using the following eqⁿ.

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$

Equation analogous for $Q(s,a)$.

If we collect and cover the state action pairs in $\mathcal{T}(s,a)$ we can express $Q(s,a)$

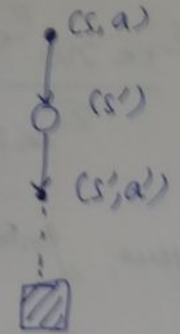
as $$Q(s,a) = \frac{\sum_{t \in \mathcal{T}(s,a)} \rho_{t+1:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s,a)} \rho_{t+1:T(t)-1}}$$

$$\boxed{\rho_{t+1:T(t)-1} = \prod_{i=t+1}^{T-1} \frac{\pi(A_i|S_i)}{b(A_i|S_i)}}$$

$T(t)$ is the first time of termination after time t and G_t represents returns from time $t+1$ till $T(t)$

Ex. 5.3.

Backup diagram for 5.3 for q_π is:



Starting with initialised 's' from sequence of episodes and action 'a' corresponding to state using policy π

Q1. Math for report explanation

$$Q_{n+1}(S,a) = \frac{1}{n}\left(\sum_{k=1}^{n} G_k\right) \Rightarrow \text{avg. of returns}$$

$$\therefore Q_{n+1}(S, a) = \frac{1}{n}\left[\sum_{k=1}^{n-1} G_k + G_n\right]$$

$$= \frac{1}{n}\left[\frac{(n-1)\sum_{k=1}^{n-1} G_k}{(n-1)} + \frac{1}{n} G_n\right]$$

$$= \frac{1}{n}\left[(n-1)Q_n + G_n\right] \rightarrow \text{Implemented in code}$$

$$Q_{n+1}(S,a) = Q_n(S,a) + \text{avg. (G returns)}$$

$$= Q_n(S,a) + \underbrace{\frac{1}{n}}_{\text{new count}}\left[\underbrace{(n-1)}_{\text{old count}}Q_n + \underbrace{G_n}_{\text{current return}}\right]$$

$\therefore$ We keep track of the visits and current return which is used to update the new $Q(S,a)$

6.3 ~~[illegible]~~

~~[illegible]~~ The TD method make the following update

~~$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$~~

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

we have $\gamma = 1$
$\alpha = 0.1$

In the first episode we only see a change the value of state A

Since $V(S) = 0.5$ for all $s \in S$
$V(s') = 0$ for s' in terminal state

now if in first episode we get reward 0 we end in the terminal state on the left.

$$V(A) = V(A) - 0.1(V(A)) = 0.9V(A)$$
$$\therefore V(A) = 0.9 \times 0.5 = 0.45$$
$$\text{difference} = 0.5 - 0.45$$
$$= 0.05$$

6.4&6.5

$$V(S_t) = V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

This α increased the weightage to the reward. The importance of each reward increases if a bigger value of α is used. Now as experimented in the notebook (plots can be found in the notebook) a smaller value of alpha leads to slower converge thus, if we keep the same number of episodes we get poor performance. Though it might lead to smoother convergence. Any other alpha values would not lead to better performance. If we increase alpha it will lead to more fluctuations. Similarly if α initialised to smaller values we

Q8

6.12

For ~~TD~~ 0 Q-learning

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$$

This modifies the state action value functions & then chooses the action as per update $Q(S, a)$.
~~Though~~ Whereas, in Sarsa we choose action as per previous, state value function and then update it. Both lead to different evaluations.

Q6.

6.2

In this case the part of bridge is common and only the initial route change. Assuming we use bootstrapping in TD, TD will ~~not~~ give a better performance as the state values of common states can be used which will mostly be common. This would give better convergence. Since there is less states that are arbitrary