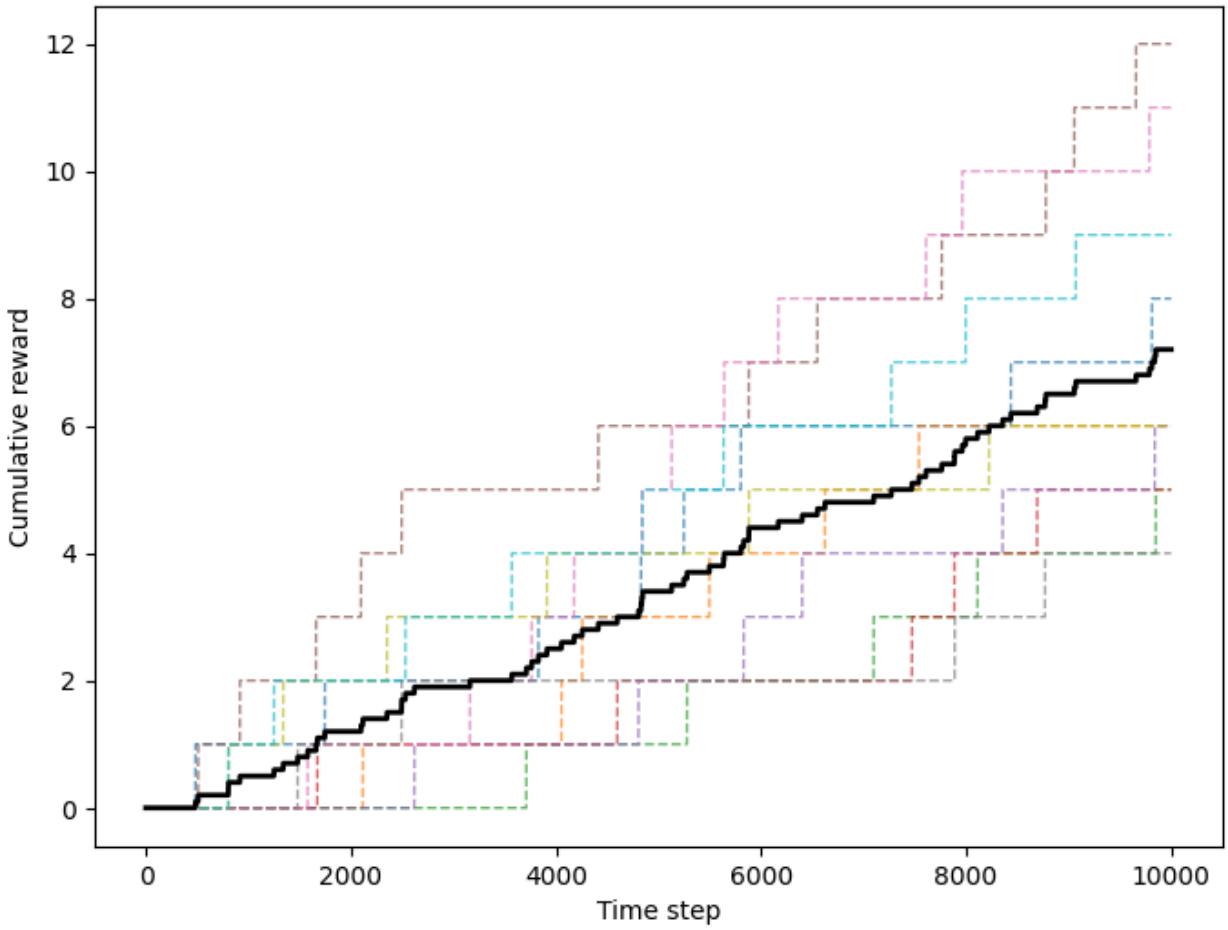
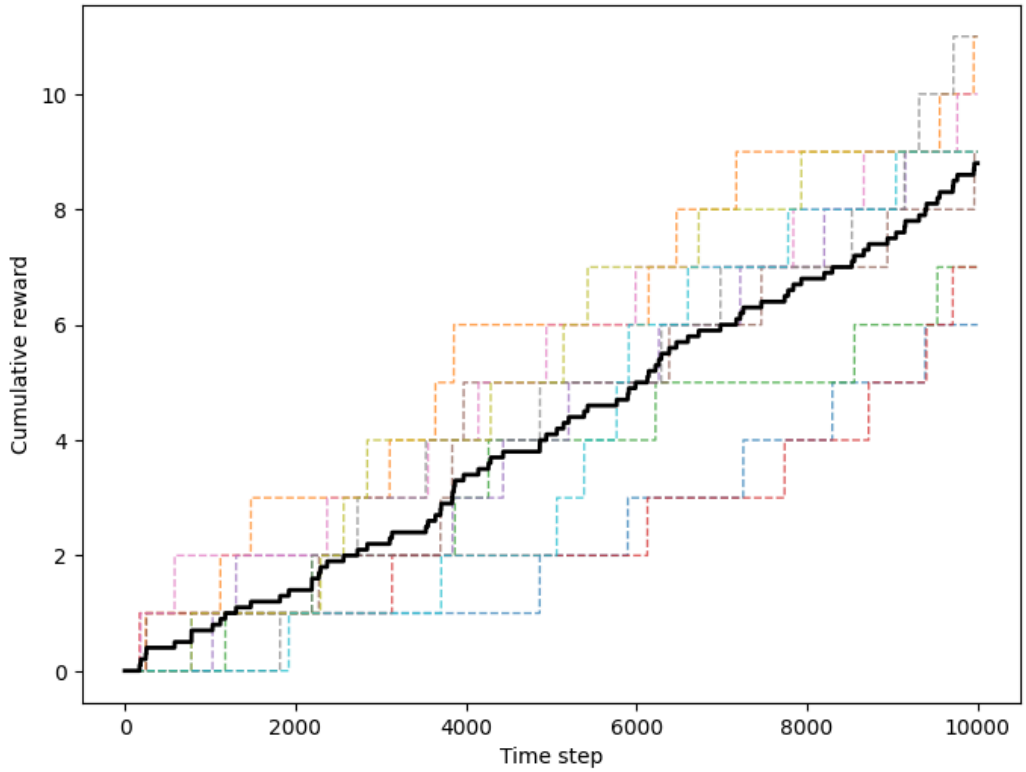
Miles Vollmer
RL Homework 1 EX0

Q3 Plot:

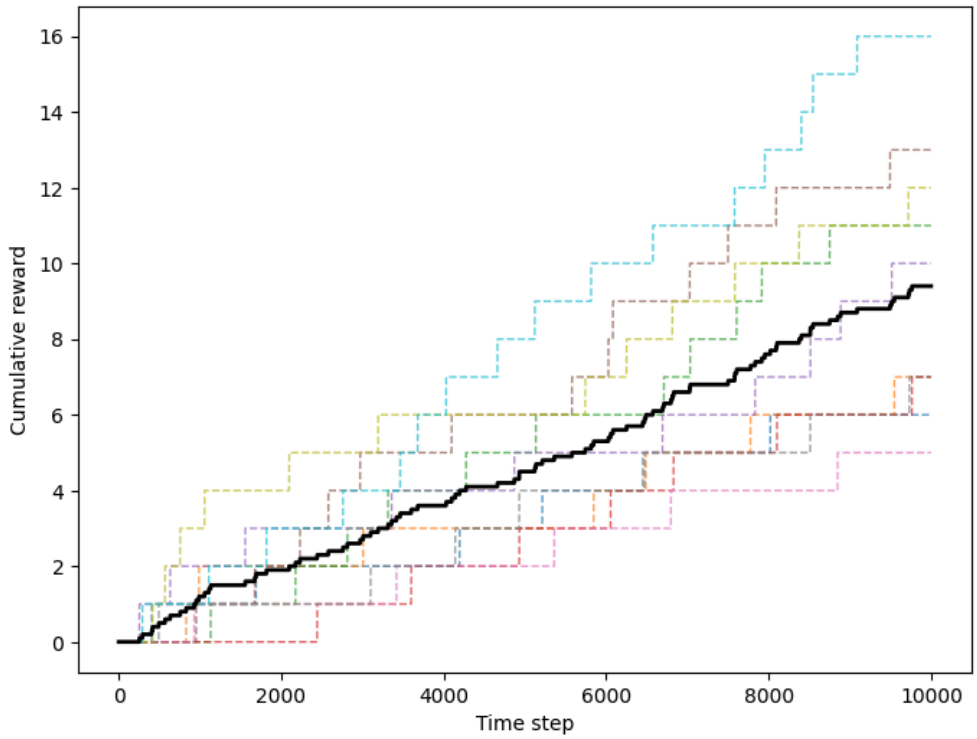


Q3 Written:
This strategy is not as strong as my manual policy. The main reason for this difference is there is strategy behind my manual policy, I try and move through the gates and get to the reward square as fast and efficiently as possible, I also have the advantage of knowing the reward state and location of the walls to formulate my strategy. Meanwhile the random policy has no strategy and simply brute forces its way to the end.

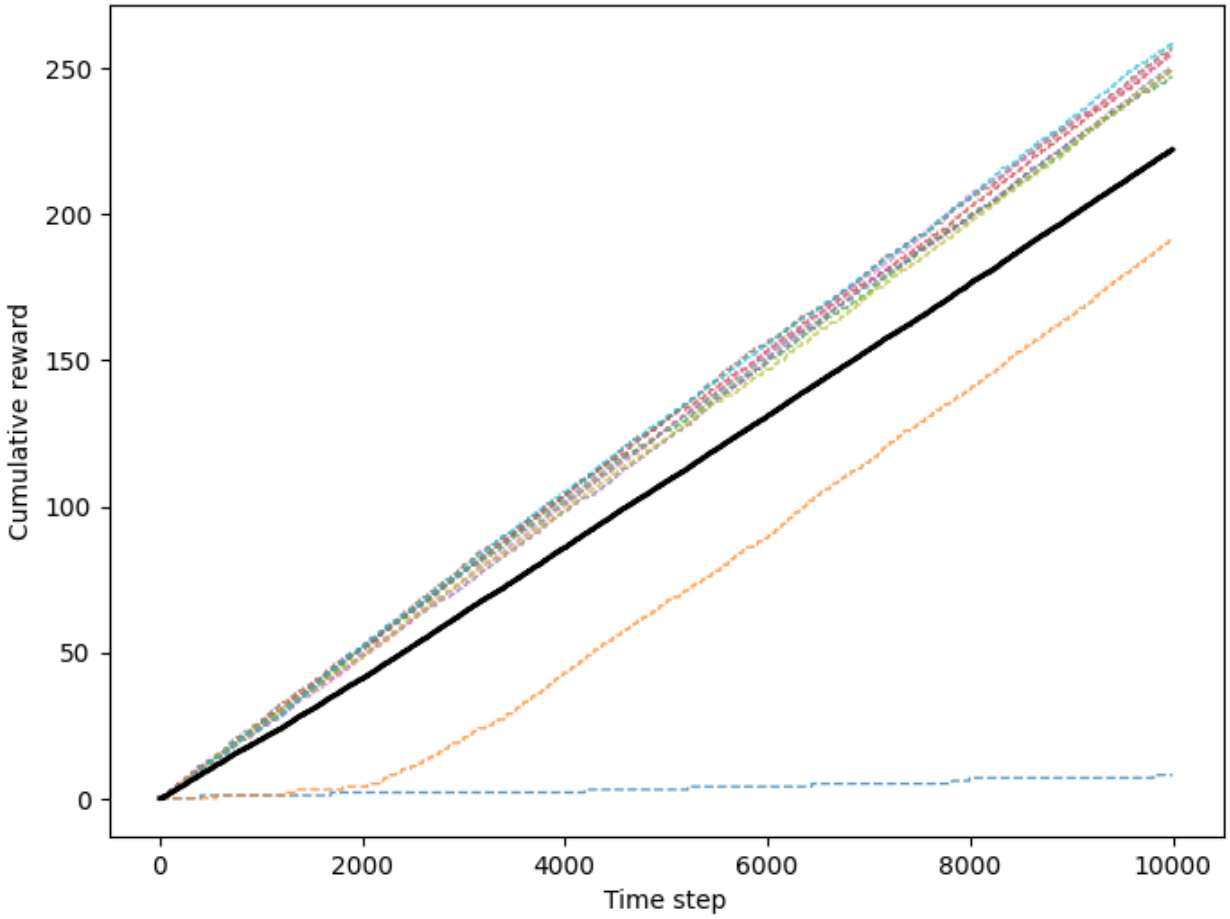
Q4 Plots:
Worse than random(No repeats):



About equal to random(E-Greedy):



Better than random(E-Greedy + Gamma):



Q4 Written:
The worse strategy is random selection but the same action cannot be taken twice in a row. The modified random policy is worse than a purely random policy in this scenario because it imposes an unnecessary constraint that limits exploration. In this environment, it is often optimal to repeat actions, such as moving multiple steps in the same direction to reach the goal. By preventing consecutive actions, the agent is forced to take suboptimal detours, increasing the time needed to reach the goal. This limitation lowers the chance of randomly taking an optimal path and thus reduces the policy's cumulative reward compared to a purely random approach.

The equally bad strategy is a E-Greedy action selection algorithm. It works by with probability E(epsilon) it chooses a random action (exploration), and with probability 1 - E it selects the "best" action. It calculates the best option by calculating an incremental estimate of the Q-Value for each action at each state by summing the rewards from each action at the state and dividing that by the number of times the action was taken in that state. When the maximum QValue is 0(ie. All actions have a QValue of 0 at the given state) a random action is choosen. This algorithm works about as well as the random one as it is not considering future rewards, so only states directly next to the goal state will have any actions with Q-Values above 0. This means it

selects randomly in all states except for the ones directly next to the goal, where it will try to move to the goal with probability 1 - E. Making their performances almost identical. I choose an E value of 0.1 for this to limit exploration, as it will already be taking many random actions due to the issue with almost all the QValues being 0, so for the situations where it has calculated QValues > 0 it uses its calculation instead of the random option 90% of the time.

The better strategy is a E-greedy algorithm with gamma. This performs better than a purely random policy due to its strategic balance between exploration and exploitation coupled with its focus on long-term rewards. This algorithm explores different actions with a probability E, ensuring the agent tries new actions, while also exploiting the best-known actions(with probability 1 - E) based on past experience to optimize performance. The gamma parameter is crucial to this, as it allows the agent to consider not only immediate rewards(like in the previous algorithm) but also future rewards, encouraging actions that may lead to greater cumulative success over time. This approach leads to more efficient learning and decision-making, in this environment where planning and foresight are necessary for optimal performance. Unlike the random policy, which brute forces its way to the goal, the E-greedy algorithm with gamma is able leverage future rewards to make informed decisions on what actions to take. I chose a E value of 0.3 and a Gamma of 0.99 for this. I choose an E value of 0.3 because I wanted to encourage enough exploration to discover more possible paths that could be optimal. I chose a Gamma of 0.99 because I wanted the QValues to be heavily weighted towards future rewards as there is only one state with a reward so finding the best path to that state is crucial to finding the most optimal paths.