RL HW EX6:

1. Planning vs. n-step returns.
    a. The nonplanning method looks particularly poor in Figure 8.3 because it is a one-step method; a method using multi-step bootstrapping would do better. Do you think one of the multi-step bootstrapping methods from Chapter 7 could do as well as the Dyna method? Explain why or why not.
        i. No, I wouldn't expect a multi-step bootstrapping (MSB) method from Chapter 7 to perform as well as Dyna. MSB methods rely solely on real experience and require more iterations to learn effectively. In contrast, Dyna's planning component allows it to propagate value updates quickly by generating simulated experiences, significantly reducing the sample complexity. This ability to update values without needing additional real experiences gives Dyna a distinct advantage in learning efficiency.
    b. See the pseudocode on page 164 in RL2e. Consider using both n-step returns and Dyna (specifically, using n-step returns instead of one-step returns in tabular Dyna-Q). We can clearly do this for the learning phase (d). Can we also do this for the planning phase (f)? What are the advantages and disadvantages of using n-step returns in the planning phase (f)?
        i. Yes, n-step returns can be used in the planning phase (f) of Dyna-Q, allowing rewards to propagate over longer distances, which could speed up learning in tasks with delayed rewards. However, this approach also could also hamper the planning phase as it increases computational cost and introduces higher variance, which may affect stability and slow down the algorithm.
2. Implementing Dyna-Q and Dyna-Q+.
    a. For some unknown reason, the textbook is quite vague about Dyna-Q+ and does not provide pseudocode for the modification. Read Section 8.3 carefully (specifically p. 168) and reconstruct the pseudocode for Dyna-

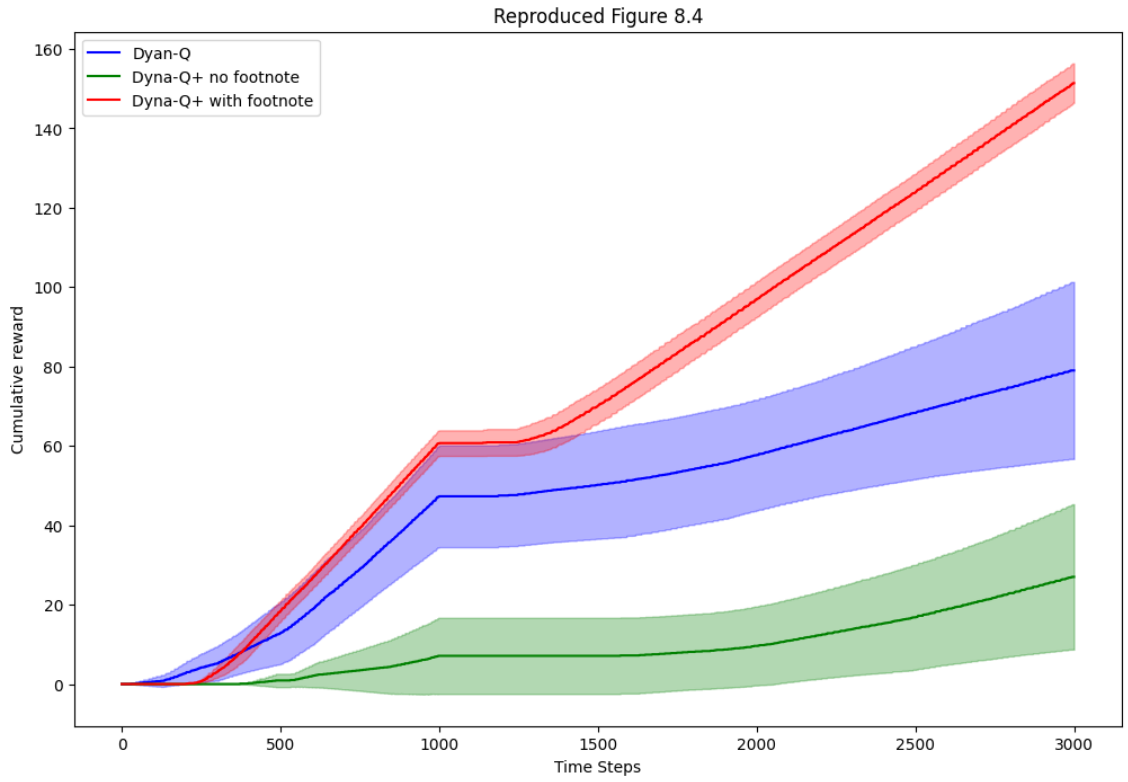
Q+. In particular, please write your pseudocode that also contains the suggestion from the footnote.

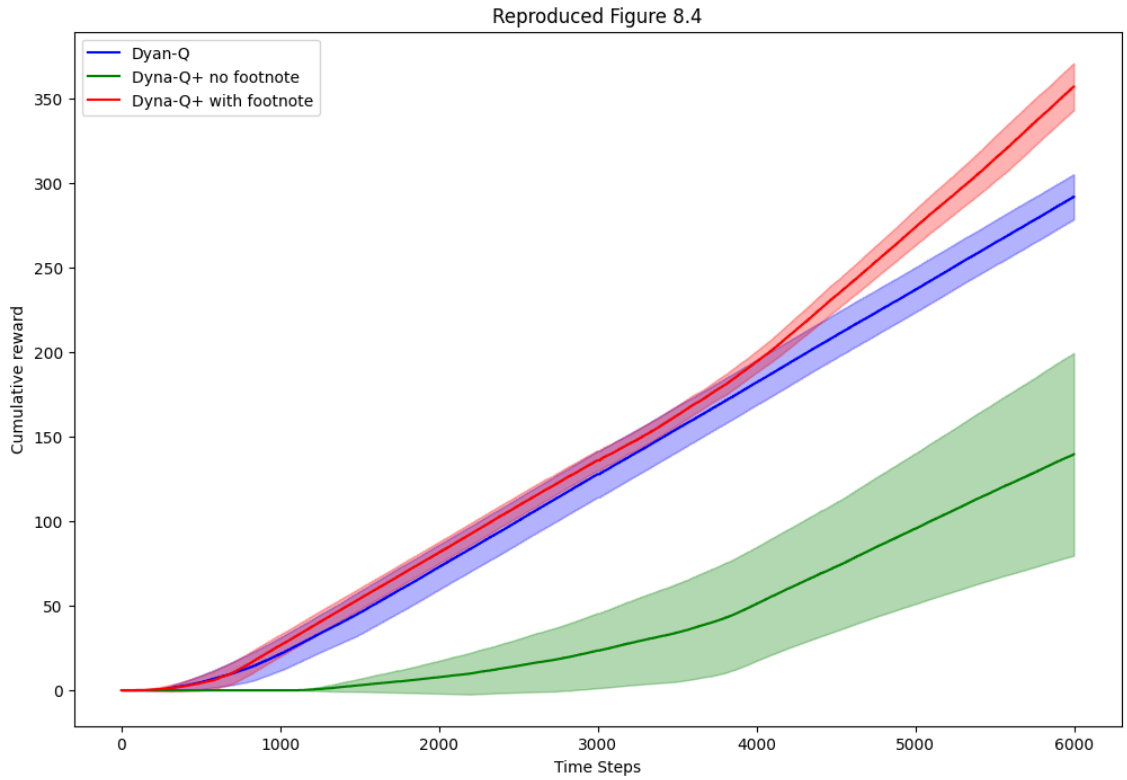
i.

```
Dyna-Q+
Initialize Q(s,a) and Model(s, a) for all s in S and a in

# Store T values for the bonus
Initialize LastTried(s, a) to 0 for all s, a

# small k for exploration bonus
Set k > 0
Loop forever:
(a) S = current (nonterminal) state
(b) A = e-greedy(S,Q)
(c) Take action A; observe resultant reward, R, and state,
(d) Q(S, A) = Q(S, A) + alpha[R + gamma * maxa Q(S', a) -
(e) Model(S, A) = R, S'(assuming deterministic environment

# Just tried to reset T
(new step) LastTried(s, a) = 0
(f) Loop repeat n times:
    S = random previously observed state

    # action not yet tried allowed to be considered in thi
    A = random action from A(S)

    # initial model for untried action leads back to same
    if (S, A) has never been tried:
      Model(S, A) = (0, S)

    R, S' = Model (S, A)

    # Exploration bonus
    bonus = k * sqrt(TimeSinceLastVisit(S, A))
```

```
    Q(S, A) = Q(S, A) + alpha[(R + bonus) + gamma * maxa (

    # Just tried to reset T
    LastTried(S, A) = 0

# Took a step so add 1 to all
(new step) LastTried += 1
```

b. Plot:





c. Did the footnote matter? Why or why not?

   i. Yes the footnote does matter as with the dyna-q+ we add a bonus reward that promotes exploration. Specifically, it promotes choosing untried actions. This is where the footnote comes in handy as without the footnote, untried actions in a state are ignored. This footnote makes a major difference in the changing environments we were looking at, as it prevents the agent from getting stuck in familiar but suboptimal actions, leading to faster discovery of optimal actions and improved performance.

3. Think carefully about the implications of applying bonuses to the reward function (“pseudo-reward”) versus during action selection. Make some predictions:

   a. What would generally happen to the policy’s behavior during the training under each approach? How do they differ from each other?

      i. When using exploration bonuses as a pseudo-reward (Dyna-Q+ style), the bonus directly impacts the Q-values by integrating into the simulated rewards, leading the learned policy to inherently favor

persistent exploration. This approach results in a policy that continuously balances exploration and exploitation, as actions with higher bonuses(even if they yield lower actual rewards) are prioritized. In contrast, when bonuses are applied only during action selection (UCB style), they temporarily influence which actions are chosen without altering the Q-values. This setup drives short-term exploration but allows the agent to gradually shift towards exploitation as it becomes more confident in the true reward structure, leading to reduced exploration once an optimal policy emerges. Thus, the key difference is that pseudo-reward bonuses embed exploration into the learned policy itself, while action selection bonuses encourage early exploration but ultimately favor exploitation.

b. What are the advantages and disadvantages of each approach?

   i. The pseudo-reward approach has the advantage of promoting ongoing exploration by embedding the bonus into Q-values, making it particularly effective in dynamic environments where things change, like in the blocking and shortcut maze examples. This lets the agent continually explore and adapt to new paths or obstacles. The downside to this is that it can distort the true value estimates, leading the agent to sometimes over-prioritize exploration over finding the actual optimal path in stable settings. In contrast, the action selection approach has an advantage in that it keeps Q-values accurate, encouraging short-term exploration that naturally shifts to exploitation, which is advantageous in stationary environments. However, in complex or changing settings, this approach can lead to exploration tapering off too quickly, causing the agent to miss out on learning better routes when the environment shifts.

5.

2.05

$\frac{2}{2}+\sqrt{\frac{2\ln 3}{2}} + \frac{0}{1}+\sqrt{\frac{2\ln 3}{1}}$

Iteration 4

Selection : Ⓒ $\xrightarrow[R]{}$ Ⓓ

$\max\left(\frac{T_r}{N_r}+\sqrt{\frac{2\ln N_c}{N_r}}, \frac{T_L}{N_L}+\sqrt{\frac{2\ln N_c}{N_L}}\right)$

Epansion : (no use md)

Iteration 4

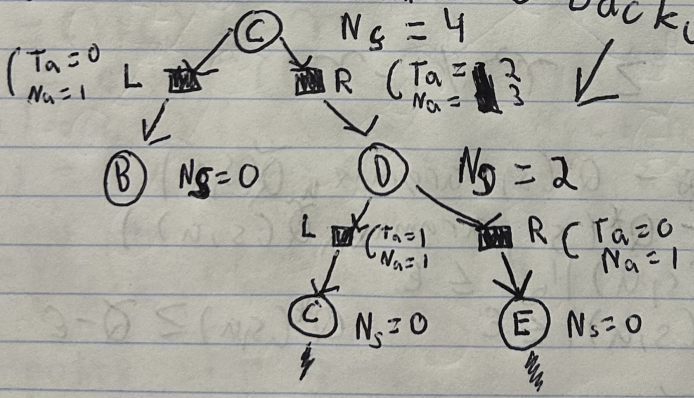
Selection: $\max_a\left(\frac{T_a}{N_a}+\sqrt{\frac{2\ln N_c}{N_r}}\right)$

$\max\left(R: (2/2)+\sqrt{2\ln(3)/2},\ L: (0/1)+\sqrt{2\ln(3)/1}\right)$

$\max(R: 2.05,\ L: 1.48) = R$

Ⓒ $\xrightarrow[R]{}$ Ⓓ

Expansion: (no use md) → Ⓔ Backup



Simulation: Ⓔ $\xrightarrow{L}$ Ⓓ $\xrightarrow{L}$ Ⓒ $\xrightarrow{R}$ Ⓓ $\xrightarrow{L}$ Ⓒ $\xrightarrow{R}$ Ⓓ $\xrightarrow{R}$ Ⓔ $\xrightarrow{L}$ Ⓓ $\xrightarrow{L}$ Ⓒ $\xrightarrow{L}$ Ⓑ

$\xrightarrow{R}$ [redacted] $\xrightarrow{R}$ Ⓓ $\xrightarrow{L}$ Ⓒ $\xrightarrow{L}$ Ⓑ $\xrightarrow{R}$ Ⓒ $\xrightarrow{R}$ Ⓓ $\xrightarrow{L}$ Ⓒ $\xrightarrow{L}$ Ⓑ $\xrightarrow{L}$ Ⓐ

$\xrightarrow{L}$ Term reward 0

Iteration 5:

Selection : C; max (R: $2/3+\sqrt{2\ln(4)/3}$, L: $\sqrt{2\ln(4)}$)

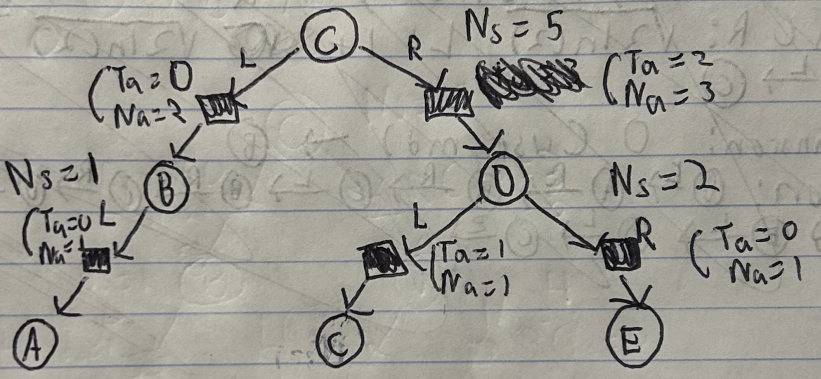
max (R: 1.63, L: 1.66) = L

(C) $\xrightarrow{L}$ (B)

Expansion: O (use rnd) → (A)

Simulation : (A) $\xrightarrow{L}$ Terminal reward 0

Back up



Iteration 6

selection: C: max(R: $2/3 + \sqrt{2\ln(5)/3}$, L: $\sqrt{2\ln(5)/2}$

max(R: 1.70, L: 1.26) = R C → D

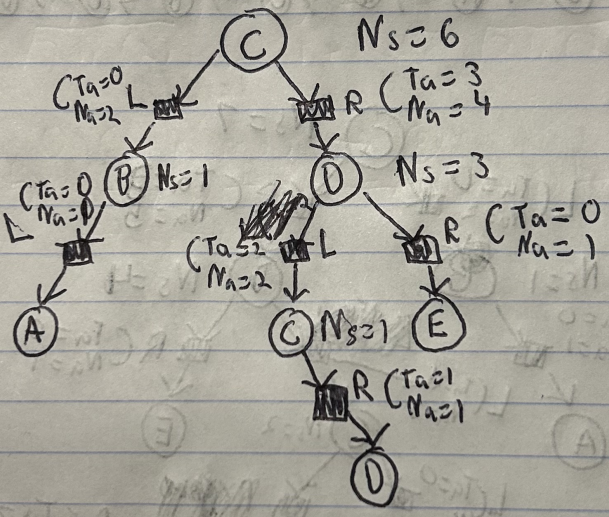
D: max(R: $\sqrt{2\ln(2)}$, L: $1 + \sqrt{2\ln(2)}$) = L

(C) R→ (D) L→ (C)

Expansion: 1 (use rnd) (C) R→ (D)

simulation: (D) R→ (E) L→ (D) R→ (E) R→ Terminal reward = 1

Backup



Iteration 7

Selection: C: max(R: 3/4 + $\sqrt{2\ln(6)/4}$, L: $\sqrt{2\ln(6)/2}$)

max(R: 1.70, L: 1.26) = R C → D

D: max(R: $\sqrt{2\ln(3)}$, L: 2/2 + $\sqrt{2\ln(3)/2}$)

max(R: 1.48, L = 2.05) = L

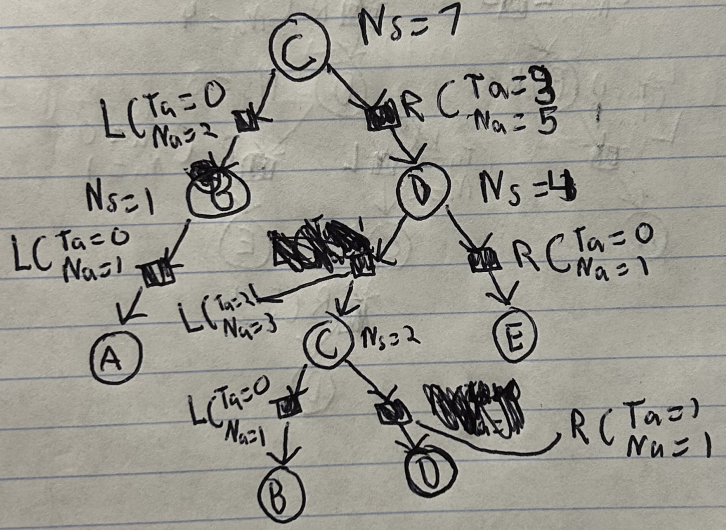
C → D → C

Expansion: O (no rnd) C → B

Simulation: B → A → B → C → B → C → B → A → Term

reward = 0

Backup



Iteration 8

Selection: C: max(R: $3/5 + \sqrt{2\ln(7)/5}$, L: $\sqrt{2\ln(7)/2}$

max(R: 1.48, L: 1.39) = R C → D

D: max(R: $\sqrt{2(\ln(4))}$, L: $2/3 + \sqrt{2\ln(4)/3}$)

max(R: 1.67, L: 1.63) = R D → E C → D → E

Expansion: i (use rnd) E → Terminal