Assignment 5 Report for Part B
Billy Lin

Epsilon-greedy Implementation

The epsilon-greedy Q-learning with a fixed-epsilon has an equal balance between exploration and exploitation. This learning becomes not intelligent when we have experienced through a substantial number of iterations. However, if we use a custom-adjusted epsilon, we can adjust our epsilon over time to improve our efficiency. In my implementation, I used an exponential decay function with a small decaying rate $\lambda = 0.001$. In this way, $\varepsilon = e^{-0.001n}$. This equation has an advantage to keep my epsilon in the range between 0 and 1 and another advantage to shift our balance toward more exploitation than exploration as the epsilon decreases fast in the beginning and slows down after a few iterations. Using the custom-adjusted epsilon, I can almost always obtain the optimal policy on the golden path under 1000 iterations, whereas when I used the fixed epsilon, I generally needed 4000 to obtain the golden path.

Exploration Function Implementation

To implement the exploration function, I used a dictionary to count the number of visits of each state. Similar to the epsilon-greedy implementation, I choose a random action in exploration and use q-values to choose the next action in exploitation. However, differently, I use the counts of the visits of the states to determine whether we want to explore or exploit. I then conducted an experiment to check when we should switch from exploration to exploitation. I chose three values to experiment, 10, 25, 50. These values are specifically chosen to represent three scenarios: using exploration more than exploitation (visit_count = 10), using exploration and exploitation equally (visit_count = 25), and using exploitation more than exploration (visit_count = 50). The experiments are proceeded with conditions of 2 disks, 20% Noise, one goal, 0 living reward, 0.9 discount rate, and 1000 iterations. The following table shows the results of the experiment.

Visit_count =	**10**	**10**	**25**	**25**	**50**	**50**
# of episodes	24	50	66	97	84	57
# of times using random action (Exploration)	180	330	453	458	645	572
# of times using q_values (Exploitation)	796	620	481	445	271	371
Ratio of exploration to exploitation	0.226	0.53	0.94	1.02	2.38	1.54
Able to derive optimal policy on golden path	Yes	No	No	No	Yes	Yes

From the table, we can see that we should use visit_count = 50 as it was able to obtain the golden path as the optimal policy after 1000 iterations. This makes sense because we want to rely more on the exploitation when we know more about a state.