Assignment 2

Zahra Parham 2122841

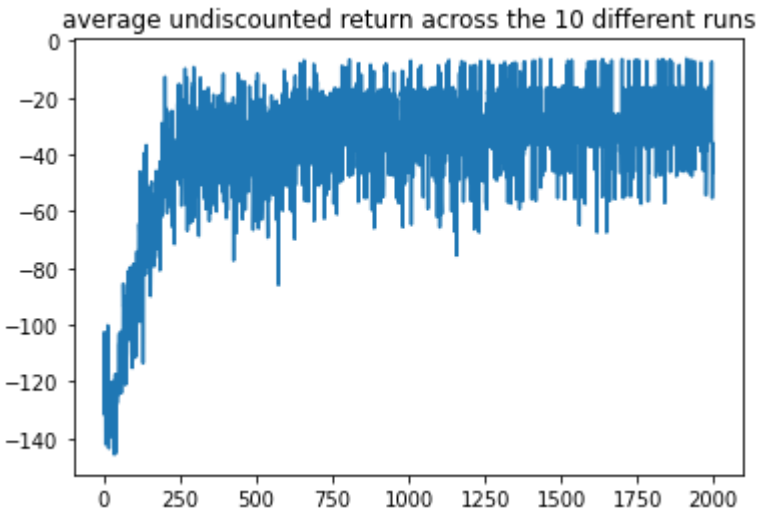
1.1

For this question, I define a function that gets policy, environment, render as input and give tuple: (state, action, reward) as output. The objective of this function is to generate episodes for frozrn_lake.

```
SFFF
FHFH
FFFH
HFFG
  (Left)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
  (Left)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
  (Up)
SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
[(0, 0, -1),
 (0, 2, -1),
 (1, 0, -1),
 (0, 2, -1),
 (1, 3, -1),
 (1, 1, -1),
 (5, 2, -100)]
```

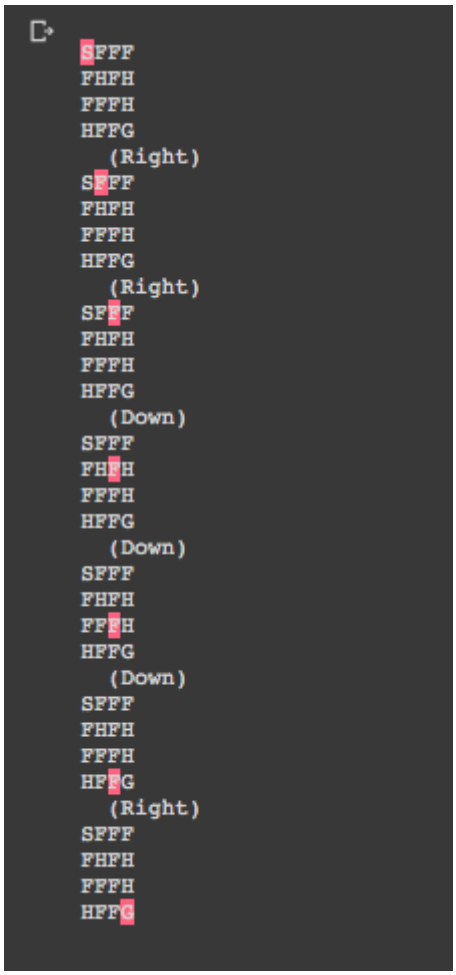
I generated one episode with the function

1.2.a



1.2.b

to verify the optimality, I just generated one episode with policy=7 that was obtained from the last question:



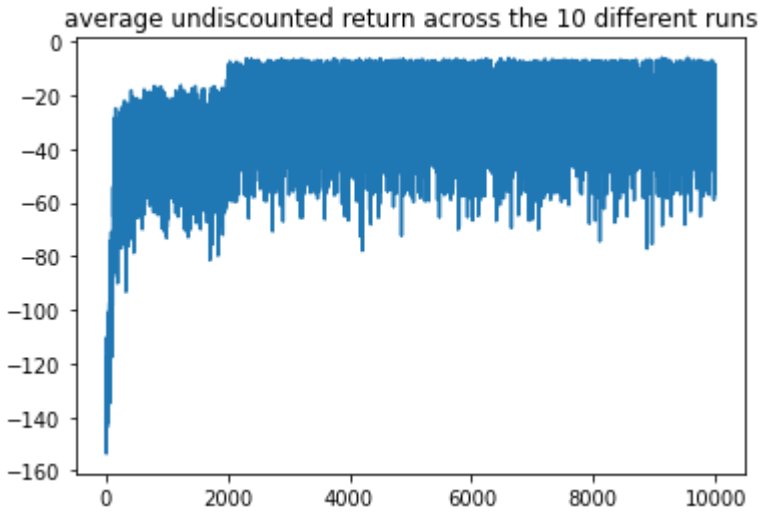
1.2.c

`average undiscounted return last 100 episodes over ten runs is:-27.25`

```
average undiscounted return last 100 episodes 10 runs is:-27.25
```

it is smaller than expected since the training is noisy, and the agent can still make inefficient action choices. Also, the environment has split_rate = 0.1, so even if the agent is trained well, it is a 10 percent possibility the agent would not reach where it decided since the floor is slippery. And also, on the other hand, the number of episodes for training(2000 episodes) it's not enough.

I just run the algorithm 10_000 times instead of 2_000, it converges better, but the average undiscounted return last 100 episodes for ten runs remain the same:

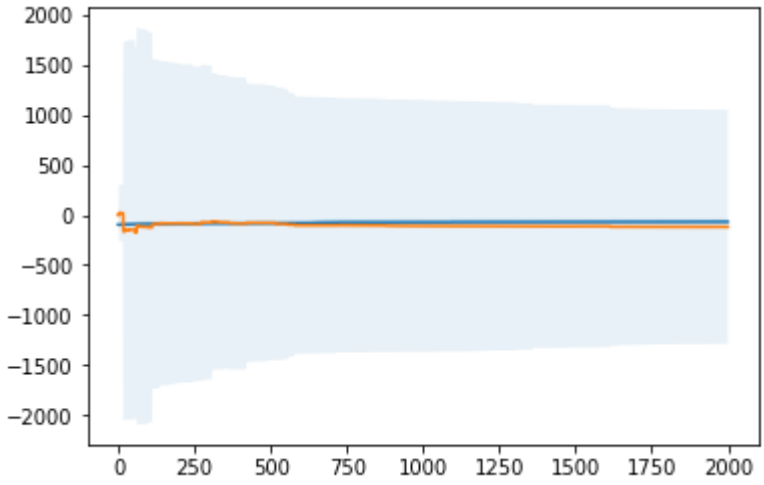



```
average undiscounted return last 100 episodes over 10 runs is:-26.423000000000002
```

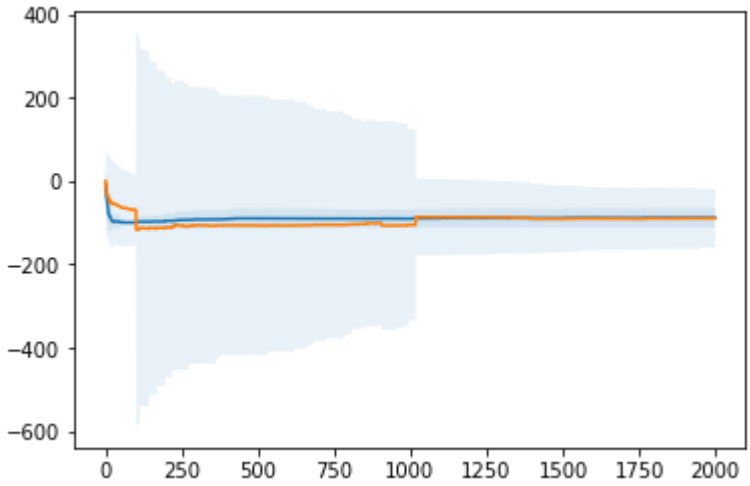
1.3.a

I choose the policy=7 was obtained from 1.2 for evaluating:

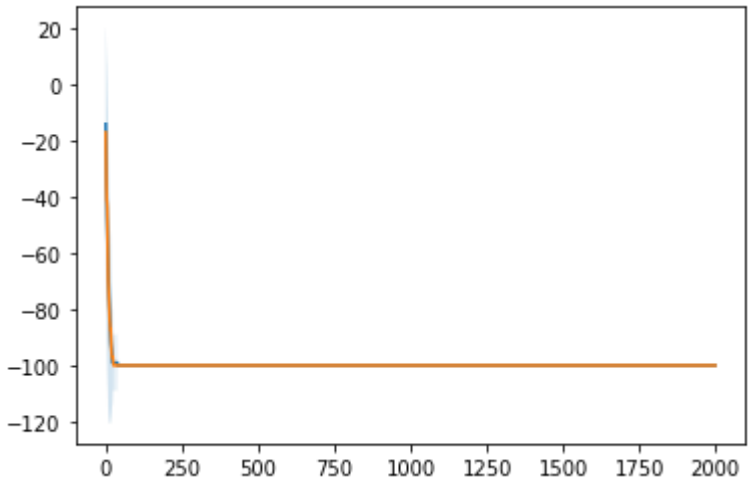
S=0(orange: WIS, Blue: OIS)



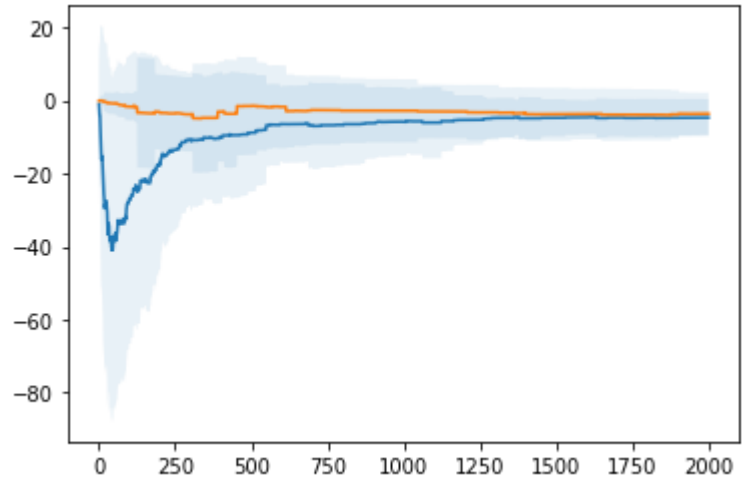
S=3(the orange is weighted importance sampling and the blue is OIS)



S=7(the orange is WIS and blue is OIS)



S=14 (Orange: WIS, Blue: OIS)



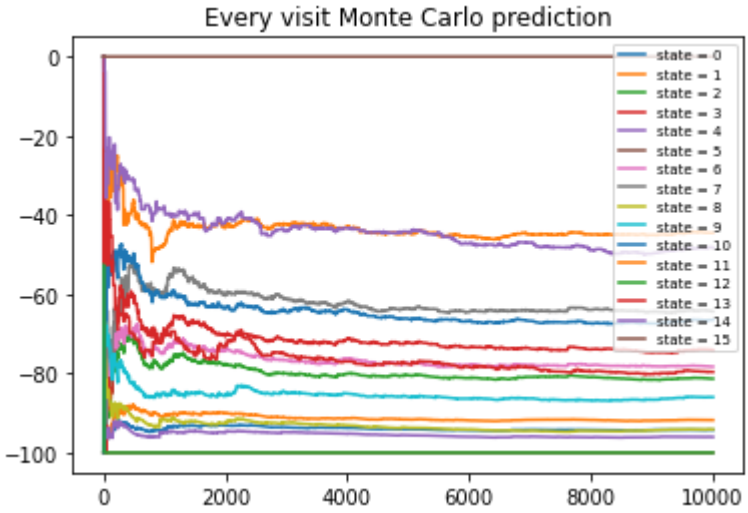
1.3.b

I expect that weighted Importance Sampling produces a lower variance; we can even observe it through the picture from the last part. The difference between OIS and WIS is displayed in their biases and variances. OIS is unbiased, and WIS is biased. To show this, consider the estimates of their first-visit methods after observing a single return from state s. The first visit of OIS is always V_pi(s), and it is unbiased, but it can be extreme; on the other hand, WIS always it's biased.

"On the other hand, the variance of ordinary importance sampling
is in general unbounded because the variance of the ratios can be unbounded, whereas in
the weighted estimator, the largest weight on any single return is one. Assuming bounded returns, the variance of the weighted importance-sampling estimator converges
to zero even if the variance of the ratios themselves is infinite" (Precup, Sutton, and Dasgupta 2001) [1]

[1] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

2.1.a

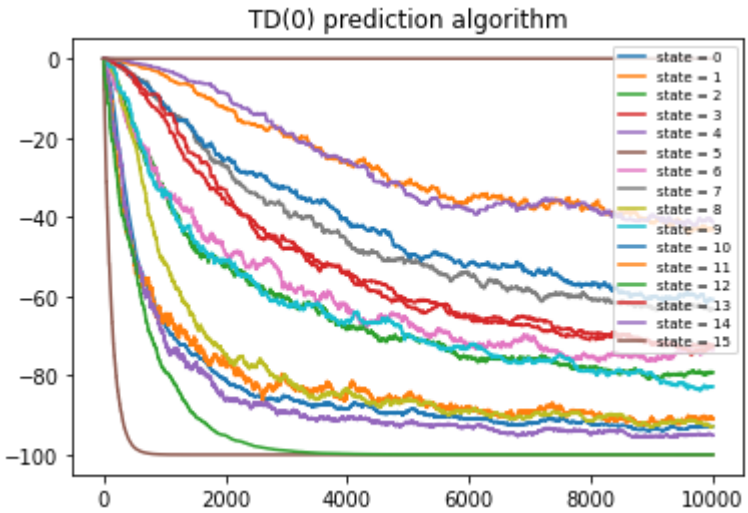


2.1.b

S -94.27	F -92.27	F -82.22	F -74.14
F -96.26	H -100.0	F -78.81	F -64.22
F -93.71	F -84.17	F -64.57	F -44.58
H -100.0	F -75.47	F -47.34	G 0.0

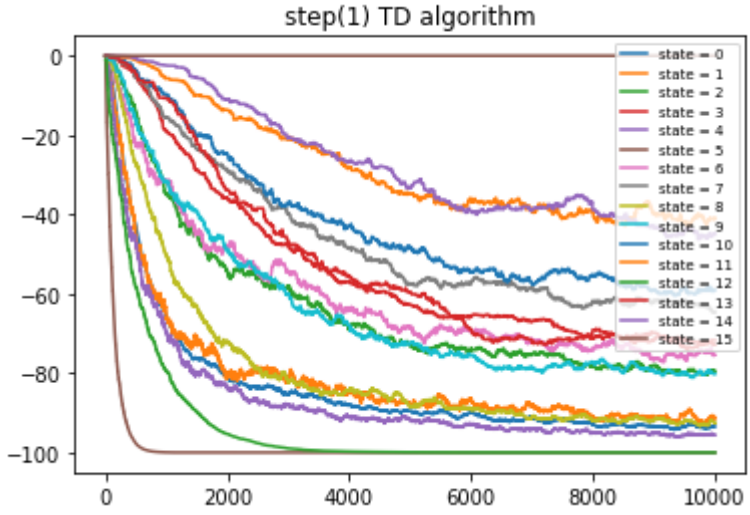
Do these values make sense? Yes, it does. The policy we choose is a random policy that is not optimal. Also, the probability for all the actions in all the states is the same (0.25), so for both forms of H, its value is -100 since it does not matter whether the agent chooses which action anymore; it already lost the game. And for the other states, we can observe a higher value function based on their closeness to the final state (G). Cause there is an action in these states(nearest to G state) that will win if the agent chooses them. And also, we can observe the lowest value function around the H state since there is an action that will lose the game if the agent decides it. So intuitively, it makes sense.

2.2



S -92.84	F -90.88	F -79.28	F -72.32
F -95.11	H -100.0	F -73.41	F -63.29
F -92.79	F -82.75	F -60.7	F -43.25
H -100.0	F -72.78	F -41.3	G 0.0

2.3.a



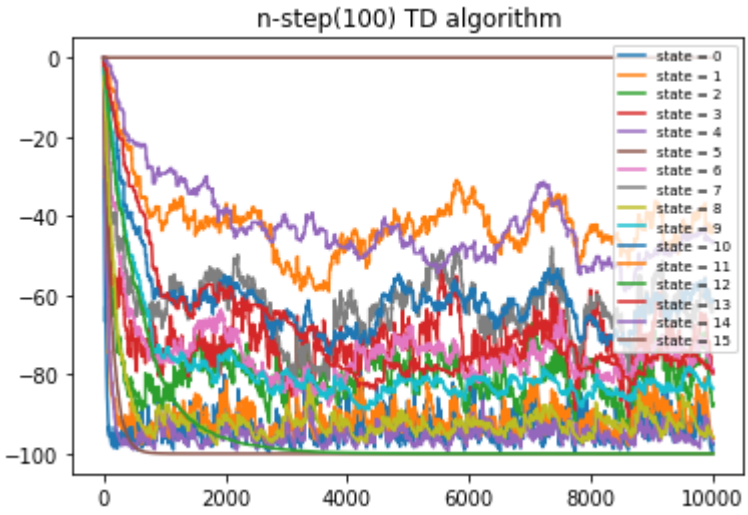
S -93.41	F -91.22	F -79.62	F -72.83
F -95.5	H -100.0	F -75.42	F -64.62
F -92.48	F -80.19	F -59.02	F -40.87
H -100.0	F -71.68	F -44.79	G 0.0

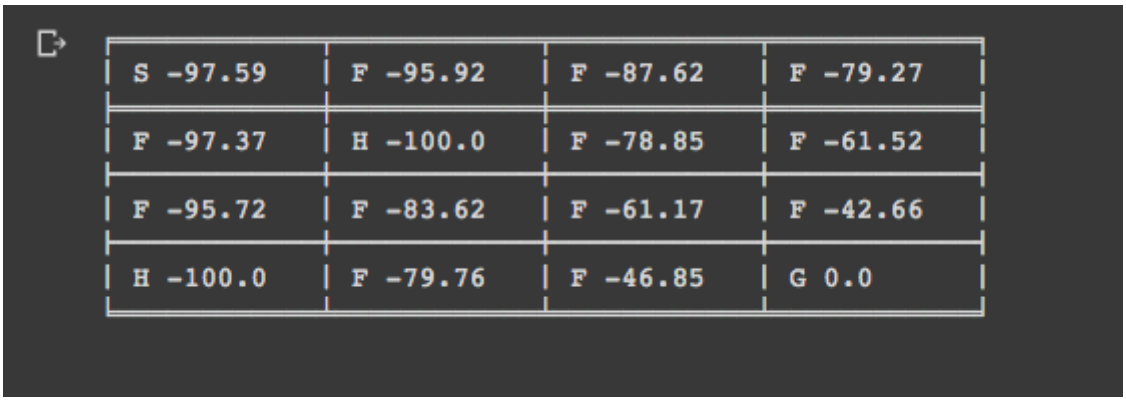
2.3.b

I expected this plot to be similar to TD(0) cause TD(0) will look ahead one step. TD(0) is a particular case of TD(n), and in n-step TD learning, if we consider n to be 1, it means that the TD algorithm looks ahead one step, so it's evident that these two will be similar. In Monte Carlo, we play a whole episode until the end; then, we compute the discounted reward for each state that appeared in the episode. Hence, I

expected that this plot would not be the same or not similar to every visit Monte Carlo and the result confirms my expectation as well.

2.4.a





2.4.b

it does not look similar to every visit Monte Carlo! We may assume that it make it identical to MC cause it looks ahead n-step(here it is 100 steps), but we average the discounted reward in every visit MC. However, in n-step TD, they still change an earlier estimate based on how it differs from a later assessment, and instead of getting an average, we have a time step alpha.

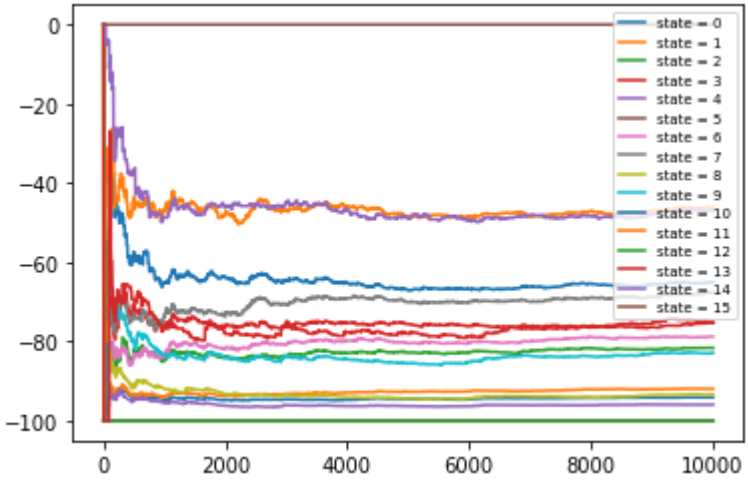
2.5.a

$$V[s] = V[s] + \alpha * (G - V[s])$$

$$\alpha = \frac{1}{|\ \tau(s)\ |}$$

$\tau(s)$ will be set of all time steps when state 's' was visited.

2.5.b



S -94.21	F -92.24	F -81.36	F -73.83
F -96.07	H -100.0	F -78.38	F -66.1
F -93.88	F -86.07	F -65.94	F -44.01
H -100.0	F -79.53	F -47.87	G 0.0

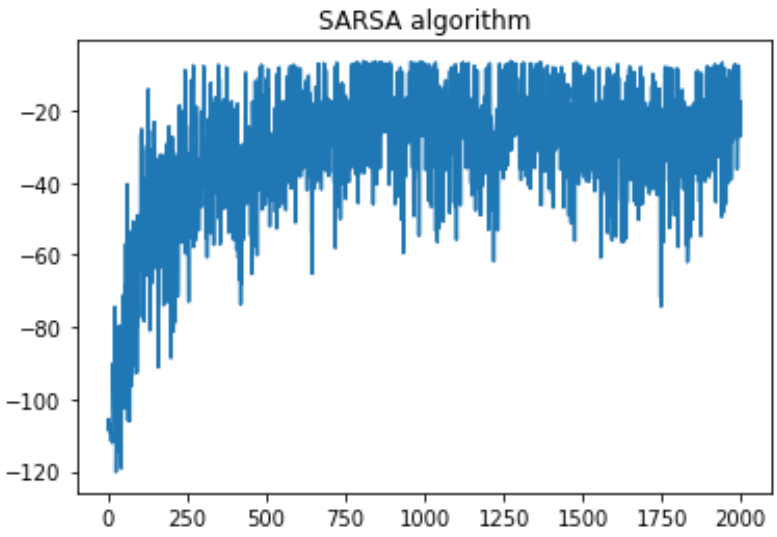
2.5.c

Yes it matches the performance of every visit Monte Carlo since the two differences between monte Carlo and n-step TD learning is that the monte Carlo looks ahead the entire episode for estimating the value function. Here in step TD learning, we consider n=100, which means we can assume that we approximately look ahead to the entire episodes to update the value function. And the second difference between these two algorithms is that in Monte Carlo, we average over the discounted reward for modifying the value function for state S. In TD learning, we have changed the time step named alpha to be the number of times that the agent modifying the value function for state S, so it means that it gets average over all the discounted reward it

is seen through entire episodes which is precisely the concept in every visit Monte Carlo.

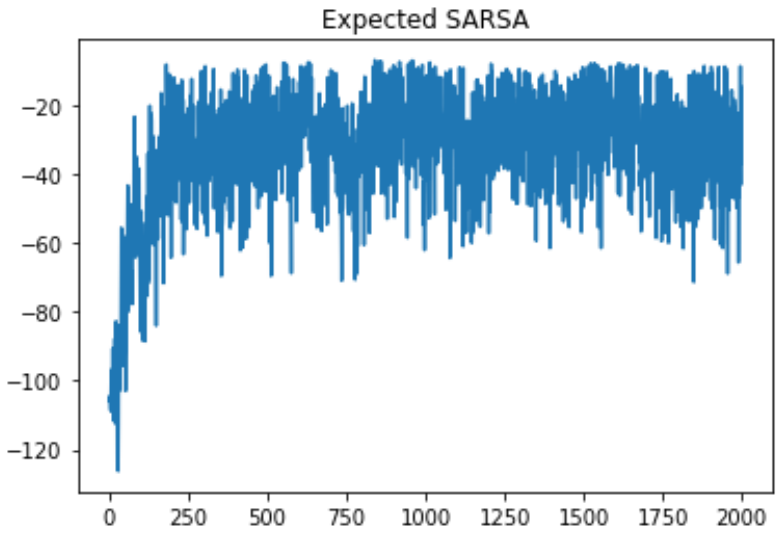
3.1

```
SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
```

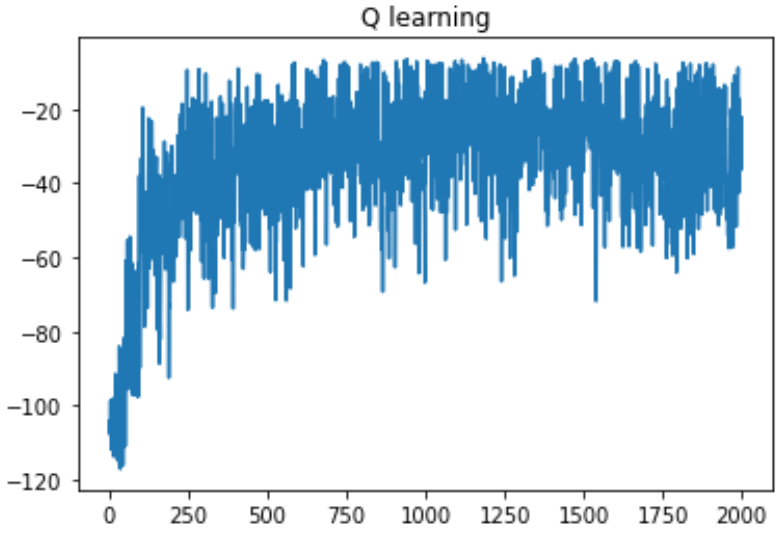


3.2

```
SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
```



3.3



3.4

The difference in updating the Q-value:

both TD learning and Monte Carlo aimed to update the Q value function:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha(X - Q(S_t, A_t))$$

for the Monte Carlo technique, the value of X will be updated by looking ahead the entire episode and computing the average discounted reward from a sample trajectory.

For temporal difference, the value of X is estimated by taking one step(n step) of sampled reward and bootstrapping the estimate by using the Bellman equation.

Bias:

For Monte Carlo, updating X is a direct sample of all undiscounted rewards, so it's the same as the expected value so it is not biased.

The TD learning is biased since the bootstrap value of X is initially whatever you set it to. So it's not exactly the actual value you are looking for, so it's biased, but

the bias decays exponentially over time.

Variance:

based on the explanation of Bias, we can infer that the Monte Carlo has high variance and TD has lower variance. Also, based on the plot, we have Monte Carlo has high variance compared to TD learning but n-step TD learning for n=100 has the highest variance among all these plots.

Convergence rate:

based on the plot that we draw, we can infer that the convergence rate of Monte Carlo is better than TD learning, and it converges significantly faster than TD learning.

Final Performance:

The performance of Monte Carlo is better than TD learning since it's unbiased and looks ahead through the end of the episodes so that the final performance would be more accurate than TD learning; we can observe this even though the plot that we have.