Chapter 6

Exercise 6.1:

Modified TD(0) update is:

$$v_{t+1}(S_t) = v_t(S_t) + \alpha * [R_{t+1} + \gamma * v_t(S_{t+1}) - v_t(S_t)] \qquad (1)$$

$$\delta_t = R_{t+1} + \gamma * v_t(S_{t+1}) - v_t(S_t) \qquad (2)$$

Monte Carlo Error:

$$\begin{aligned} G_t - v_t(S_t) &= R_{t+1} + \gamma * G_{t+1} - v_t(S_t) + \gamma * v_t(S_{t+1}) - \gamma * v_t(S_{t+1}) \\ &= \delta_t + \gamma * [G_{t+1} - v_t(S_{t+1})] \\ &= \delta_t + \gamma * [G_{t+1} - v_{t+1}(S_{t+1}) + v_{t+1}(S_{t+1}) - v_t(S_{t+1})] \\ &= \delta_t + \gamma * [G_{t+1} - v_{t+1}(S_{t+1})] + \gamma * [v_{t+1}(S_{t+1}) - v_t(S_{t+1})] \\ &= \delta_t + \gamma * [G_{t+1} - v_{t+1}(S_{t+1})] + \gamma * h_{t+1} \end{aligned}$$

Where $h_{t+1} = \alpha * [R_{t+1} + \gamma * v_t(S_{t+2}) - v_t(S_{t+1})]$ from (1)

$$\begin{aligned} &= \delta_t + \gamma * \delta_{t+1} + \gamma^2 * [G_{t+2} - v_{t+2}(S_{t+2})] + \gamma * h_{t+1} + \gamma^2 * h_{t+2} \\ &= \sum_{k=t}^{T-1} \gamma^{k-t} * \delta_k + \sum_{k=t}^{T-1} \gamma^{k-t+1} * h_{k+1} \end{aligned}$$

Therefore, additional amount added: $\sum_{k=t}^{T-1} \gamma^{k-t+1} * h_{k+1}$

Exercise 6.2:

Consider a scenario where we are following a path regularly and suddenly we have to change some portion of the path we follow. This is described in the exercise. In this scenario, we still have estimates of certain states which we are confident about. If some middle portion of the path is changed we can be confident about the estimates of the suffix of our path being correct. When we use TD, these estimates are used to immediately form an estimate about the new segment of road which we are using. This new estimate then leads to an estimate for the prefix of our path and so on. Monte Carlo will have to wait for several trips to complete before an accurate estimate of the new road segment can be formed.

Exercise 6.3:

With TD(0), in this example, it was expected that the value of only one state would change. This is because all the values are initialised by $v(s) = 0.5$ which will make the TD error = 0 for all the states in the first episode except the last state since reward for all non-terminating transitions is 0.

The fact that only $v(A)$ changed tells us that the first episode ended on the left with a return 0. $v(A)$ was thus updated to $0.5 + \alpha * [0 - v(A)] = 0.5 - 0.5 * 0.1 = 0.45$.

Exercise 6.4:

A wider range of alpha will not bring any changes to the graph. The general nature of the graph will remain the same. The smaller the alpha, the closer the estimates will get to the real value. Thus both TD and monte carlo will have better final errors with smaller alphas. So there is no magic value of alpha for which a particular algorithm would perform better.

Exercise 6.5:

This effect is due to the large values of alpha. Due to large values, estimates change rapidly and move closer to real time returns more quickly than smaller values of alpha. However, since the step size is large, the values keep bouncing around the true value. A shifted estimate then shifts the estimate of the other states even more leading to a slight rise in errors. The initial values only govern the value of the error in the initial episodes when the estimated values are far from the true values.

Exercise 6.6:

One way is to compute the state values using several iterations of DP till the values converge. This might be time consuming and precision would depend on the smallest change allowed in the algorithm.
Another approach is to simply write out the equations for each state in terms of its left and right states. We will end up with 6 equations. Solving these equations will give the exact true values for all states. We can use the fact that the value of middle state has to be 0.5 because of symmetry.

Exercise 6.7:

In the off policy Monte Carlo code, change the target from G to a TD target.

Exercise 6.8:

TD(0) Error=

$$\delta_t = R_{t+1} + \gamma * q(S_{t+1}, A_{t+1}) - q(S_t, A_t)$$

Monte Carlo error=

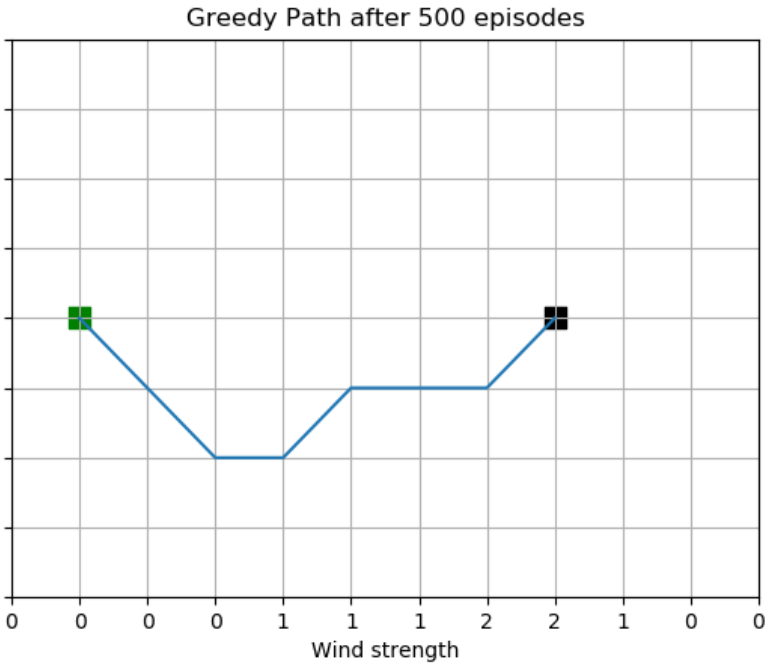
$$\begin{aligned} G_t - q(S_t, A_t) &= R_{t+1} + \gamma * G_{t+1} - \gamma * q(S_{t+1}, A_{t+1}) + \gamma * q(S_{t+1}, A_{t+1}) - q(S_t, A_t) \\ &= \delta_t + \gamma * [G_{t+1} - q(S_{t+1}, A_{t+1})] \\ &= \delta_t + \gamma * \delta_{t+1} + \gamma^2 * [G_{t+2} - q(S_{t+2}, A_{t+2})] \\ = \delta_t + \gamma * \delta_{t+1} + \gamma^2 * \delta_{t+2} + ... + \gamma^{T-1} * \delta_{T-1} + \gamma^T * [G_T - q(S_T, A_T)] \\ = \delta_t + \gamma * \delta_{t+1} + \gamma^2 * \delta_{t+2} + ... + \gamma^{T-1} * \delta_{T-1} + \gamma^T * [0 - 0] \\ = \sum_{k=t}^{T-1} \gamma^{k-t} * \delta_k \end{aligned}$$

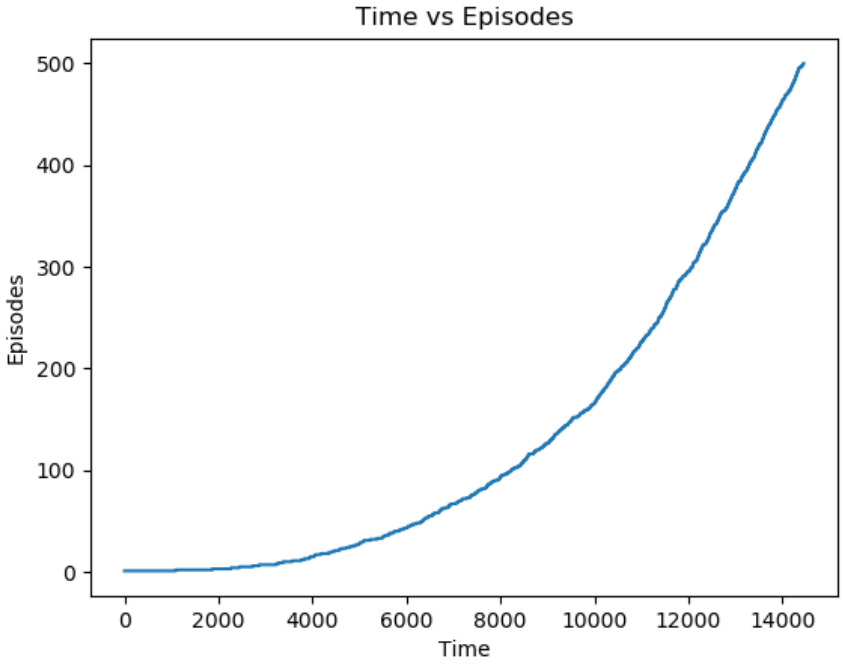
Exercise 6.9 (Programming):

Kings Moves:

Code : Check Github

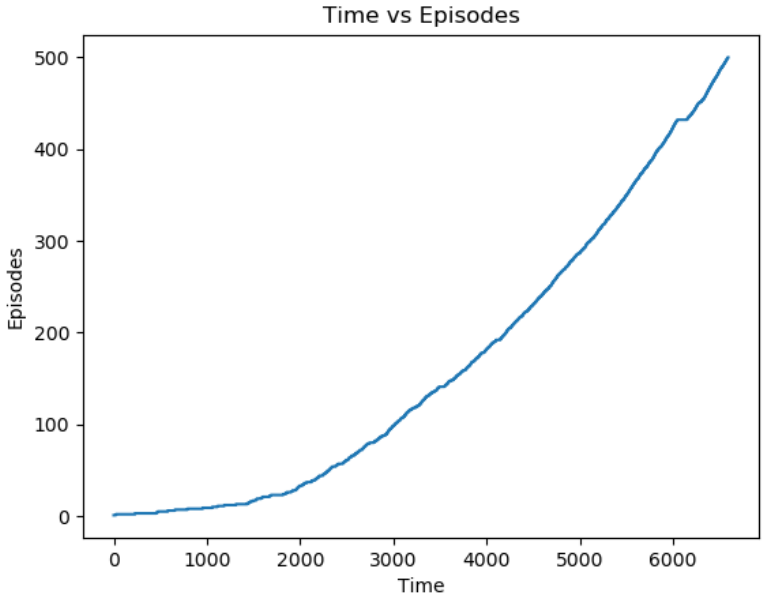
Results:





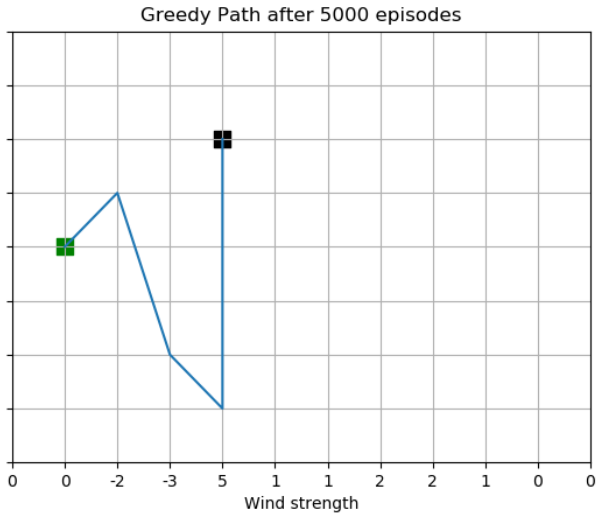
Including a 9th move brings no benefit in the given grid. This is because the shortest path length remains the same and the additional move does not give any benefit as such.

I ran a modified grid which benefits from the additional move.
Result with additional move:

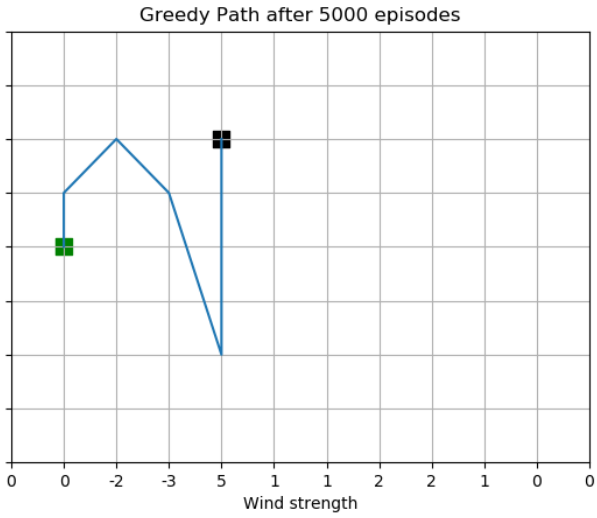


To compare the performance between normal king moves and king moves with additional move, I ran the code on a grid in which additional move leads to a shorter path.

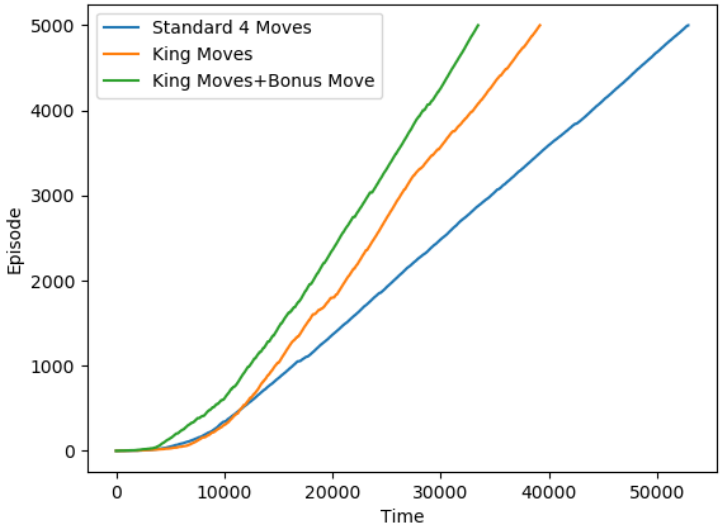
Results:



Path with Bonus Move



Path Without Bonus Move

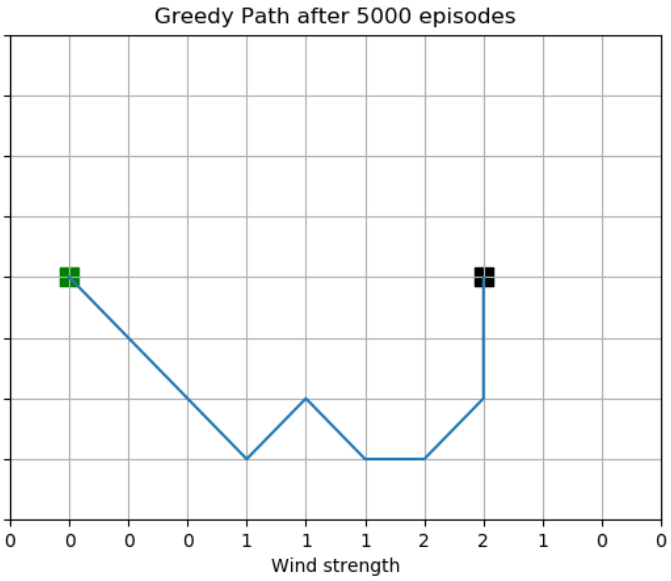


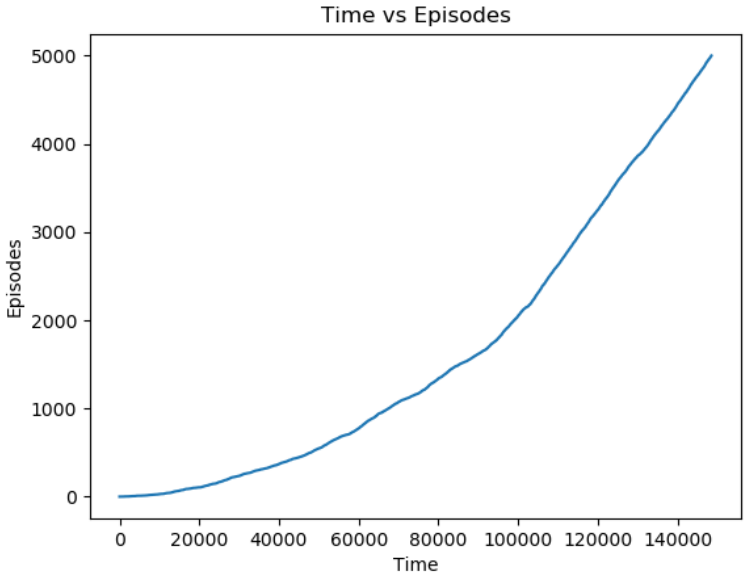
Exercise 6.10 (Programming):

Stochastic Wind:

Code : Check Github

Results:





Exercise 6.11:

In the SARSA TD target, we bootstrap from the estimate of the next state action pair which is determined by the behaviour policy which is generating the episode. In Q-learning however, we bootstrap from the maximum of the Q value of the next state over all actions. Therefore, we are bootstrapping from the state action pair which is determined from a target policy which is completely greedy. That is why Q-Learning is considered an off-policy method.

Exercise 6.12:

If action selection is greedy, both Q-Learning and SARSA seem to make the same updates and follow the same trajectory. However, both algorithms are fundamentally different in the sense that SARSA uses the already generated next action to bootstrap from while Q-Learning does not generate the next action until the current state action pair is updated.

Exercise 6.13:

With 0.5 Probability,

$$Q_1(s_t) \leftarrow Q_1(s_t) + \alpha * [R_{t+1} + \gamma * \sum_a \pi(a|s_{t+1}) Q_2(s_{t+1}) - Q_1(s_t)]$$

else:

$$Q_2(s_t) \leftarrow Q_2(s_t) + \alpha * [R_{t+1} + \gamma * \sum_a \pi(a|s_{t+1}) Q_1(s_{t+1}) - Q_2(s_t)]$$

Where $\pi(s)$ is $\varepsilon - greedy$ with respect to $Q_1 + Q_2$

Exercise 6.14:

In the car rental problem, the state was the number of cars at each location and the action was the number of cars moved between the two locations. Therefore the after-state will be the number of cars at both locations in the next morning. Several configurations of state, action will lead to the same after-state and thus, using after-states will be more computationally efficient.