Report: CS747
Assignment 3
Siddhesh Pawar
17D17011

Common Assumptions:

1. The values for α and ε are kept as 0.5 and 0.1 respectively. The number of episodes is kept 200, and the maximum step limit is set to be 10000 and discount factor is set to 1
2. A ε-greedy policy is used to select an action from a state to go to the next state. This also ensures that the SARSA implementation converges.
3. When the agent is at the border of the gridworld, and the next action is such that it takes the agent out of the grid, the agent stays at its original place(and gets reward -1).
4. The plots are plotted by running all the algorithms 15 times(over different seeds) and then taking average over the runs
5. States are encoded as 2D numpy array

Code Structure:

1. Function **policy** that implements the ε-greedy
2. Function **get_nextstate_reward** that takes in input firstly whether the wind is stochastic or not.
   Then takes in the action number and gives the output as the next state considering the wind.The reward is -1 if the next state is not the end state and 0 otherwise. All the 8 actions have been encoded, only 4 of which are required for the tasks without king's moves

TASK 1

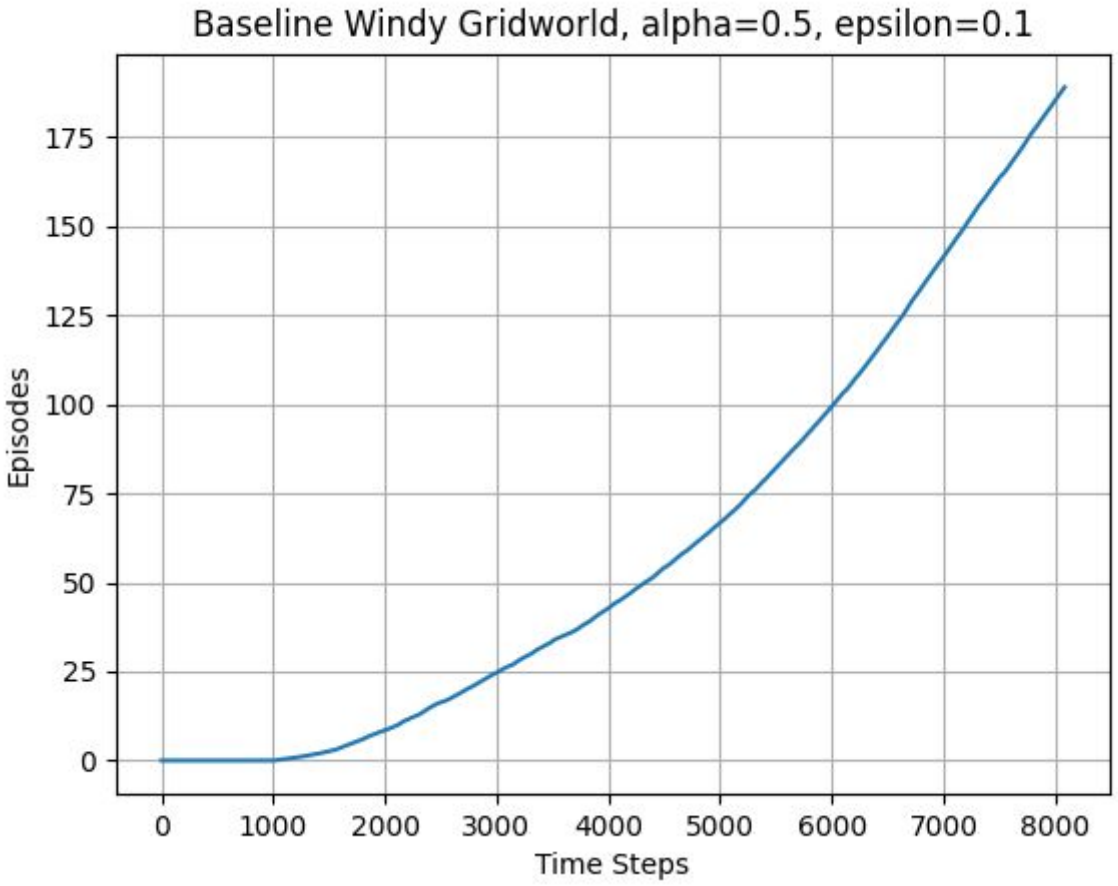
Implement Windy Gridworld as an episodic MDP.
The GrindyWorld maze has been set as an mdp with rows = 7 and columns = 10.
Start_state = (3,0)
End_state = (3,7)
No of actions = 4 (Up, Down, Left, Right)
Discount factor = 1
Exploration Rate(epsilon) = 0.1
Alpha(Learning Rate) = 0.5
Winds = [0,0,0,1,1,1,2,2,1,0]

TASK 2

Implement Sarsa(0) that is on policy sarsa.

No of actions = 4 (Up, Down, Left, Right,Northeast,Northwest,Southeast,Southwest)

Plot:



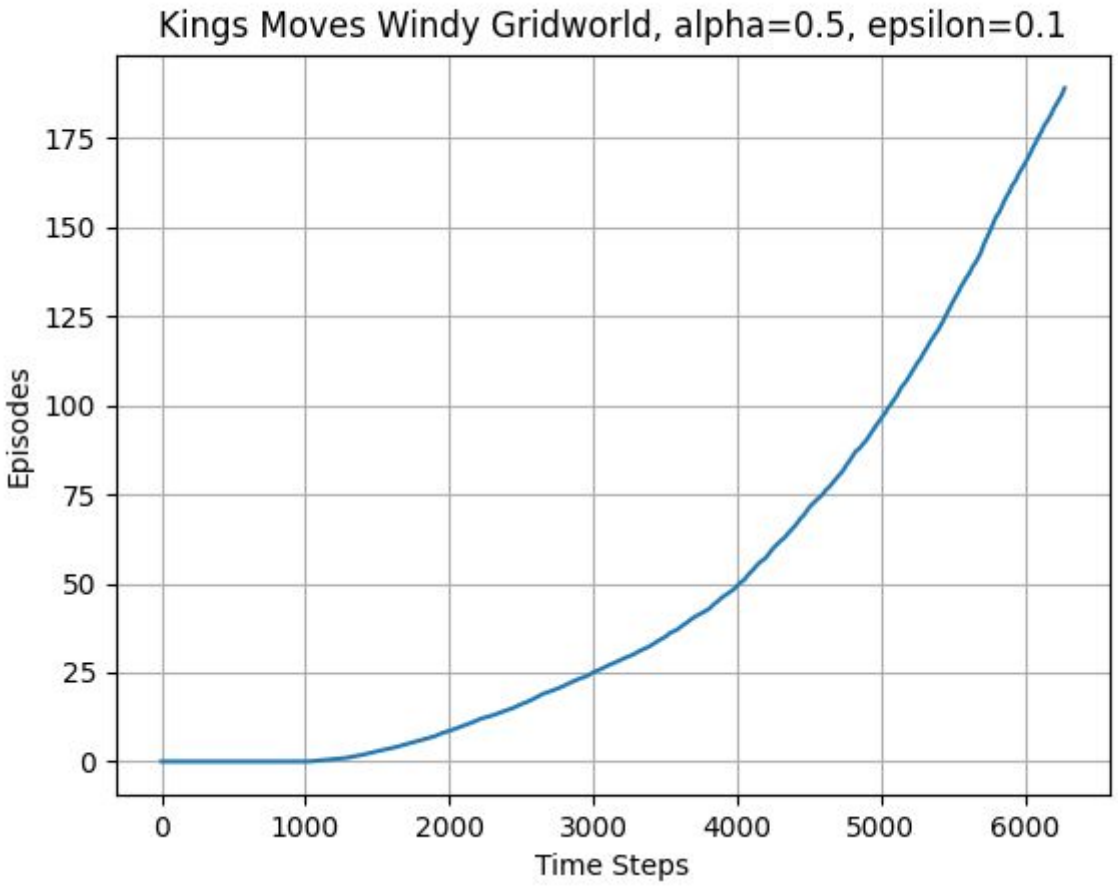
Observations:-

The no of time steps increase exponentially in the beginning and then the graph gets steeper. This shows that the agent reaches the goal rather quickly after sufficient time steps when Q becomes approximately similar to Q*.The increasing slope of the graph shows that the goal was reached more quickly over time.

TASK 3

For king’s move the total number of actions are 8 which include the original actions as well as diagonal moves.

Plot:-



Observations:-

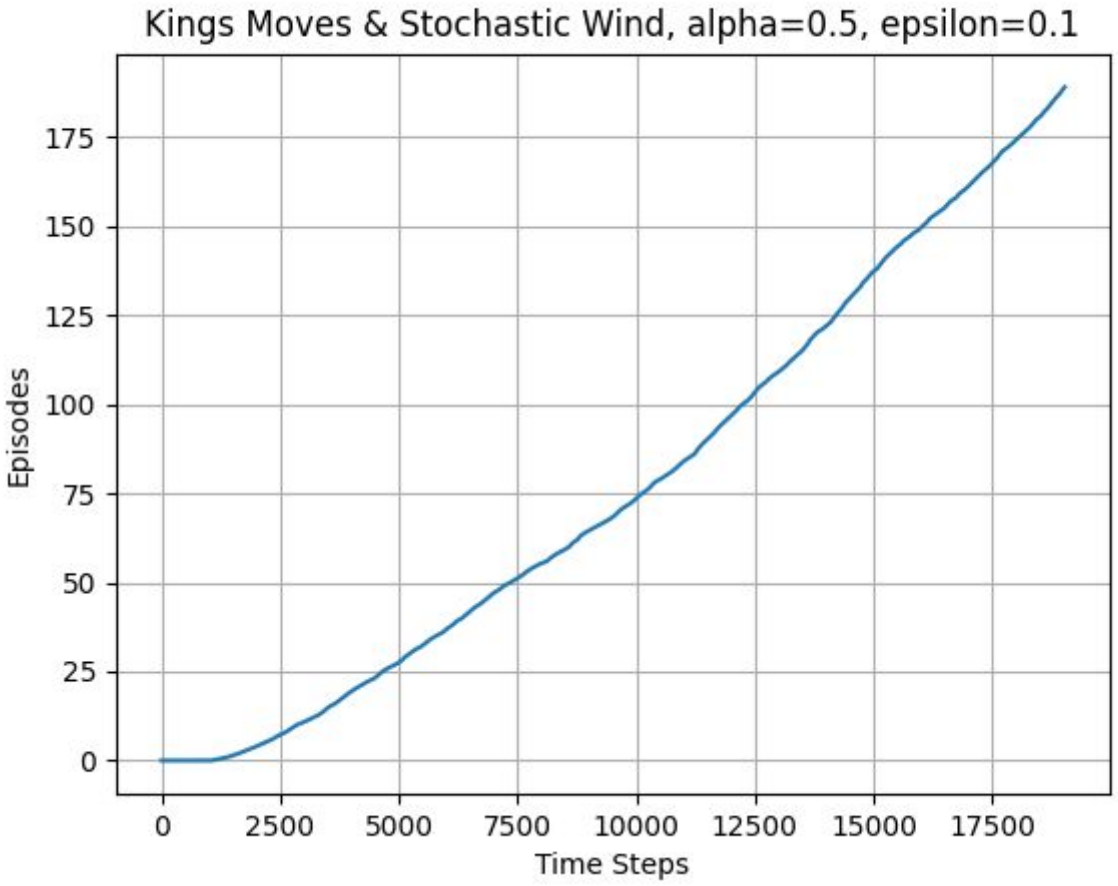
- The no of time steps increase exponentially in the beginning and then the graph becomes linear and steeper as episodes are increased. This indicates that the agent has learnt the best possible policy to navigate the grid.
- The number of timesteps required to reach the convergence is less as compared to the windy gridworld with 4 moves as this has more degrees of freedom and can reach the end points earlier

TASK 4

For stochasticity, the wind will be [w-1,w,w+1]. Out of these, at any location one value of wind will be randomly selected with equal probability.

Plots:-

Time step = 200 episodes



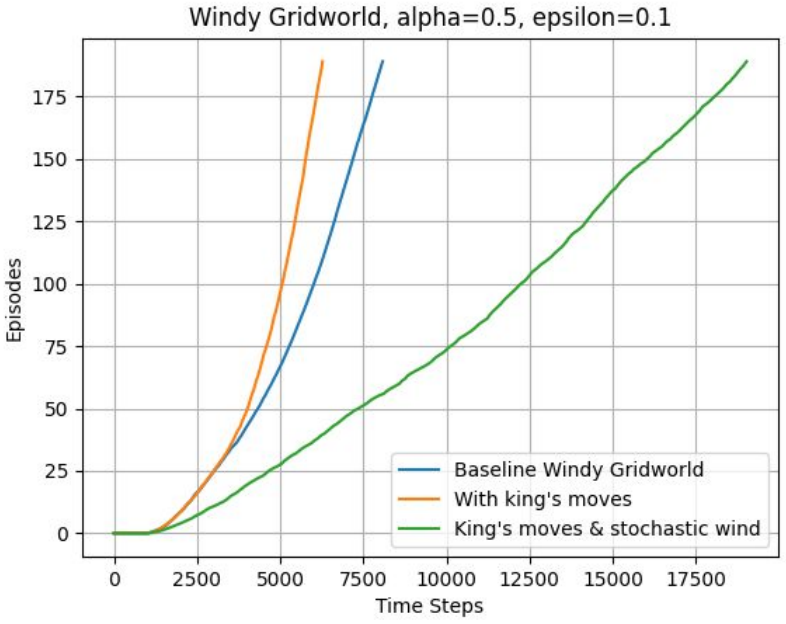
Time step = 1000 episodes:-

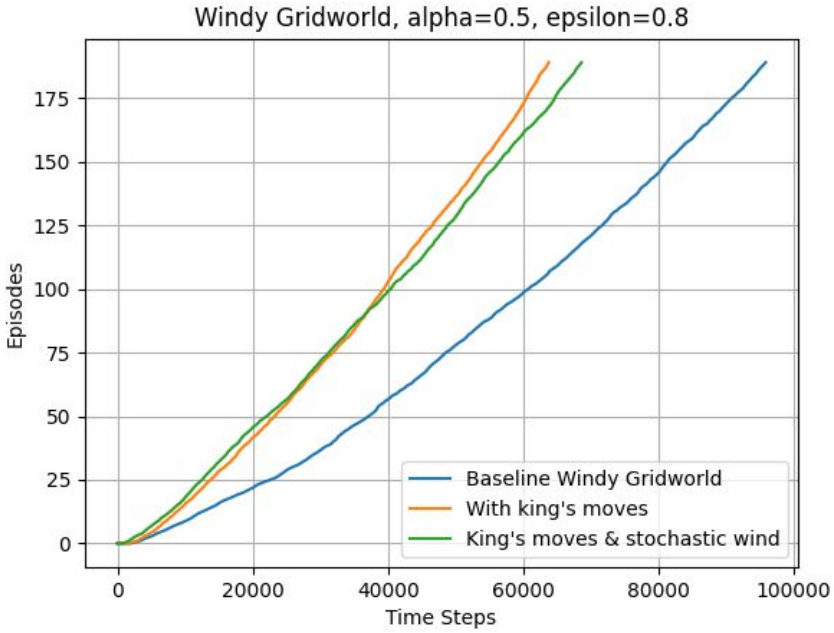
Observations:-

- The no of time steps increase exponentially in the beginning and then linearly as episodes are increased. This indicates that the agent has learnt the best possible policy to navigate the grid.
- The number of episodes required to achieve linear is higher for this case as compared to without as optimal policy has to be learnt for the stochastic wind.

- As the wind strength is fluctuating randomly and therefore, it is harder to learn an optimum policy. Therefore, the algorithm takes more steps than the other situations
- Increasing the epsilon even more worsens the performance of all the algorithms but when epsilon is high enough, the policy becomes almost like random exploration in that case the task with stochastic wind is learned faster

Combined Plot:

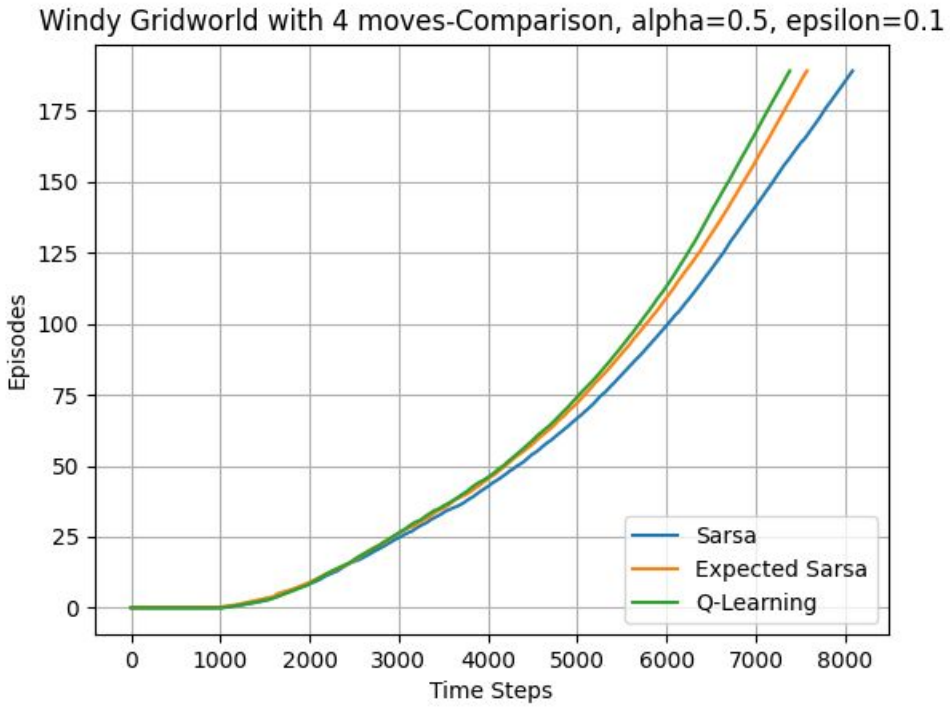




TASK 5

Implement Expected sarasa and Q-Learning

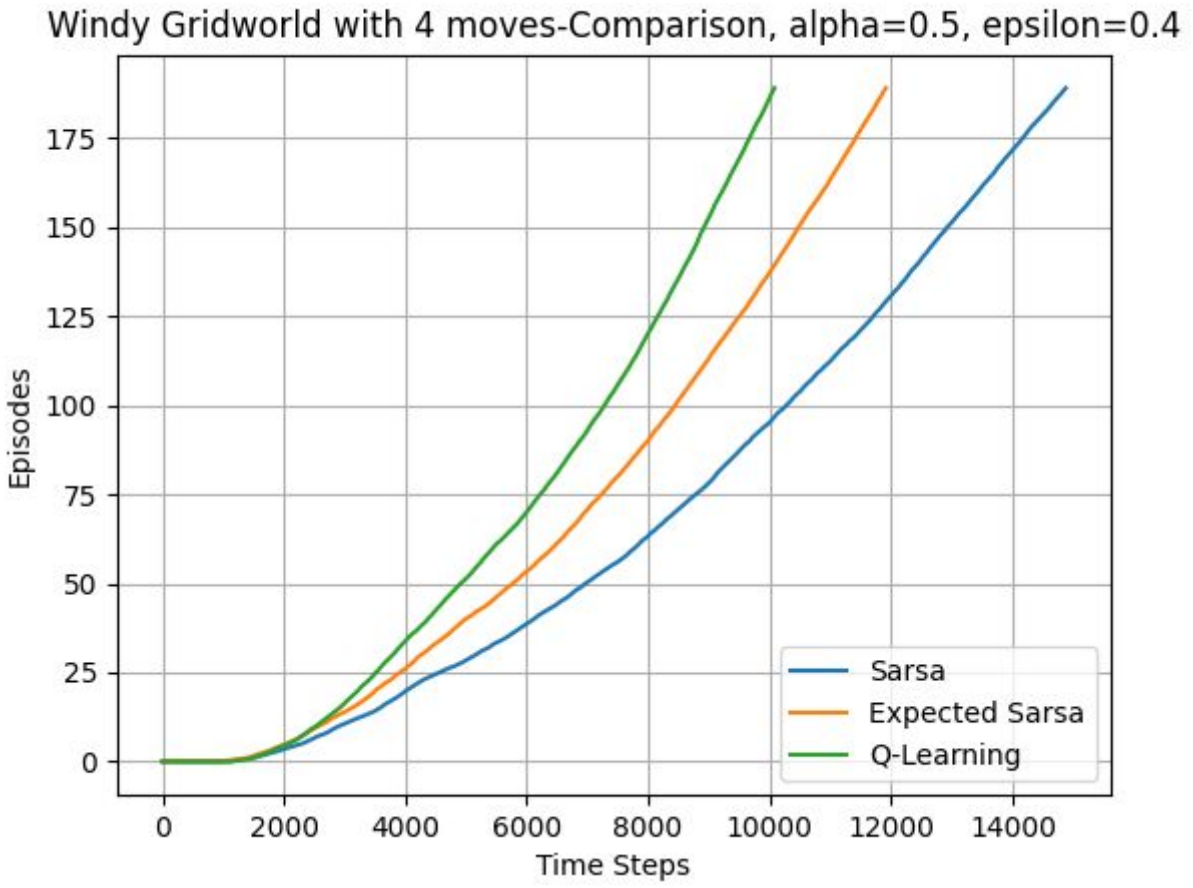
Time step = 200 episodes



Observations:-

- The number of time steps increase exponentially in the beginning and then the graph steepens. This indicates that the agent has learnt the best possible policy to navigate the grid.
- The number of time steps required for Q-Learning is the least and sarsa is the highest among the 3.
- The Q-Learning is expected to perform better than sarsa and expected sarsa as it is an off-policy update.SARSA and expected will approach convergence allowing for possible penalties from exploratory moves, whilst Q-learning will ignore them. Q-learning directly learns the optimal policy, whilst SARSA learns a near-optimal policy whilst exploring(there are penalties).
- Suppose we follow a greedy policy then Expected Sarsa is exactly Q-learning. In this sense Expected Sarsa subsumes and generalizes Q-learning while reliably improving over Sarsa
- When the epsilon is increased Q-learning will converge even faster

- Expected sarsa performs better than sarsa because it penalizes exploration less as compared to sarsa



,