CSE 546 Reinforcement Learning Checkpoint 1 Naman Tejaswi

1. Describe the deterministic and stochastic environments, which were defined (set of actions/states/rewards, main objective, etc).

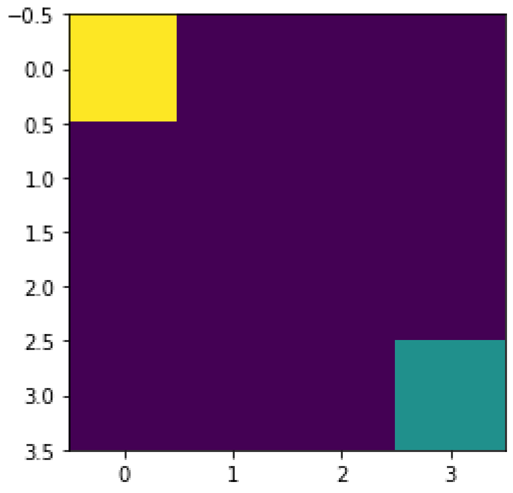
- Deterministic Environments

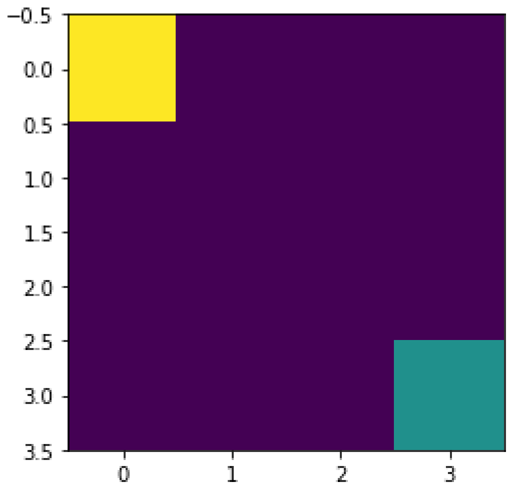
  1. Actions- Move up ,Move Down, Move Left, Move Right by 1 incremental step done by manipulating x and y coordinates
  2. States - Location in the grid from [0][0] to [3][3] inclusive
  3. Rewards- Zero -0.5 +0.85 and 1
  4. Main objectives reach the rightmost bottom corner.

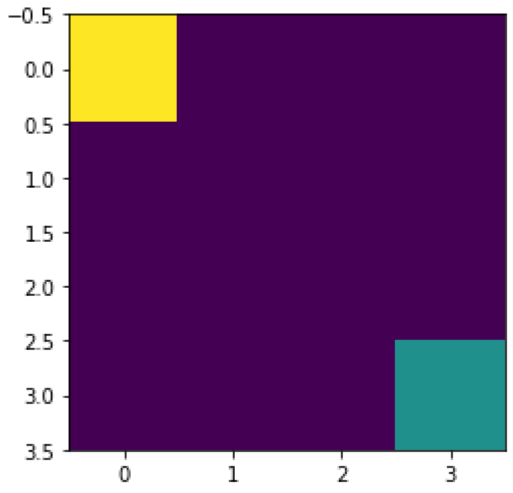
- Stochastic Environment

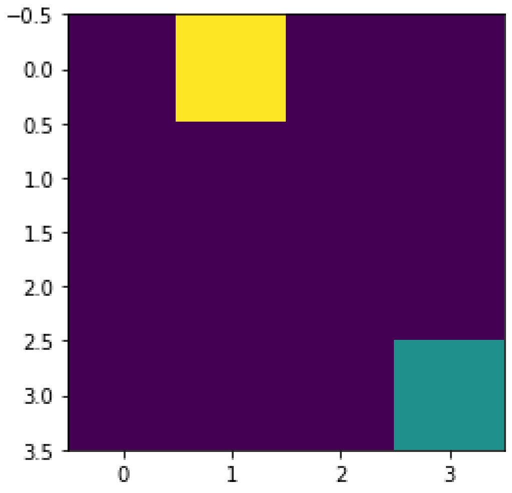
  1. Actions- Move up ,Move Down, Move Left, Move Right by 1 incremental step done by manipulating x and y coordinates using a probability that the agent moves in the designated direction 60% of time as we multiply by -1 if we get randint less than 0.4
  2. States - Location in the grid from [0][0] to [3][3] inclusive
  3. Rewards- -0.5 +0.85 and 1 but the probability of our action changes in a stochastic environment leading to only a 60% conversion rate
  4. Main objectives reach the rightmost bottom corner.

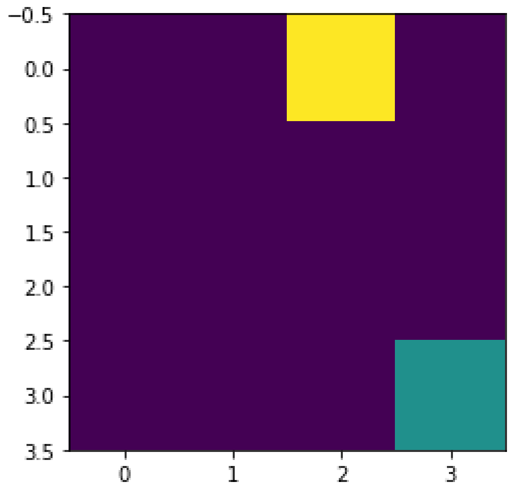
2. Provide visualizations of your environments.

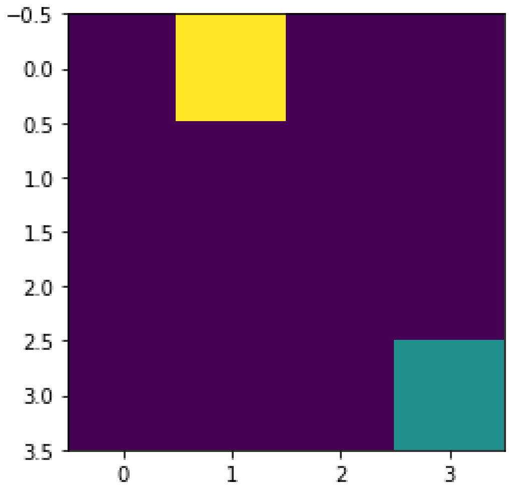


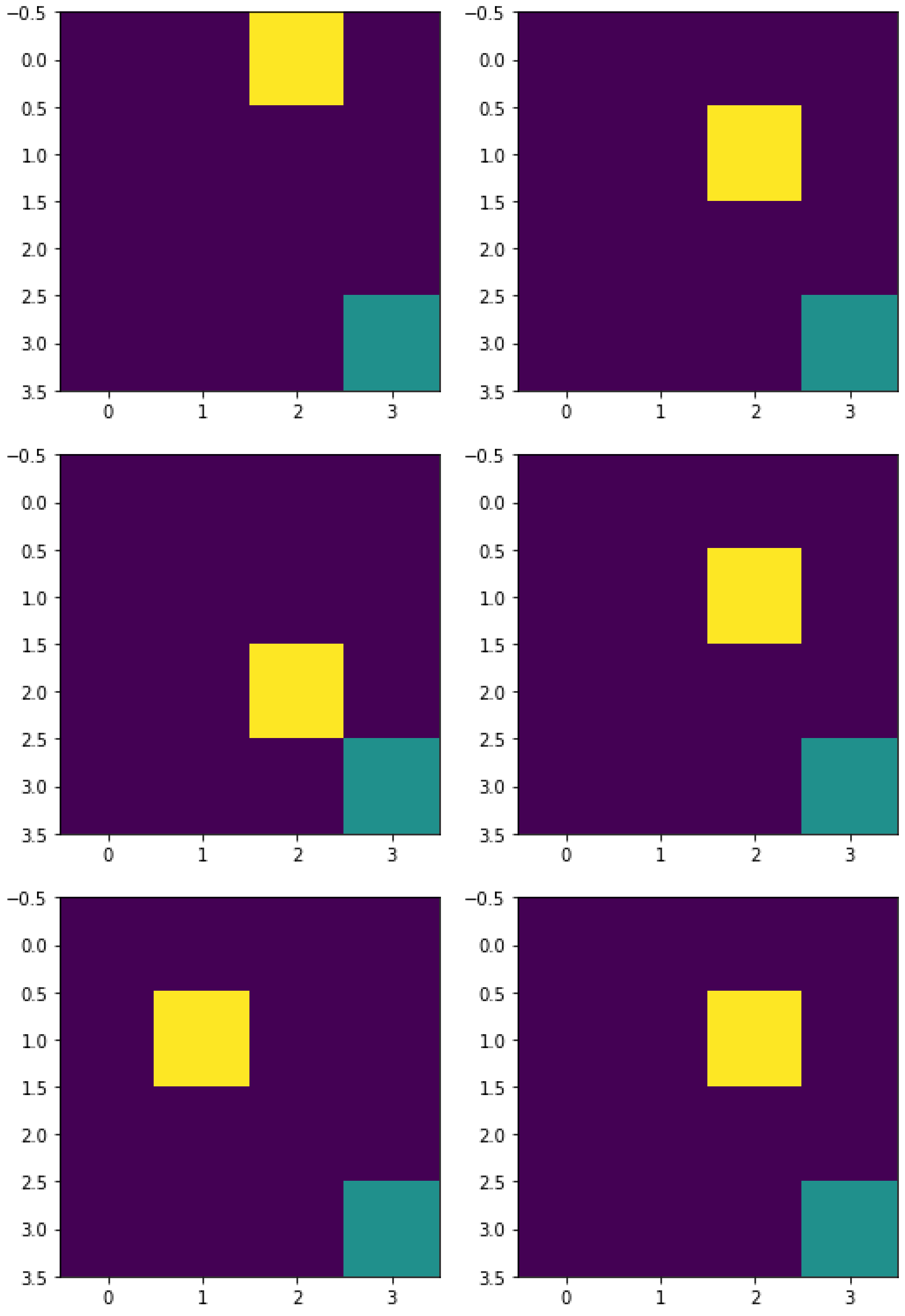


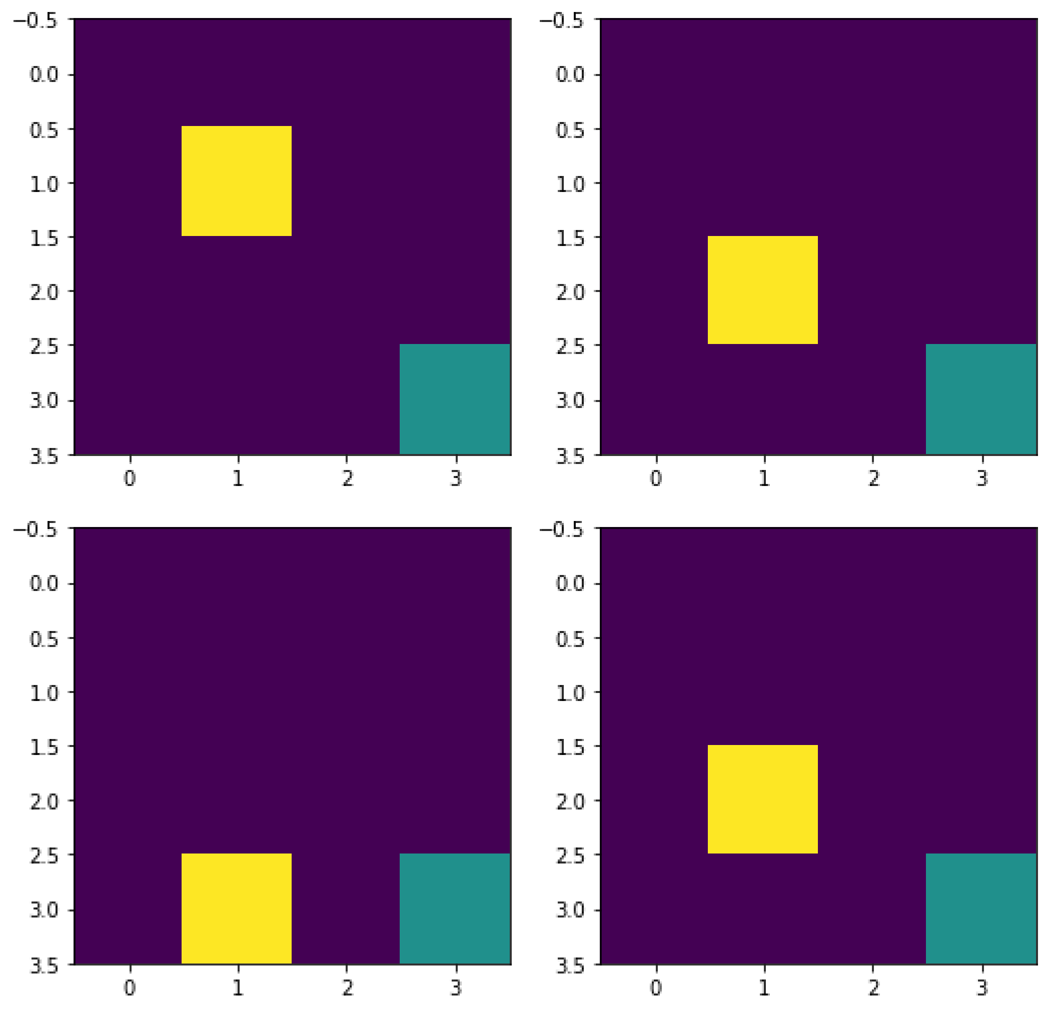









−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
1
2
3



```
Reward:  0.85
```

3. How did you define the stochastic environment?

I have taken the deterministic environment and multiplied the action with an opposite sign that is left if right top if down and vice versa conditional to the fact that the random value i am generating is less than 0.40 this means that only 60% of times will the true deterministic actions will be followed the rest of the 40% of the time the opposite action will be followed

4. What is the difference between the deterministic and stochastic environments?

In a deterministic environment my agent will always move according the policy action that is if we determine it has to go left it will always go left thus the probability is either zero or one in an deterministic environment whereas in a stochastic environment there is a degree of randomness and the agent will not always move in a decided direction, thus we have continuous distribution of probability.

5.Safety in AI: Write a brief review explaining how you ensure the safety of your environments.

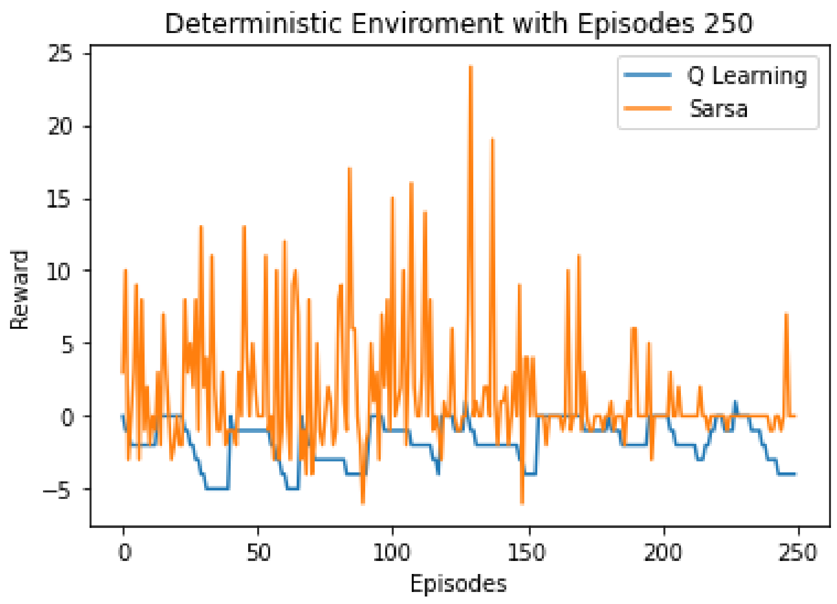
I have used np.clip to ensure that at any point of time the agent does not go beyond the specified grid limit. This is very important in the real world as for a robot going beyond its designated environment can be fatal.

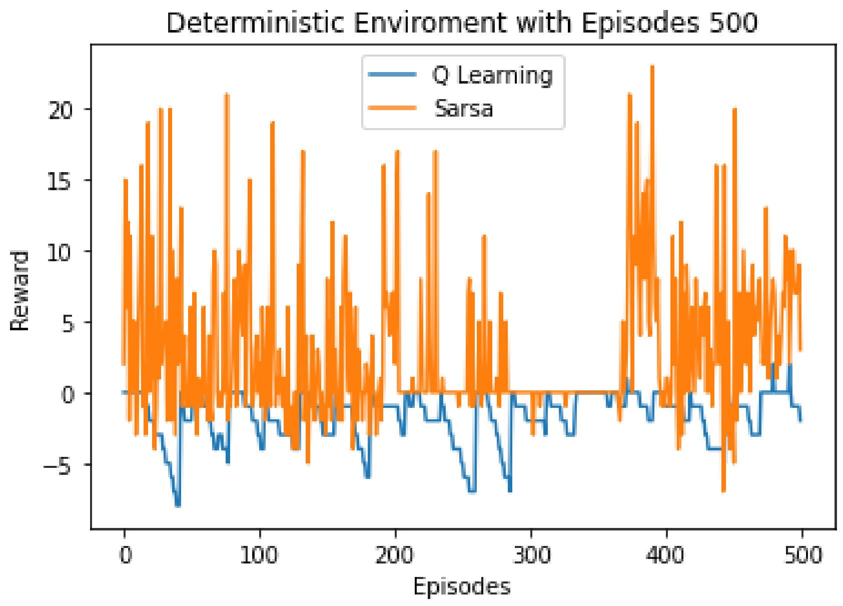
Part 2

In addition to the designated q learning I have used sarsa, state action reward state action as my second reinforcement learning policy implementation .

1.Plots are labelled and presented in the jupyter notebook.

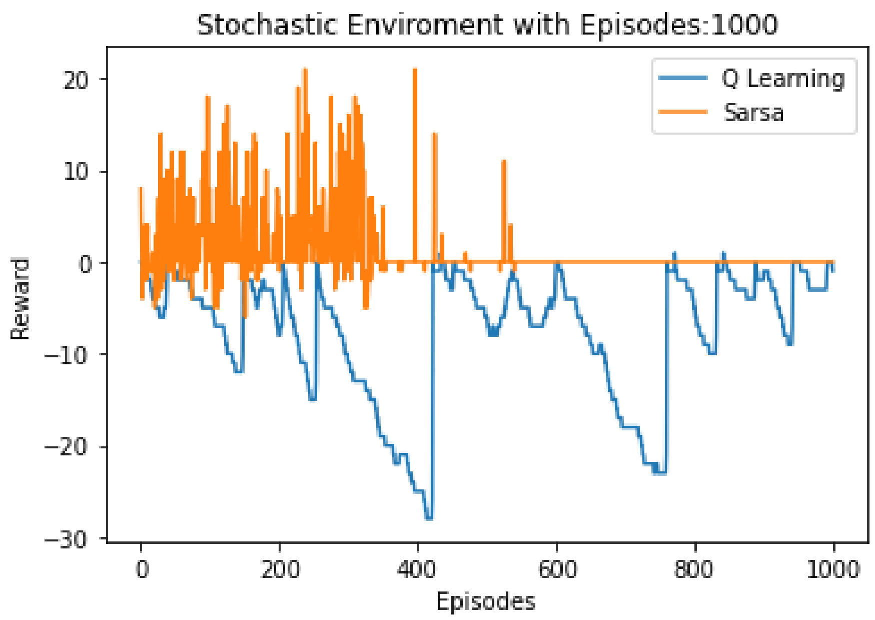
2. Compare the performance of both algorithms on the same deterministic environment (e.g. show one graph with two reward dynamics) and give your interpretation of the results.

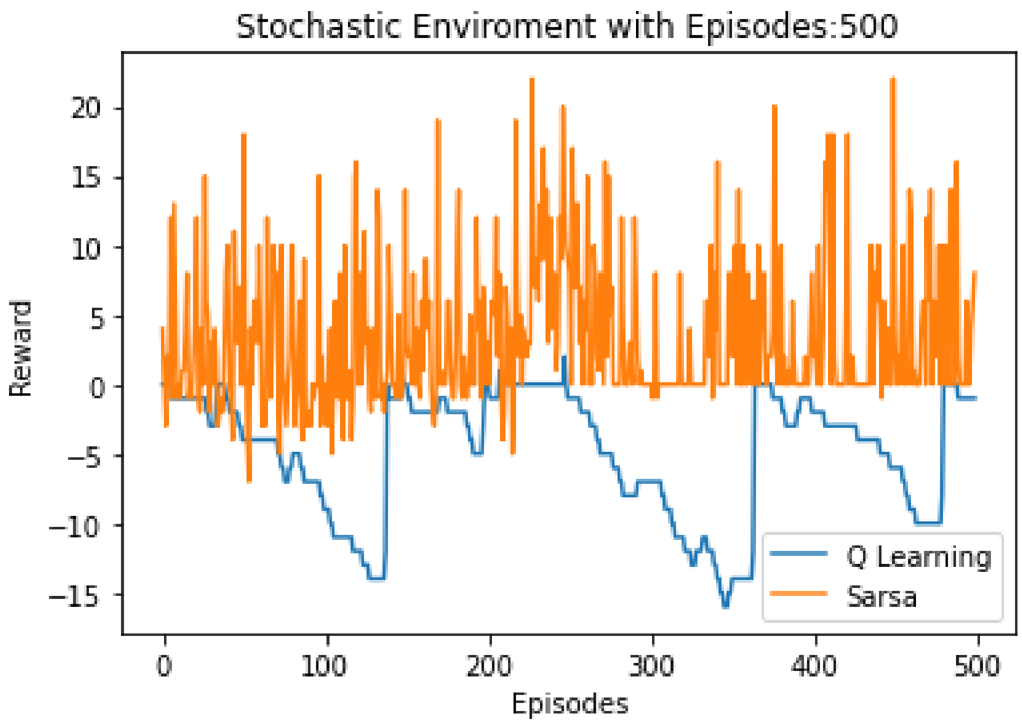




Firstly from the graph of deterministic and stochastic environments we can tell that deterministic policy allows for better and faster convergence than stochastic policy which has a certain degree of randomness as it is not always that our action will translate into the expected results.
The difference in performance in case of a stochastic and a deterministic environment is due to the fact that, once we are able to determine the best action, we can easily follow a greedy approach without any randomness, if we were able to figure out a good policy the performance in a deterministic setting will be better on the other hand stochastic setting is useful if there is a lot to learn. Basically it is the same as exploration and exploitation dilemma. This to an extent controlled by the epsilon decay which decreases our models randomness or ability to explore with time as we learn something and formulate a policy.

3. Compare how both algorithms perform in the same stochastic environment (e.g. show one graph with two reward dynamics) and give your interpretation of the results.





For the same stochastic environment sarsa is better if we are looking at time to convergence however given sufficient iterations q learning tends to arrive at similar results. This is due to the fact that sarsa has simultaneous updation in its update function so in effect is faster to learn.

4. Briefly explain the tabular methods, including Q-learning, that were used to solve the problems. Provide their update functions and key features.

Q learning

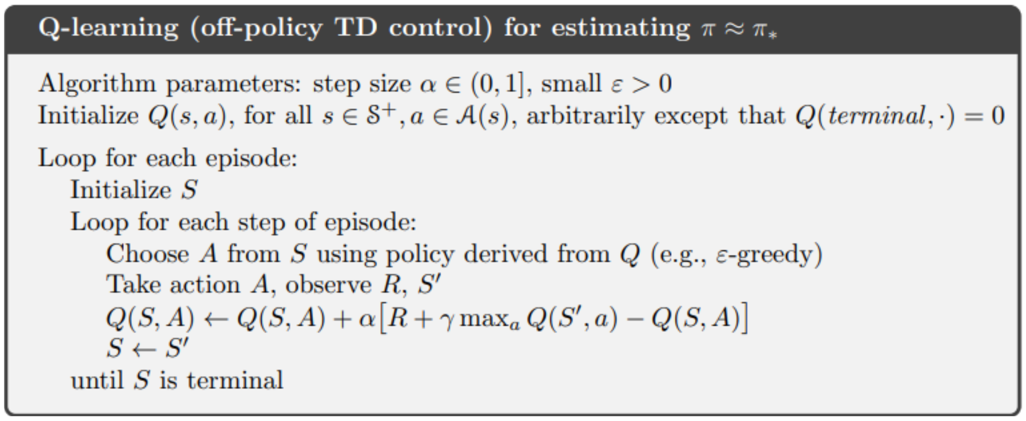
Model free tabulation method for reinforcement learning that relies on temporal difference learning.

Q Learning is a value based reinforcement learning algorithm where the values also known as q values are updated by using a value function for example the Bellman Equation. The Q in q learning stands for quality it represents the utility of a particular action in a particular state with respect to the objective of maximizing future cumulative reward.
It is an off policy model free learning algorithm

It uses the temporal differences to update the Q values
Q new (S(t), A(t) Q(s(t),A(t))+ α .[r(t) + γ. Max(Q(s(t+1),A) – Q(S(t),A(t)]
Policy Improvement is done with the temporal difference learning where the q or quality values are updated.

Pros: Can calculate expected utility of an action without a model and a policy
Cons: Learning is expensive for the agent in the beginning specially if we have a lot of possible states, in such a case we must keep discount factor as close to 1 as possible.

The update function used was



```
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize $S$
    Loop for each step of episode:
        Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        Take action $A$, observe $R$, $S'$
        $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
        $S \leftarrow S'$
    until $S$ is terminal
```

Here alpha was the learning rate and gamma the decay rate which penalizes actions which have no net impact and it is particularly useful for cases where we have a time constrained environment.

My implementation

We start from the destination and find the optimum action for each state.

```
   def find_qvalues(self):

        #we start from the back or the terminal state
        self.event.reverse()
        count=len(self.event)
        for i in range(count-1):
            oldQvalue = self.Q_table[tuple(self.event[i+1][0])][self.event
[i][1]]
            self.Q_table[tuple(self.event[i+1][0])][self.event[i][1]] = ol
dQvalue + self.alpha * (self.event[i][2] + self.gamma * max(self.Q_table[t
uple(self.event[i][0])]) - oldQvalue)
```

Sarsa is Q learning but with a slight modification to the update function

SARSA stands for state action reward state action, it is a modified q learning algorithm where target policy is same as behavior policy.
We have the tuple state action reward state action (s, a, r, s', a')

As Target policy is same as behavior policy SARSA is an on policy learning algorithm. The 2 consecutive state action pairs and the immediate reward received by the agent while transitioning from the first state to the next state determines the updated Q value

$Q(s_{t},a_{t}) = Q(s_{t},a_{t}) + \alpha (r_{t+1}+ \gamma Q(s_{t+1},a_{t+1})- Q(s_{t},a_{t}))$

Pros: The learning is faster in SARSA as it takes action optimal in the next step thus policy formulation and learning is faster.
Cons: Sarsa can be trapped in a local minima and can keep on trying to find the best policy till perpetuity.

My Implementation

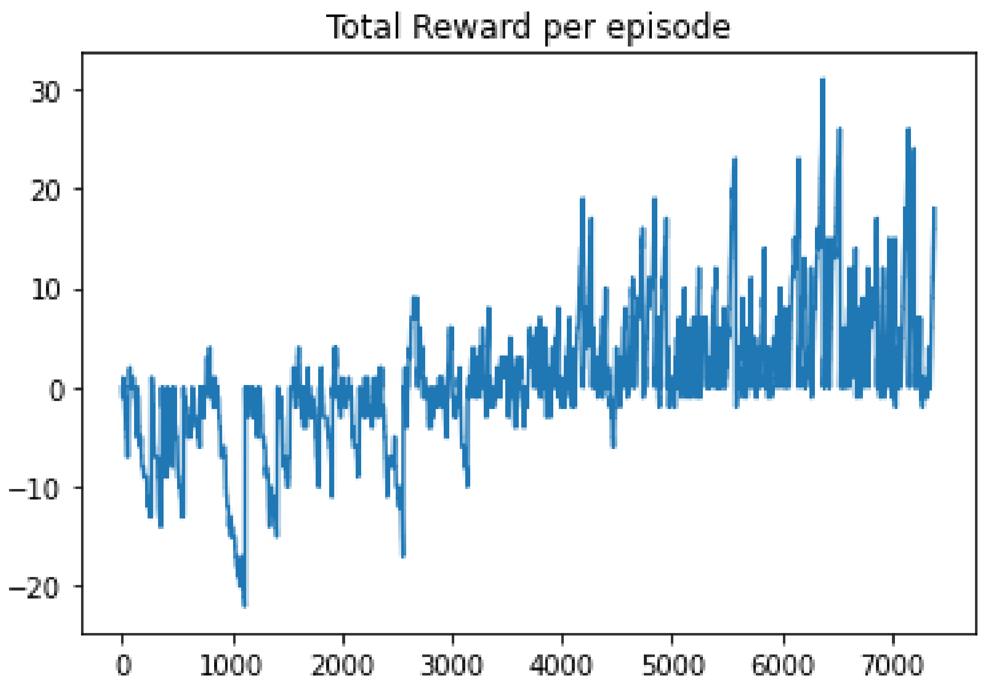
```
    def q_val_sarsa(self):
      self.event.reverse()
      for i in range(len(self.event)-1):
          oldQvalue = self.Q_table[tuple(self.event[i+1][0])][self.event[i
][1]]
          allStates = [i[0] for i in self.event]
          actionChain = self.possible_actions(allStates)
          if not self.done:
              statesChain = self.getAllFinalPosition(actionChain, self.eve
nt[i][0])
              actionSelected, stateSelected = self.get_state_action(action
Chain, statesChain)
              self.Q_table[tuple(self.event[i+1][0])][self.event[i][1]] =
oldQvalue + self.alpha * (self.event[i][2] + self.gamma * max(self.Q_table
[tuple(stateSelected)][actionSelected]) - oldQvalue)

          self.Q_table[tuple(self.event[i+1][0])][self.event[i][1]] = oldQ
value + self.alpha * (self.event[i][2] - oldQvalue)
```

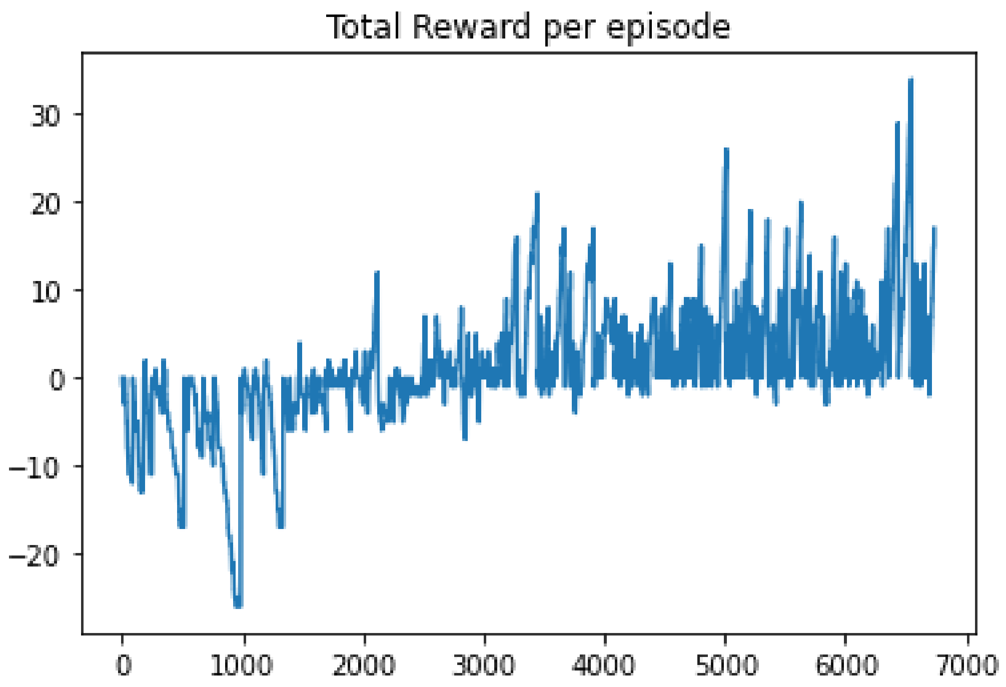
3.Hyperparameter Tuning

I have take 4 different values of alpha and 4 different values of gamma

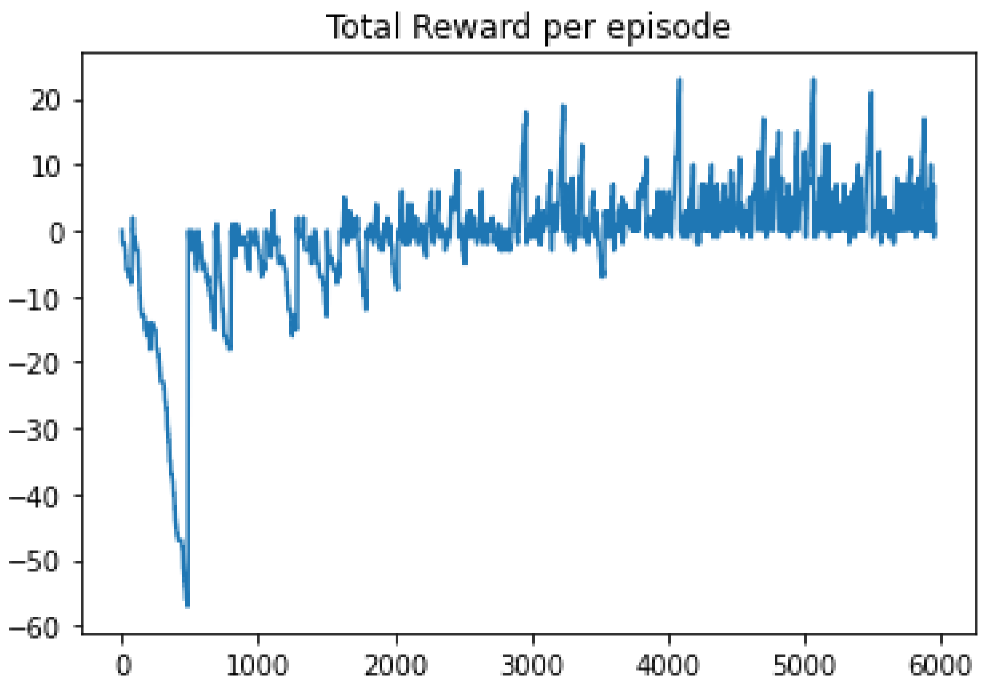
```
Alpha Value  0.3 Gamma Value 0.25
gamma or discount factor we are using 0.25
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.3
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



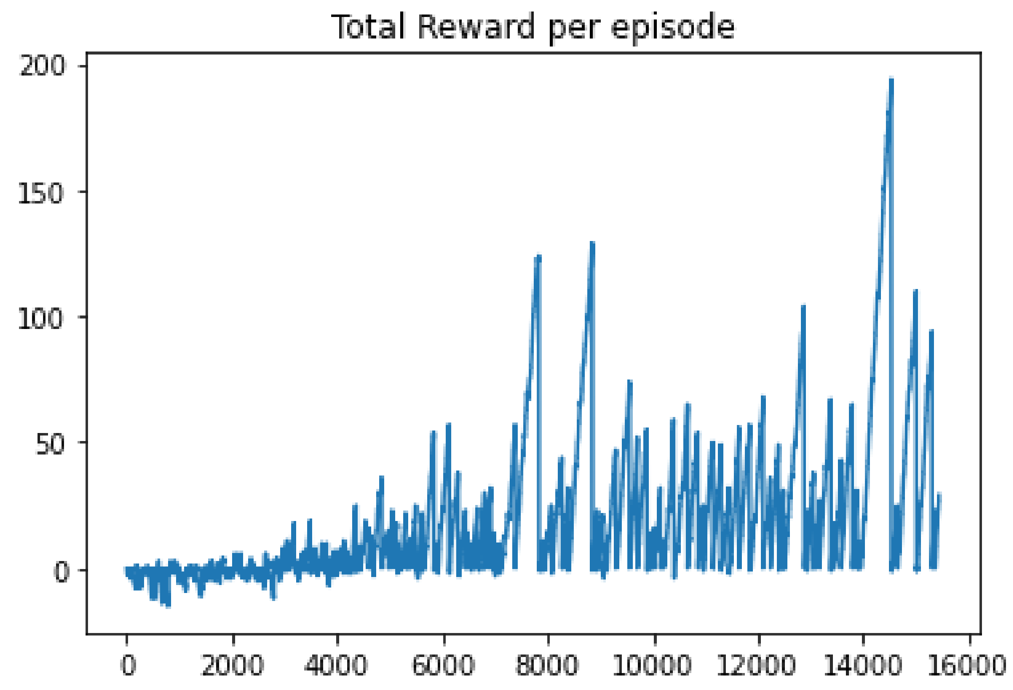
```
Alpha Value  0.3 Gamma Value 0.4
gamma or discount factor we are using 0.4
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.3
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



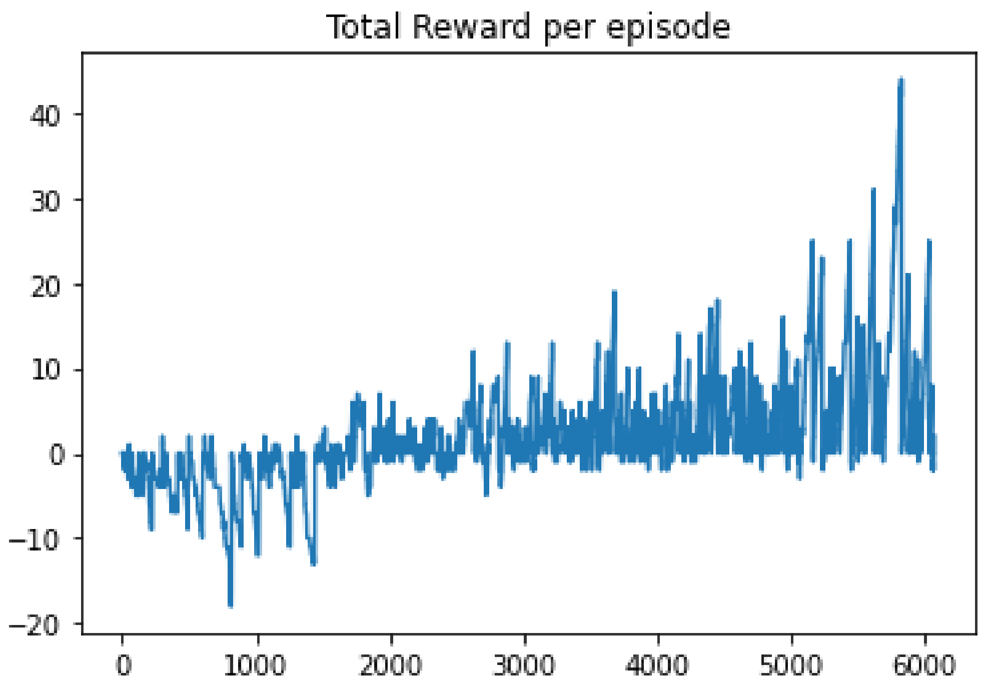
```
Alpha Value  0.3 Gamma Value 0.6
gamma or discount factor we are using 0.6
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.3
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



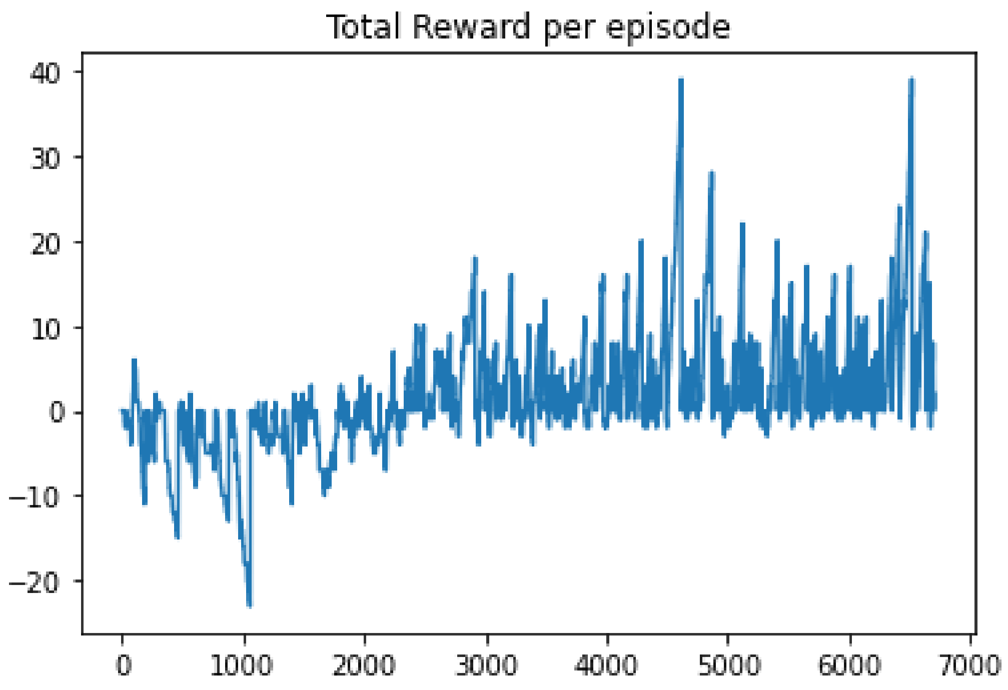
```
Alpha Value  0.3 Gamma Value 0.8
gamma or discount factor we are using 0.8
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.3
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



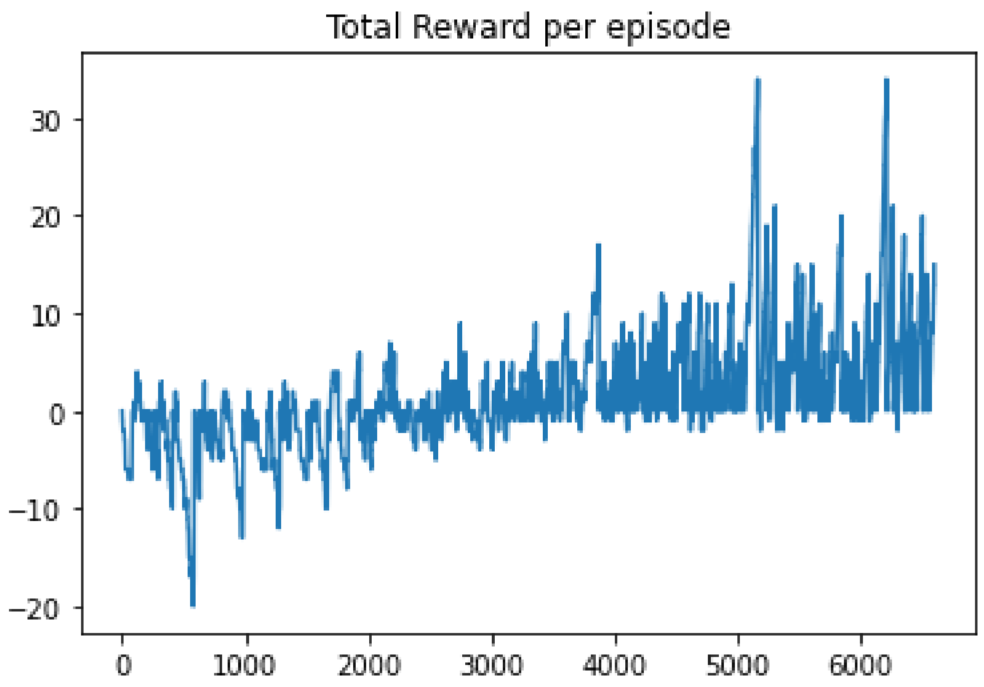
```
Alpha Value  0.5 Gamma Value 0.25
gamma or discount factor we are using 0.25
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.5
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



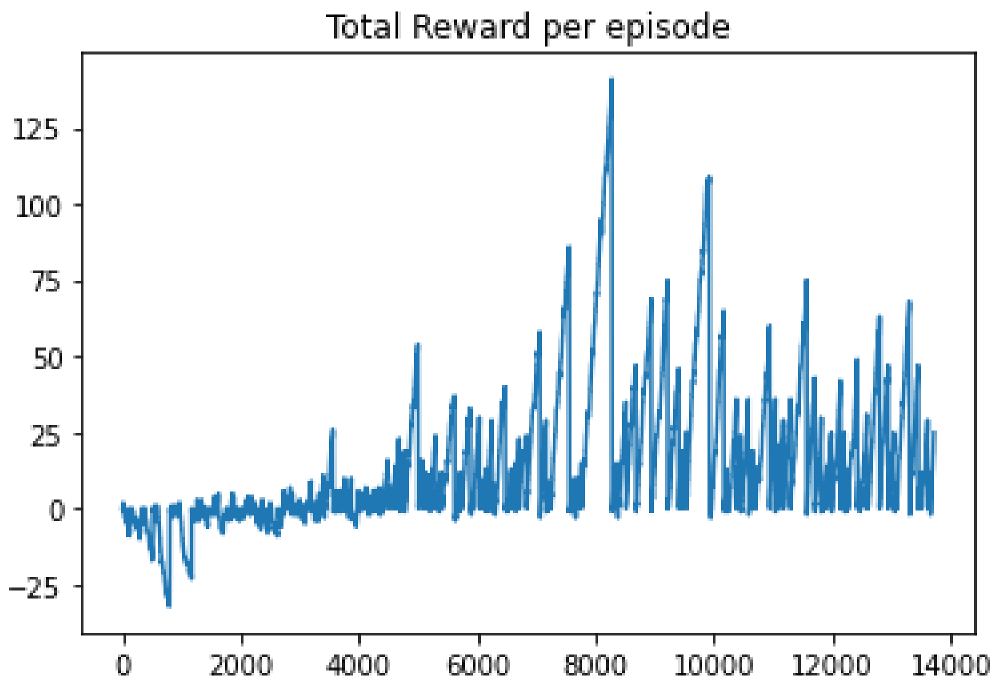
```
Alpha Value  0.5 Gamma Value 0.4
gamma or discount factor we are using 0.4
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.5
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



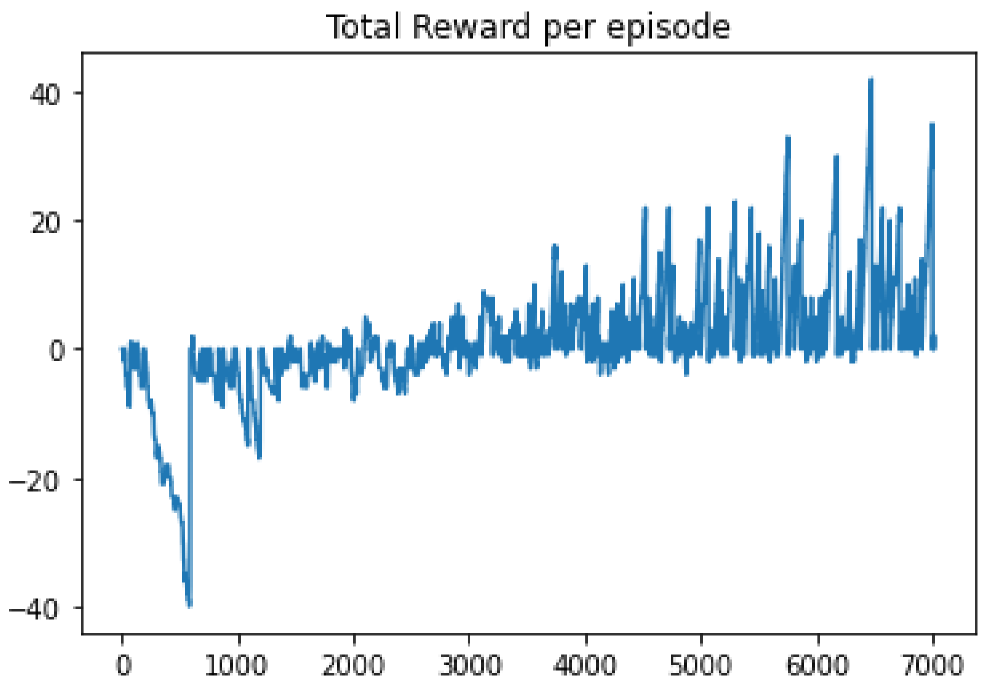
```
Alpha Value  0.5 Gamma Value 0.6
gamma or discount factor we are using 0.6
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.5
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



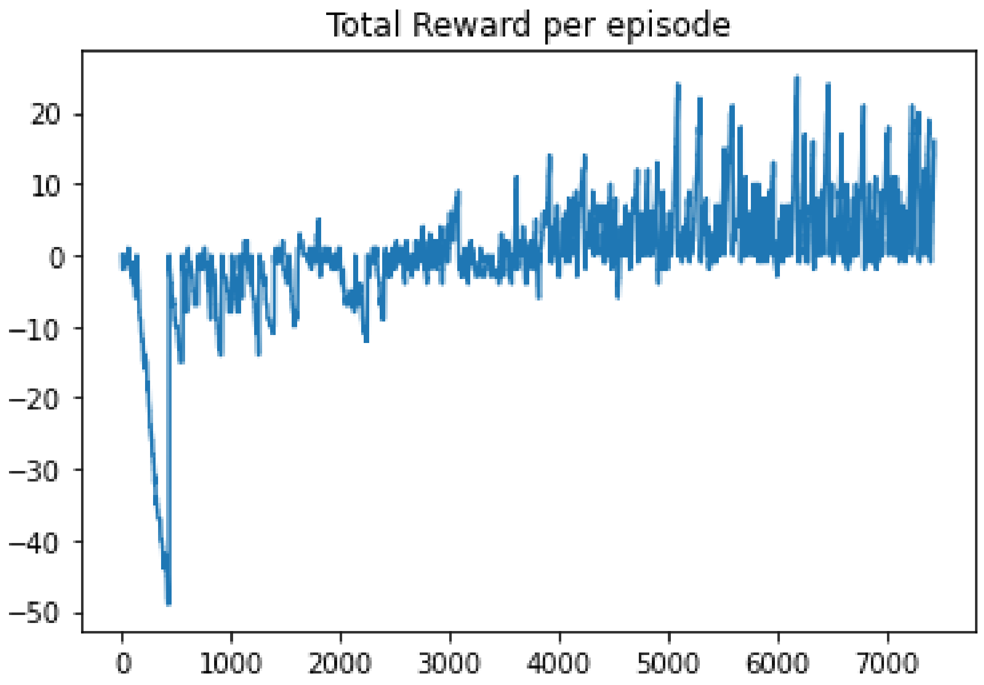
```
Alpha Value  0.5 Gamma Value 0.8
gamma or discount factor we are using 0.8
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.5
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



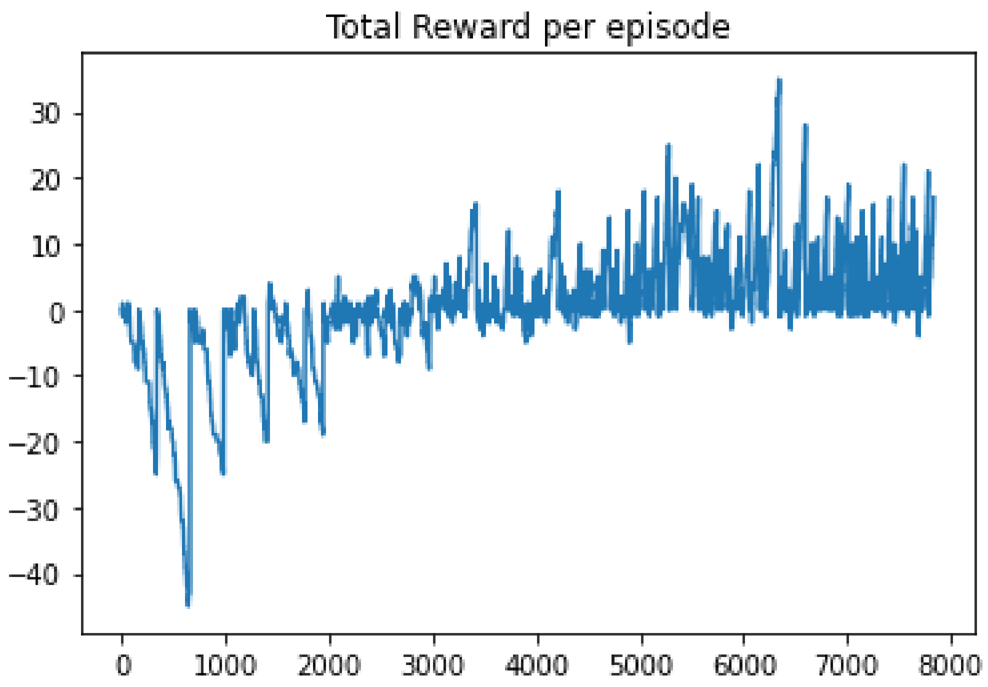
```
Alpha Value  0.7 Gamma Value 0.25
gamma or discount factor we are using 0.25
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.7
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



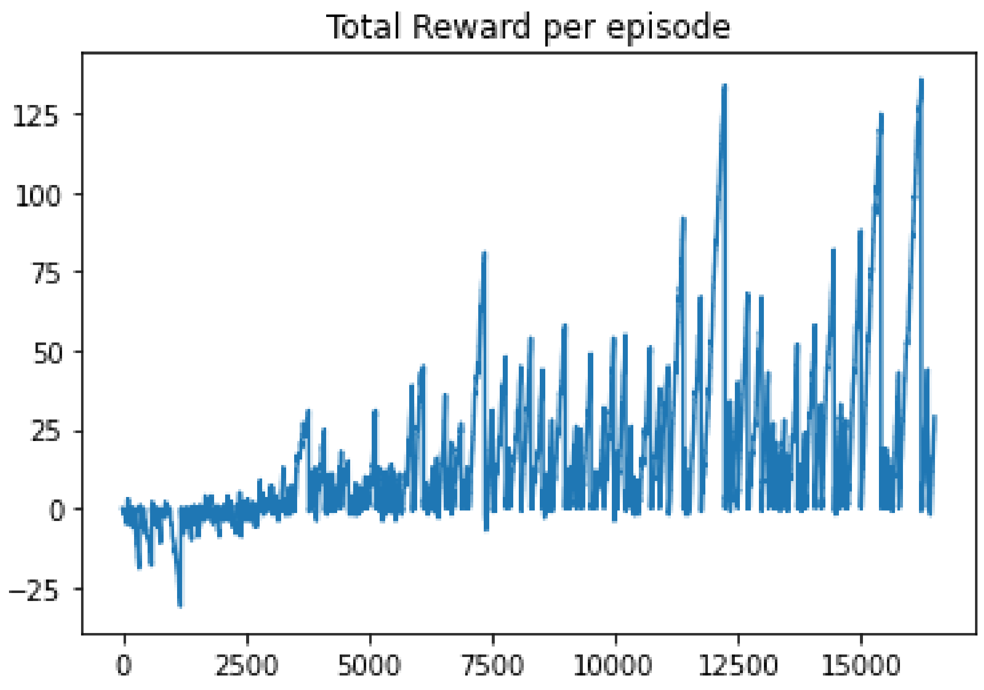
```
Alpha Value  0.7 Gamma Value 0.4
gamma or discount factor we are using 0.4
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.7
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



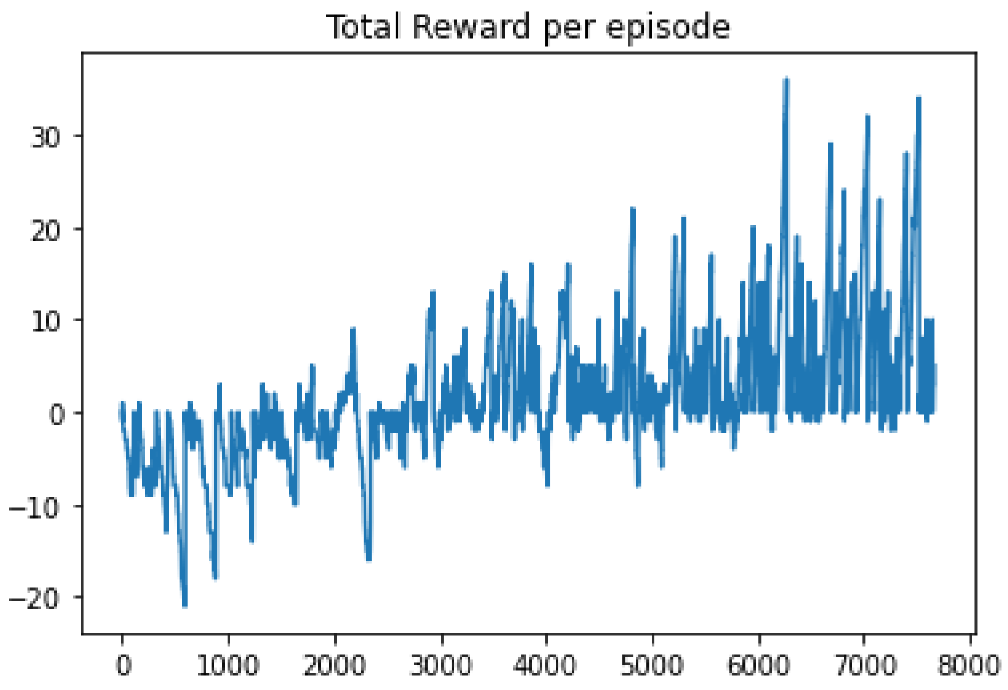
```
Alpha Value  0.7 Gamma Value 0.6
gamma or discount factor we are using 0.6
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.7
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



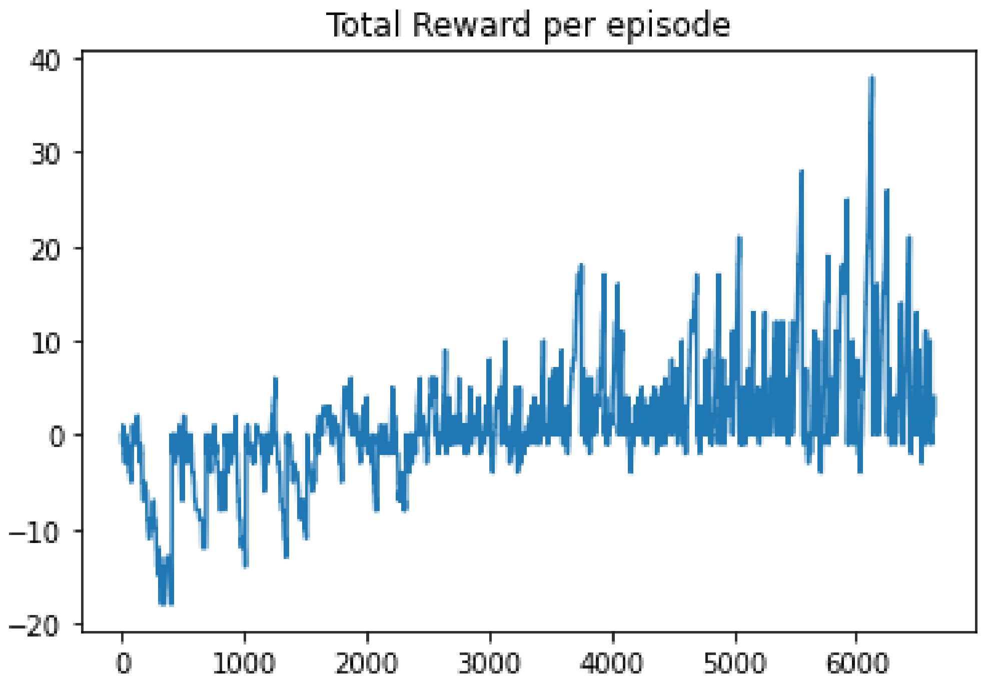
```
Alpha Value  0.7 Gamma Value 0.8
gamma or discount factor we are using 0.8
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.7
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



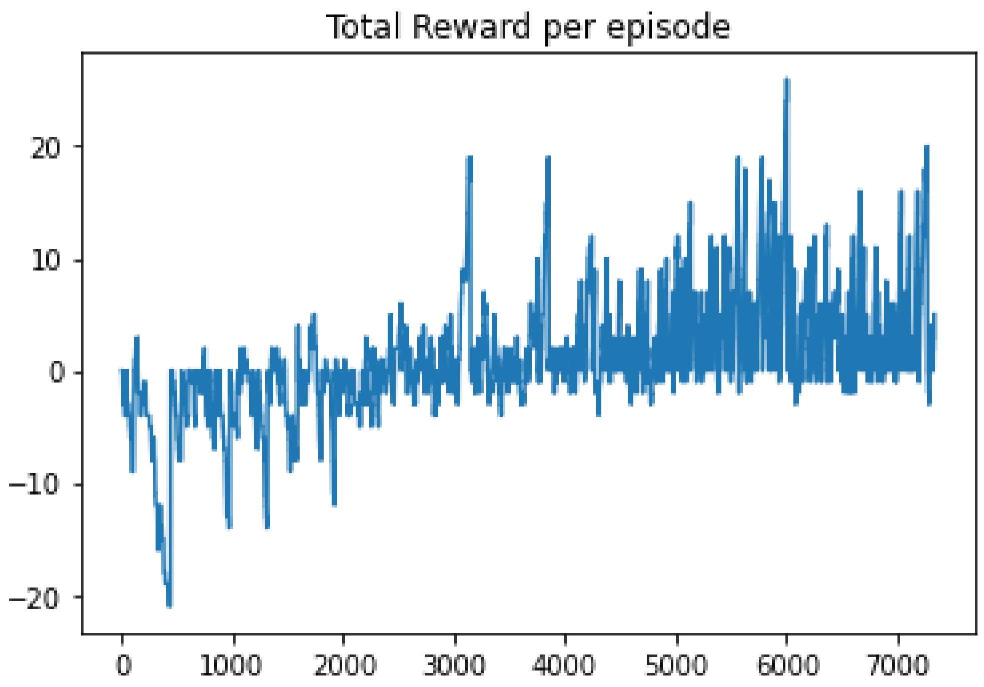
```
Alpha Value  0.9 Gamma Value 0.25
gamma or discount factor we are using 0.25
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.9
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



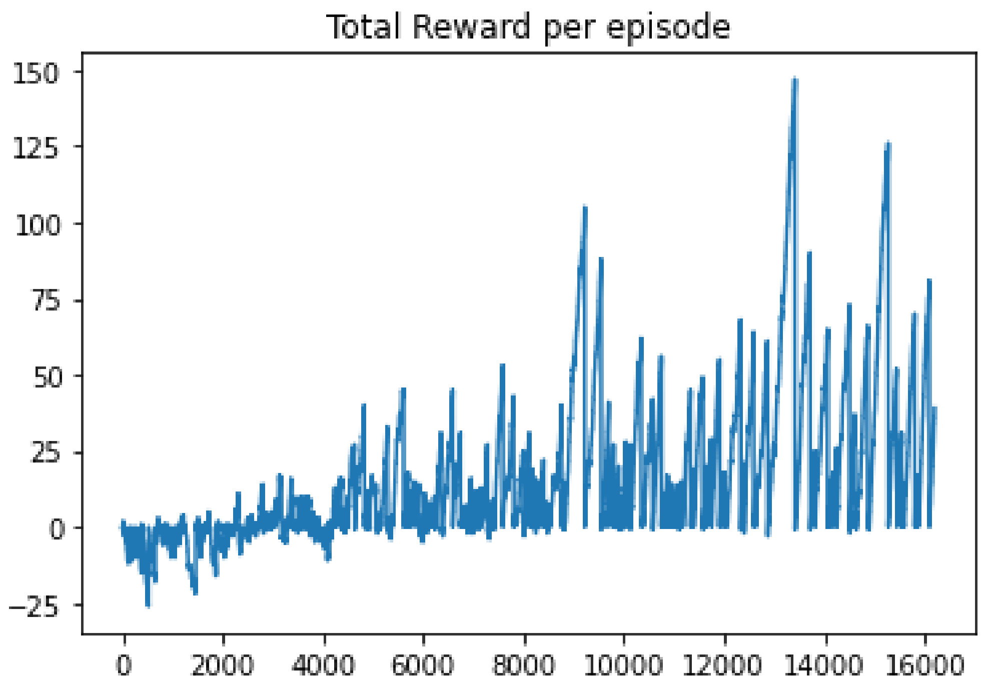
```
Alpha Value  0.9 Gamma Value 0.4
gamma or discount factor we are using 0.4
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.9
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



```
Alpha Value  0.9 Gamma Value 0.6
gamma or discount factor we are using 0.6
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.9
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```



```
Alpha Value  0.9 Gamma Value 0.8
gamma or discount factor we are using 0.8
epsilon or degree of randomness we are using 0.9
As we learn about the appropiate actions in our Q table we decrease the
stochasticity i.e. the epsilon value
epsilon decay we are using is  0.9
Learning rate alpha we are using is 0.9
we will train for  200  and the max actions in a episode berfore termination
is 25
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Our Rewards corresponding to different states 0
```


Total Reward per episode
150
125
100
75
50
25
0
−25
0
2000
4000
6000
8000
10000
12000
14000
16000