Q1:

(a) For the driving home example: since the entrance into the highway is the same, we just need to re-learn to reach the entrance into the highway from our new home/new parking lot. For TD method, it is able to do this, which is very efficient. For MC method, however, it needs to learn the complete journey before it can be finished. Therefore, in this case (when we only need to update the different initial conditions due to the current state and do not have to change the subsequent actions because we already have past experience), the TD method is more efficient than the MC method.

However, if it is in the original scenario (i.e., we do not have any relevant past experience with this scenario), the MC method has a greater advantage over the TD method. Because in this scenario, the TD method needs to run many episodes to learn the whole journey, while the MC method will do all the updates in only one episode.

(b) For blackjack, the MC method is more suitable than the TD method because the Monte Carlo method assumes an episodic environment, while blackjack happens to be an episode game. In blackjack, we can update the Q table and update our policy at the end of each episode using MC control, and we do not need to identify the misbehavior that led to the loss, which is TD method's advantage, to wait until the next time step to update the value estimates rather than wait until the end of the episode. Thus, it makes no sense to use TD method for this scenario.

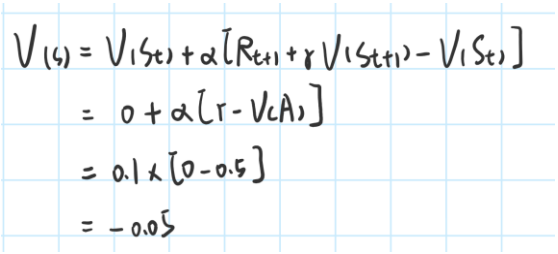
Q2:

(a) Although Q-learning follows the e-greedy policy, it will try to evaluate value function for 'greedy' policy. It directly takes the action that will lead to the greatest value at the next moment (but does not necessarily lead to the greatest value) rather than using e-greedy policy to update the action. Therefore, it is considered an off-policy control method.

(b) If action selection is greedy, then the SARSA will be the same algorithm as Q-Learning(using greedy policy), thus, they will make exactly the same action selection and weight updates.

Q3:

(a) The first episodes terminate on the extreme left. For all states, the approximate value function was initialized to the intermediate value V(s) = 0.5, and only V(A) changed, others stay the same.



(b) The specific results show in the right graph of the random walk example are affected by α, and the results finally converge to a relative stable value that won't change a lot anymore. From the graph, we could know that TD are always have better performance than MC (TD performs best when α = 0.05 and MC performs best when α = 0.01) even with different value of α, thus, the conclusion about which algorithm is better won't be affected by changing the range of values of α.

I don't think there is a fixed α value that would make either algorithm perform better, because the graph already shows almost the best results

(c) The different value of α may be one of the reasons that cause this. And it also could be a function of how approximate value function was initialized.

(d) Because in this chapter we need to compare the effect of different n values on the performance of the TD method. Thus, we need to set the n to a different value for investigating.

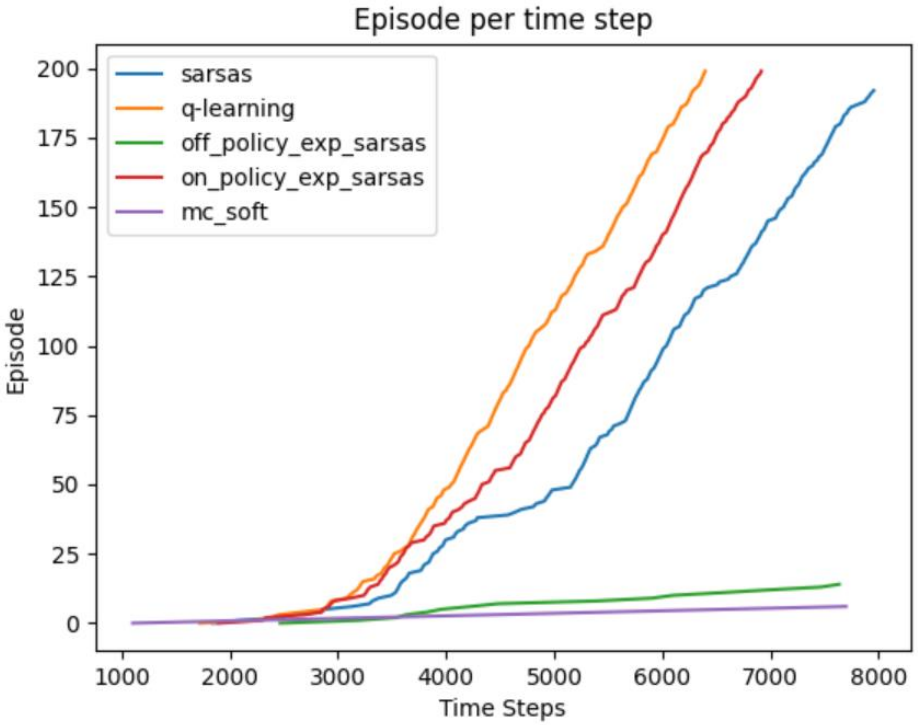
A smaller walk does affect the results regarding which value of n yields the best performance. For the reason different number of states will result in different environments.

It won’t make any difference in the best value of n. There is not much difference between the left-side reward being 0 and -1. As long as the reward on the right-side is greater than the left side, it will be fine.

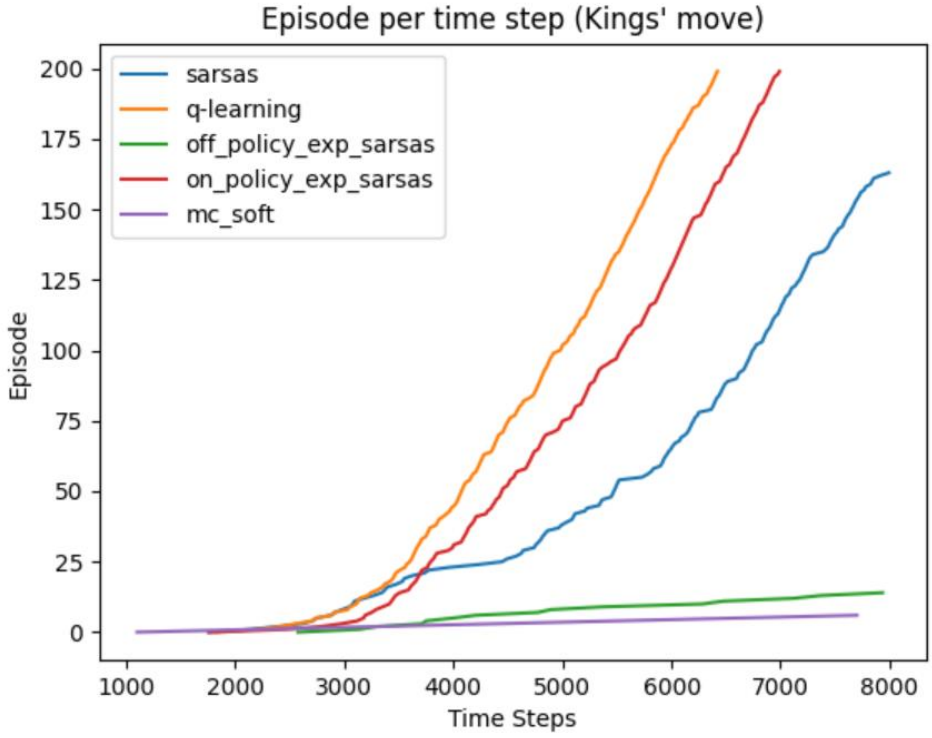
While state and transition remain the same, the best value of n won’t change.

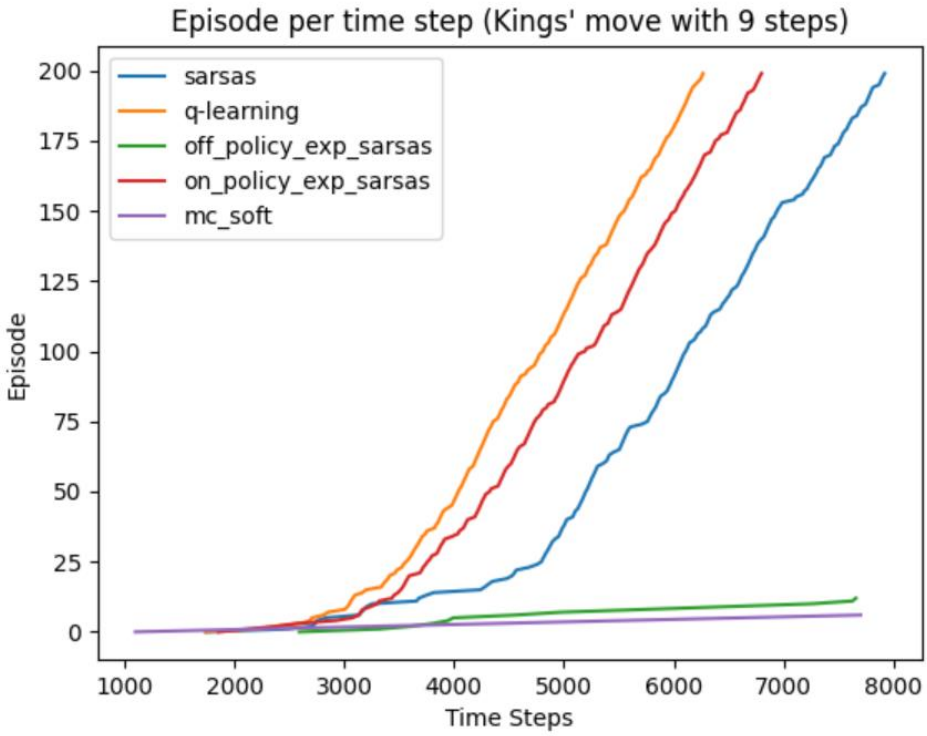
Q4:

(b)

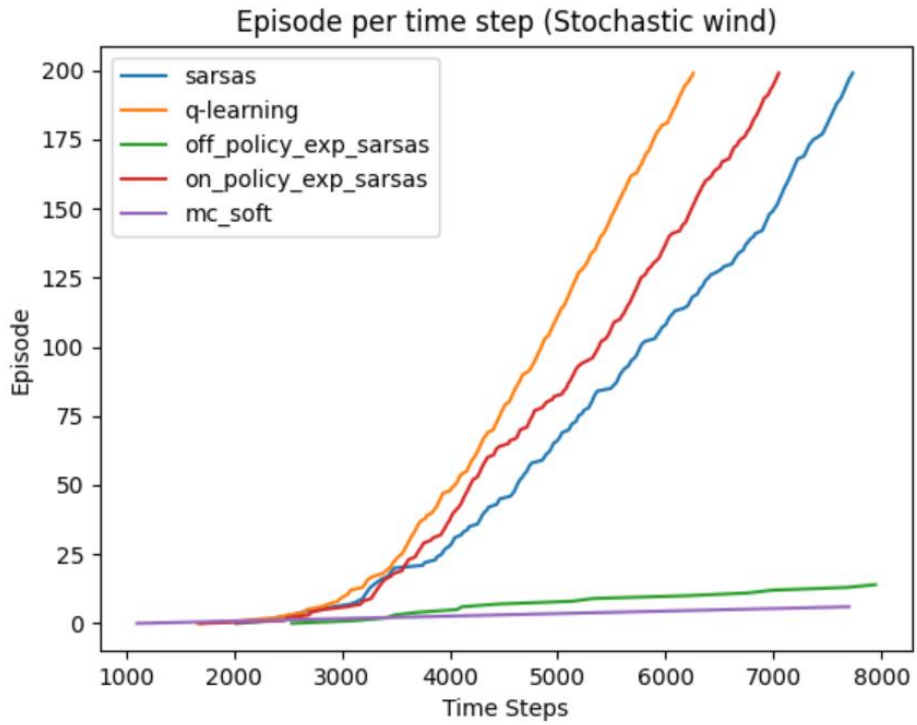

Episode per time step
sarsas
q-learning
off_policy_exp_sarsas
on_policy_exp_sarsas
mc_soft
Episode
Time Steps
0
25
50
75
100
125
150
175
200
1000
2000
3000
4000
5000
6000
7000
8000

(c)


Episode per time step (Kings' move)
sarsas
q-learning
off_policy_exp_sarsas
on_policy_exp_sarsas
mc_soft
Episode
Time Steps
0
25
50
75
100
125
150
175
200
1000
2000
3000
4000
5000
6000
7000
8000


Episode per time step (Kings' move with 9 steps)
sarsas
q-learning
off_policy_exp_sarsas
on_policy_exp_sarsas
mc_soft
Episode
Time Steps
0
25
50
75
100
125
150
175
200
1000
2000
3000
4000
5000
6000
7000
8000

(d)


Episode per time step (Stochastic wind)
sarsas
q-learning
off_policy_exp_sarsas
on_policy_exp_sarsas
mc_soft
Episode
Time Steps
0
25
50
75
100
125
150
175
200
1000
2000
3000
4000
5000
6000
7000
8000