CS 5180: Exercise 4
Harin Kumar Nallaguntla

1.a. The straightforward modification involves eliminating the "Returns list" and updating the value of V(s) as follows:

$$V(s) \leftarrow V(s) + ((G_t - V(s)) / N(s)$$

b. We need to add code for incremental averaging which gives us the following equation and it needs to be replaced in the pseudocode:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + ((G - Q(S_t, A_t)) / N(S_t, A_t)$$

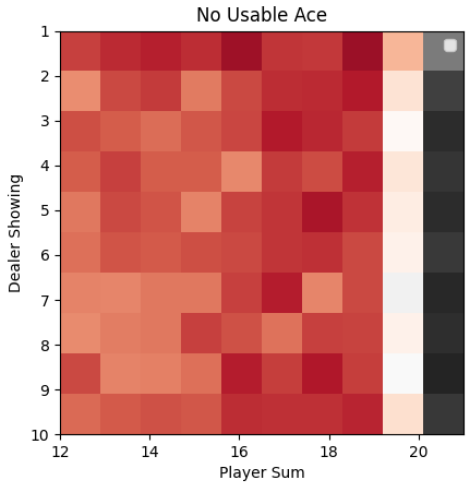
2.a. In the case of the Every-Visit Monte Carlo method, training cycles become faster because the average is computed over a broader range of rewards with discounts. However, since the discount factor γ is set to 1 for this specific game, there won't be any distinction between the First-Visit Monte Carlo and Every-Visit Monte Carlo methods in this game.

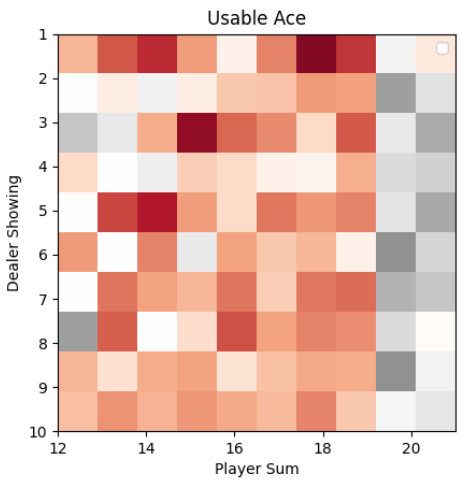
b. Upon analyzing the first-visit algorithm, we observe that the value of first visit is 1. Conversely, when implementing a quick script for the every-visit method, we obtain a value of 5.5.

```
>>> g = 0
>>> rewards = []
>>> r = 1
>>> n_steps = 10
>>> for _ in range(n_steps):
...     g += r
...     rewards.append(g)
...
>>> sum(rewards) / n_steps
5.5
```

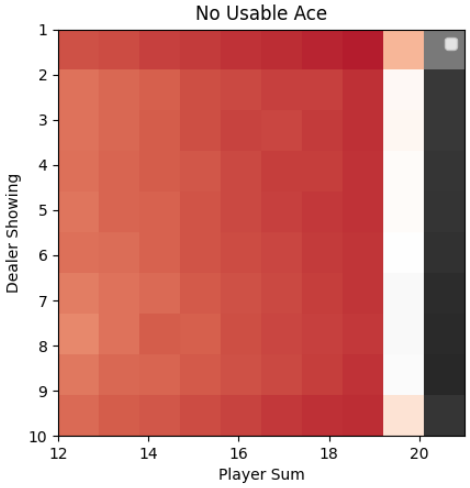
c. The variance of the estimator would consistently be infinite, as the expected return remains at 1 for each visit to the state. The primary disparity between first-visit and every-visit Monte Carlo methods in this context is that the number of terms in the sum increases proportionally to the number of visits to the state, and this increment, denoted by α, perpetually extends to infinity.

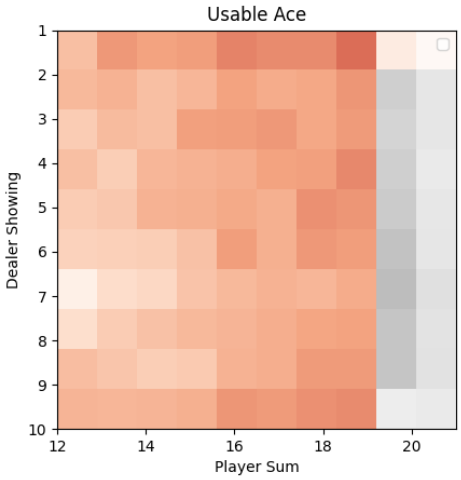
3.a. After 10k epidoes:



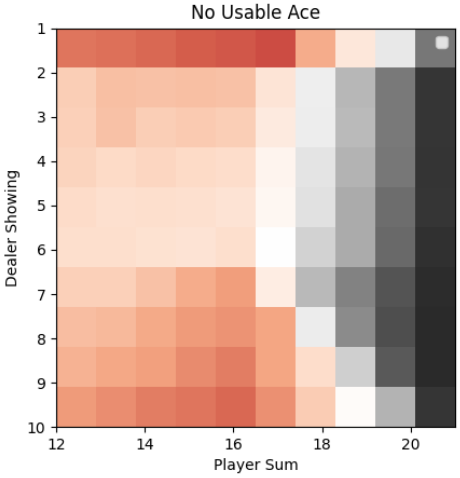


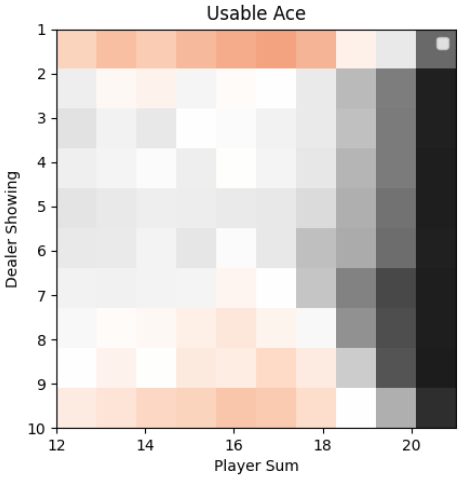
After 500k episodes:



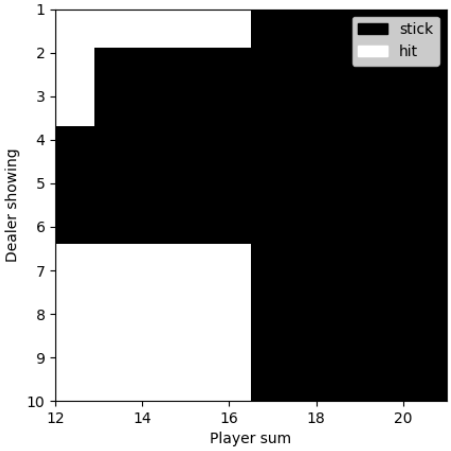


b. v^*:

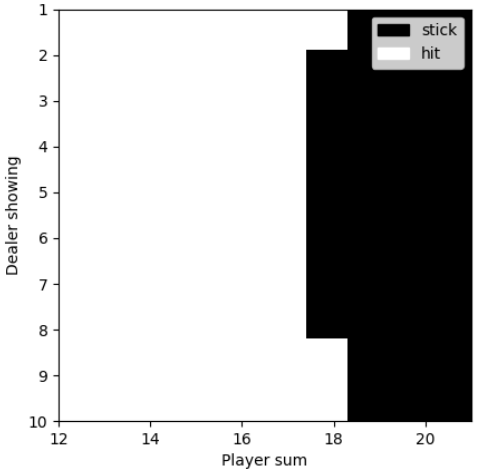




π^*:

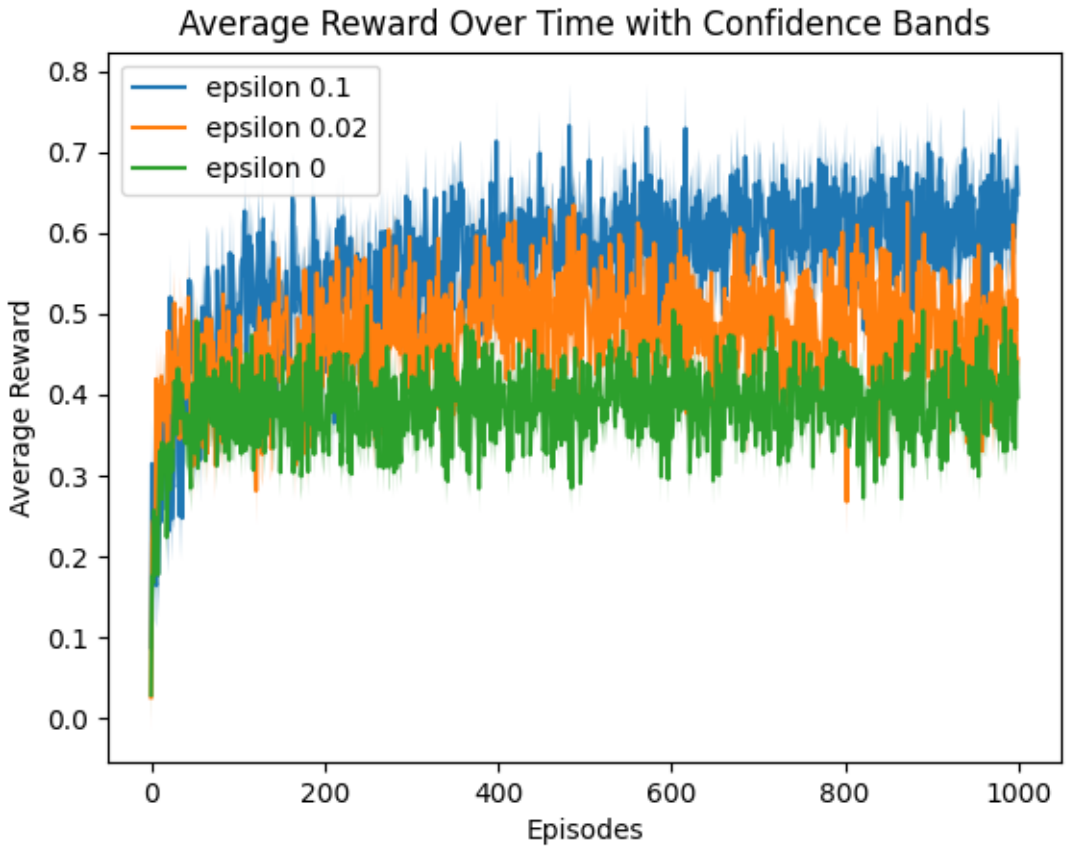


No Usable Ace

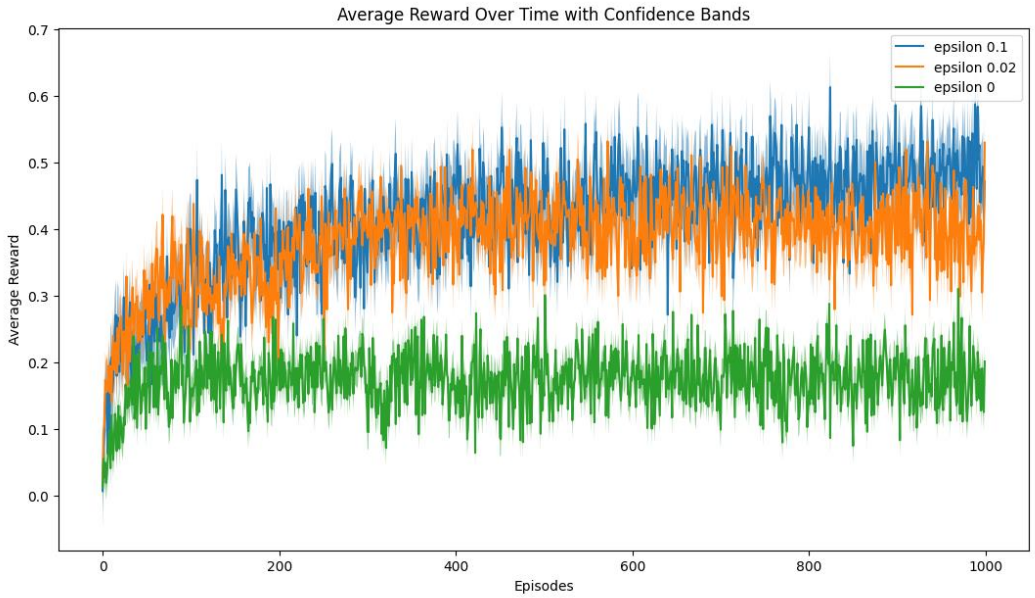


Usable Ace

4.a. This is the plot for goal position = (8, 4), as we can see the average reward increases as the episodes go on. From this, we can say that it learns to reach random goal positions (used a timeout of 459)



b. (used a timeout of 459)



c. In Monte Carlo Exploring Starts (ES), setting ε to 0 enforces a rigid, deterministic policy, eliminating room for exploration. Embracing exploration is crucial, as without it, the agent can get stuck in local optima, miss global solutions, and fail to learn across the entire state space. Lack of exploration may lead to suboptimal policies and inaccuracies. Furthermore, exploration fosters generalization, adaptability, and effective problem-solving. Omitting exploration hinders learning and limits the agent's ability to tackle complex challenges and attain optimal policies.

5.a.

There are a few things here that C_n represents the cumulative weight till now such that $C_{n+1} = C_n + W_{n+1}$

$$V_n = \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}$$

$$V_{n+1} = \frac{\sum_{k=1}^{n} W_k G_k}{\sum_{k=1}^{n} W_k}$$

$$= \frac{1}{W_n + \sum_{k=1}^{n-1} W_k} \left(W_n G_n + \sum_{k=1}^{n-1} W_k G_k\right)$$

$$= \frac{W_n G_n}{C_n} + \frac{\sum_{k=1}^{n-1} W_k G_k}{C_n}$$

$$= \frac{W_n G_n}{C_n} + \frac{\sum_{k=1}^{n} W_k}{C_n} V_n$$

$$= \frac{W_n G_n}{C_n} + \frac{C_n - W_n}{C_n} V_n$$

$$V_{n+1} = V_n + \frac{W_n}{C_n}(G_n - V_n)$$

b. Due to our policy π(St) being deterministic and greedy, we exclusively encounter trajectories in which π(At|St) equals 1. Consequently, in the equation, the numerator is constantly set to 1.

6. I am solving the 6th question right now. I did not have time to do the plots