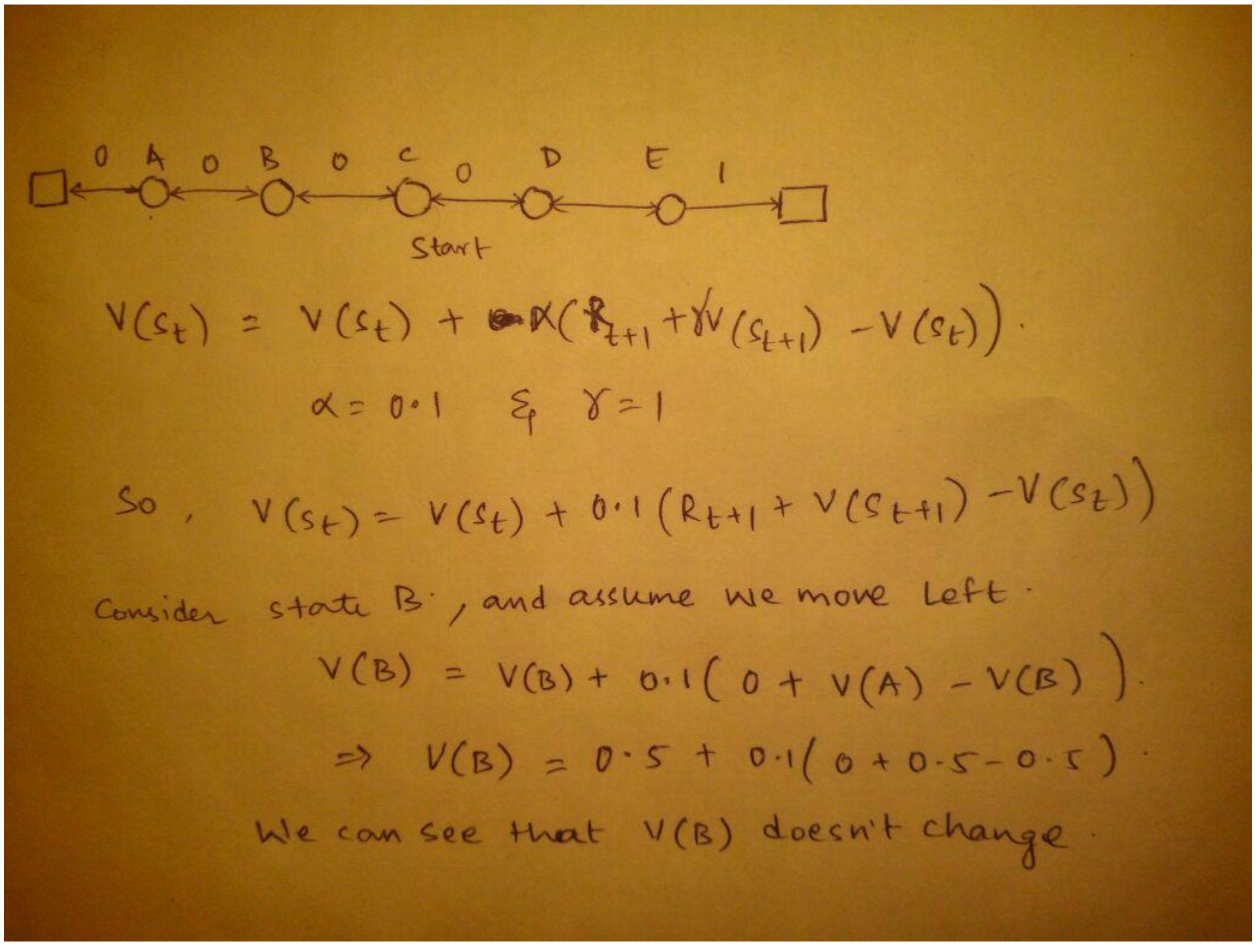
Assignment 3

1) The first episode results in a change in only V (A) tells that the first episode terminated on the left end with reward 0.


0
A
0
B
0
C
0
D
E
1
Start

$$V(s_t) = V(s_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(s_t)).$$

$\alpha = 0.1$ & $\gamma = 1$

So, $V(s_t) = V(s_t) + 0.1(R_{t+1} + V(S_{t+1}) - V(s_t))$

Consider state B, and assume we move left.

$$V(B) = V(B) + 0.1(0 + V(A) - V(B)).$$

$$\Rightarrow V(B) = 0.5 + 0.1(0 + 0.5 - 0.5).$$

We can see that V(B) doesn't change.

Similarly none of the values of other states change no matter how many times the agent moves around from one state to another. As we initially had all V's assigned to 0.5 and the reward is 0, we always get 0 increments.

But this changes when there is a terminal state, whose value is 0. Suppose see the calculation of $V(A)$ when we assume agent moves from A to left terminal state.

$$V(A) = V(A) + 0.1 * \big(0 + V(Left\ terminal\ state) - V(A)\big) = V(A) + 0.1 * (0 + 0 - 0.5)$$
$$V(A) = 0.5 + 0.1 * (0 + 0 - 0.5) = 0.5 - 0.05 = 0.45$$

If the agent had ended up in the right terminal state, $V(E)$ would have incremented by 0.05, since here we have reward of 1, which makes the increment term $0.1 * (1 + 0 - 0.5)$. So we can conclude that agent ends up in left terminal state at the end of first episode seeing the plot.

2) From figure 2.6 we could see that plot of RMS error averaged over all states vs alpha, looks like a bell shape. RMS increasing as we increase alpha, and decrease after a certain value of alpha. That should hold true here also. We can see that for the plot in figure 6.2. Consider MC for example, at alpha=0.1 the RMS error seems to be high, at 0.2 it gets pretty low, at 0.3 it increases slightly, and at 0.4 it increases quite a bit. Similarly for TD, at alpha=0.15 RMS is high, at 0.1 it becomes a bit low and at 0.05 it becomes even lower. If we further decrease alpha, RMS error is bound to increase.
   It seems alphas here were cleverly chosen where we can observe how RMS error first increases as alpha increase and then decrease after certain point. So we can say that even if we choose a wider range of alphas they would fall on the either left or right side of the bell curve, where the RMS error is bound to be higher than for the alphas chosen in the plot. So it is quite evident that TD performs better in this problem domain, for all alphas.

3) We can use the Bellman expectation equations or policy evaluation method to solve. Using the Bellman expectation equations we can arrive at the solution pretty quickly. As shown in the picture, the policy evaluation shows no sign of convergence anytime soon. So I guess Bellman expectation equations were used.

$$\begin{tikzcd}
T \ \square & A \arrow[l, "0"'] \arrow[r, leftrightarrow, "0"] & B \arrow[r, leftrightarrow, "0"] & C \arrow[r, leftrightarrow, "0"] & D \arrow[r, leftrightarrow, "0"] & E \arrow[r, "1"] & \square \ T'
\end{tikzcd}$$

$V(T) = 0$
$V(T') = 0$

$$V_\pi(s) = \sum_{a \in A} \pi(a/s) \left(R_s^a + \gamma \sum_{s' \in S} P(s,a,s') V_\pi(s') \right)$$

Take E for example

$$V_\pi(E) = \tfrac{1}{2}(1+0) + \tfrac{1}{2}(0 + V_\pi(D))$$

Similarly we can write equations for all other states and solve them easily.

To, double check the true values, Lets take D & E

$$V_\pi(E) = \tfrac{1}{2}(1+0) + \tfrac{1}{2}\left(0 + \tfrac{4}{6}\right)$$

$$= \tfrac{1}{2} + \tfrac{2}{6} = \tfrac{5}{6}.$$

For policy evaluation, Lets take all $V_\pi(s) = 0$ initially.

	A	B	C	D	E	
□	0	0	0	0	0	□
	0	0	0	0	0.5	
	0	0	0	0.25	0.5	
	0	0	0.125	0.25	0.5	

$\longrightarrow V_\pi(E) = \tfrac{1}{2}(1+0) + \tfrac{1}{2}(0+0) = 0.5$

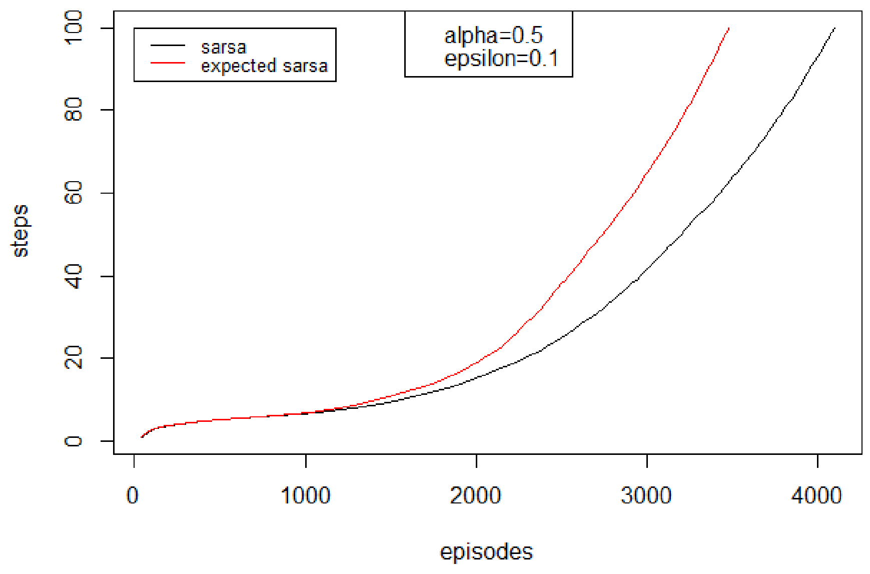
The others will have 0 values after first step

Second step

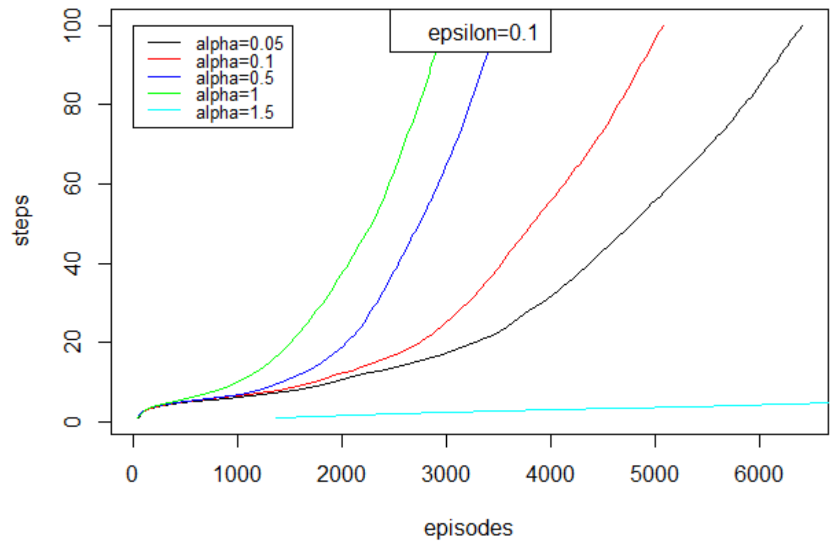
$V_\pi(E) = \tfrac{1}{2}(1+0) + \tfrac{1}{2}(0+0) = 0.5$

$V_\pi(D) = \tfrac{1}{2}(0+0) + \tfrac{1}{2}(0+0.5) = 0.25$

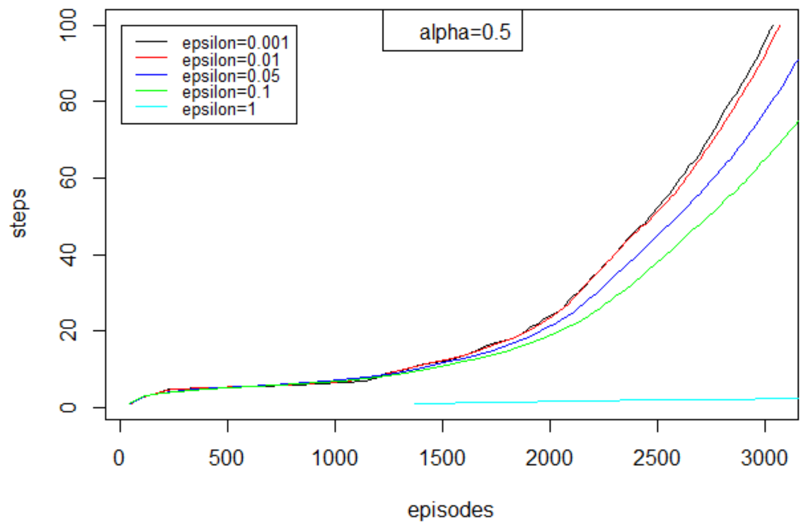
4) Code for Sarsa and Expected Sarsa is in their respective folders. The code can be directly run using make and make run. Once the code is run, it prints the average number of steps taken in each episode onto the console. Copy that and paste it in the $question4\_plot\_template.R$ file, for the respective n values. An example plot code is given in $question4\_eps0.1\_alpha0.5.R$ file, where epsilon = 0.1 and alpha =0.1. Here is how the plot looks.



Testing for different alpha while epsilon=0.1, I got a plot like this. This is consistent with the plot given in figure 2.6, as alpha increases performance increases and then after certain alpha, the performance decreases.



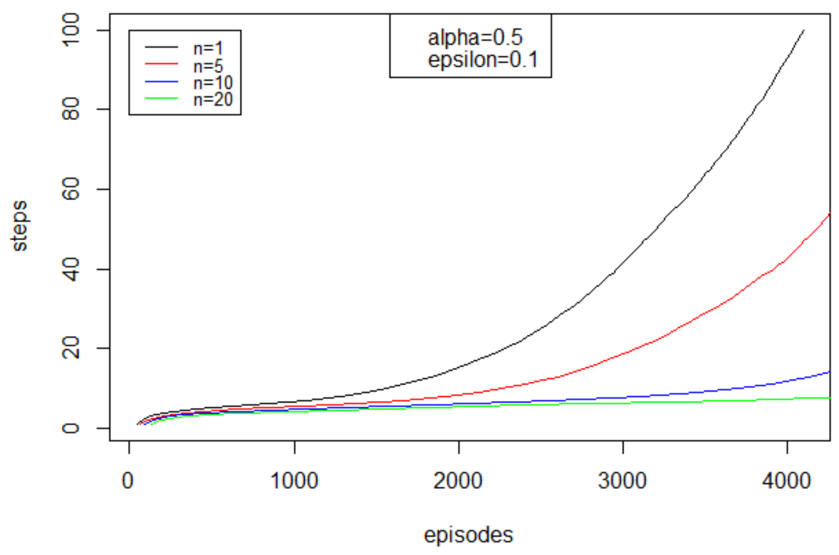
Similarly by keeping alpha=0.5 and changing epsilon, I get a plot like this. As epsilon increases performance seems to decrease and at large epsilon (at 1 for example in the plot, which is greedy) performance significantly decreases.



5) Q-learning is an off policy control because the action selection is done with respective to e-greedy selection, while the action value function update is done by maximizing over all those actions possible in the next state.

6) If we have 5 states, then for an equiprobable policy the average number of steps in an episode would be around 10 or something. So for this case there is no point in having more than 10 step methods. For a 19 states problem the average number of steps would be quite high, so it can be used to test for a large step methods.
The change of reward to -1 should not have any effect on best value of n. I think it would just make the convergence faster.

7) The code can be directly run using make and make run. Once the code is run, it prints the average number of steps taken in each episode onto the console. Copy that and paste it in the $question7\_plot\_template.R$ file, for the respective n values. An example plot code is given in $question7\_eps0.1\_alpha0.5.R$ file, where epsilon = 0.1 and alpha =0.1. Here is how the plot looks.



For the step values 5, 10, 20 decreasing the epsilon and alpha seem to have positive effect.