Q1)

a)

I do not think so, because in Dyna algorithms we are improving our policy and Q value even after learning (in planning) but in n-step returns algorithm we do not improve policy and Q value after learning so they will be slower than Dyna algorithm.

b)

The disadvantage of using n-step return in planning phase is that before n step we will not have any planning and it reduces the speed of algorithm. The advantage of n-step return is its error reduction property which we could have in planning too, so we will have more accurate planning. We could find an optimal n and alpha to have the best performance.

Q2)

In the first part of experiment, because Dyna + has much more exploration than Dyna it will find the narrow path earlier and its cumulative rewards increase faster. Dyna algorithm also could find the narrow path because it does exploration too but later than Dyna+. However, after a while because the environment is not being changed and because more exploration has a more cost, the cumulative rewards of Dyna+ gradually will reduce till reach Dyna's cumulative rewards so gap between these two will diminish gradually.

Q3)

a)

```
Initialize $Q(s, a)$ , $Model(s, a)$ and $\tau (s,a)$ for all $s \in S$, $a \in A(s)$ , $t=0$
Loop forever:
    (a) S current (nonterminal) state
    (b) $A \leftarrow \varepsilon$ -reedy(S, Q)
    (c) Take action A; observe resultant reward, R, and state, S'
    (d) $Q(S, A) \leftarrow Q(S, A) + \alpha[ R+ \gamma Max\ Q(s',a) - Q(s,a)]$
    (e) $Model(S, A) \leftarrow R, S'$ (assuming deterministic environment)
    (f) $t=t+1$
    (g) $\tau (S, A)=t$
    (h) Loop repeat n times:
        S random previously observed state
        A random action in S
        If A previously taken in S
            $Bonus(S,A)= \kappa \times sqrt\{| \tau (S, A) -t|\}$
            $R=R+ Bonus(S,A)$, $S' \leftarrow Model(S, A)$
        Otherwise
        $\tau (S, A)=t$
            $R=0$, $S'=S \leftarrow Model(S, A)$
        $Q(S, A) \leftarrow Q(S, A) + \alpha[ R+ \gamma Max_a Q(s',a) - Q(s,a)]$
```

For running simulation you need to run the main file containing:

Blocked_Maze()

Shortcut__Maze()

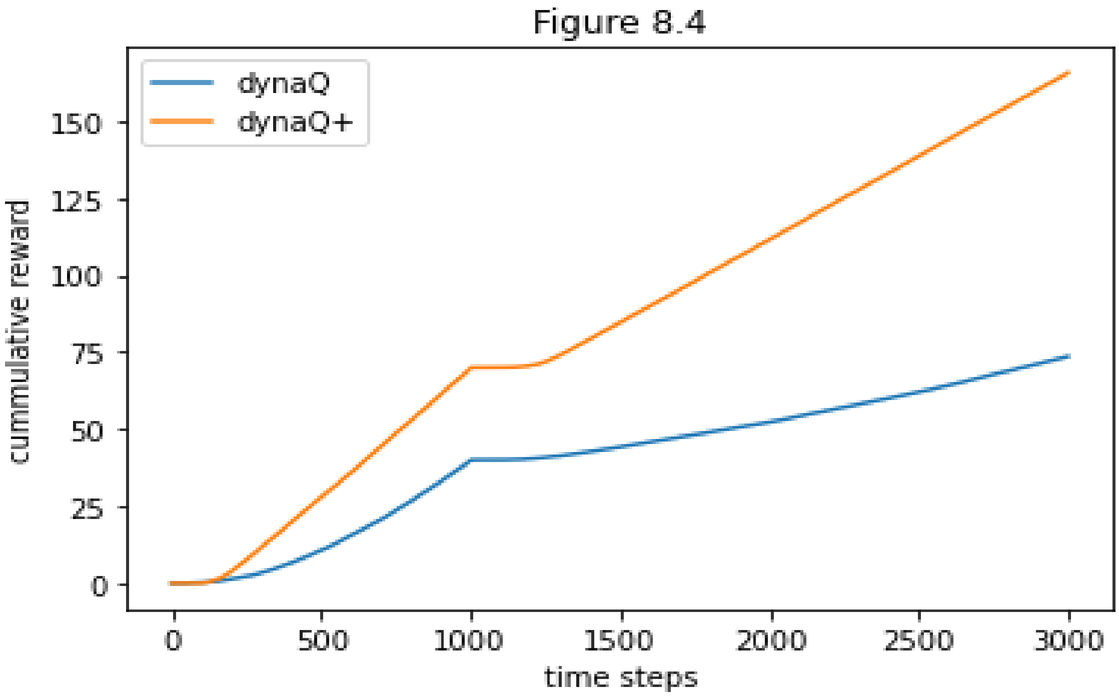
Blocked_Maze_no_footnote()

Shortcut__Maze_no_footnote()

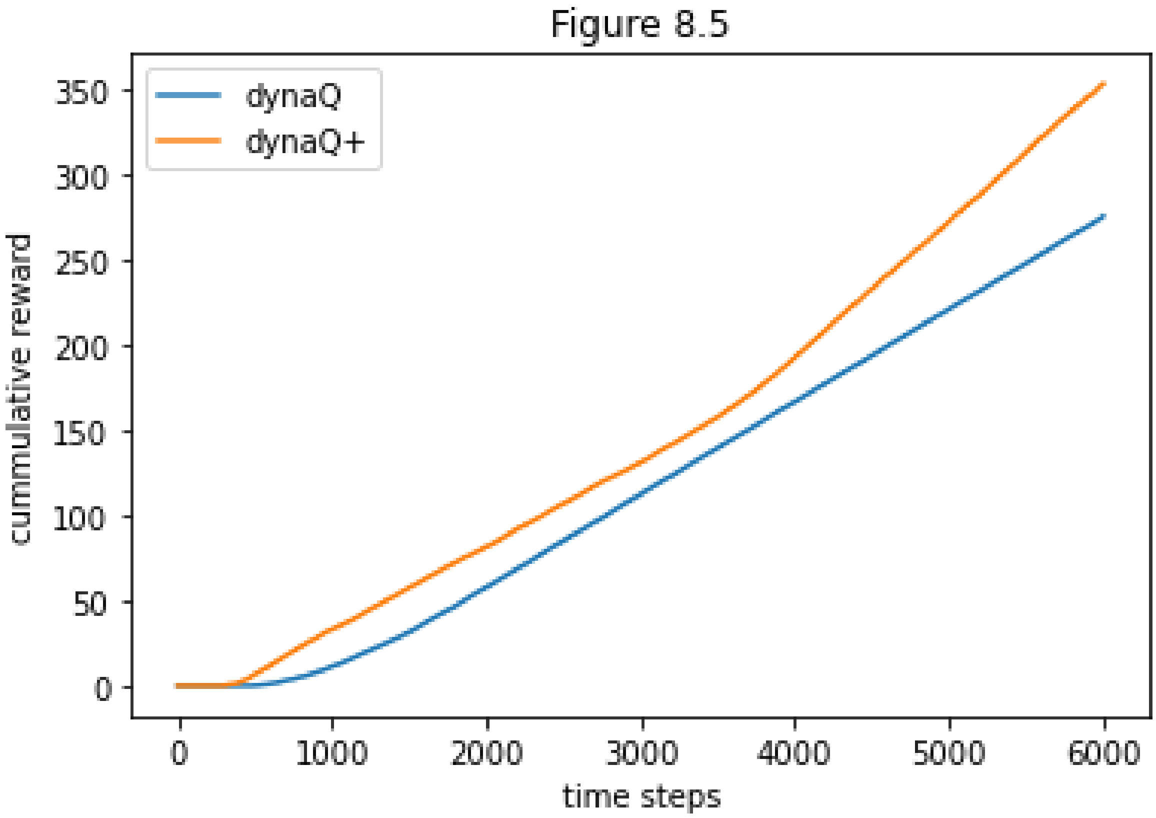
Which right now Blocked_Maze() is uncomentted

b)

κ=10e-4   n=100



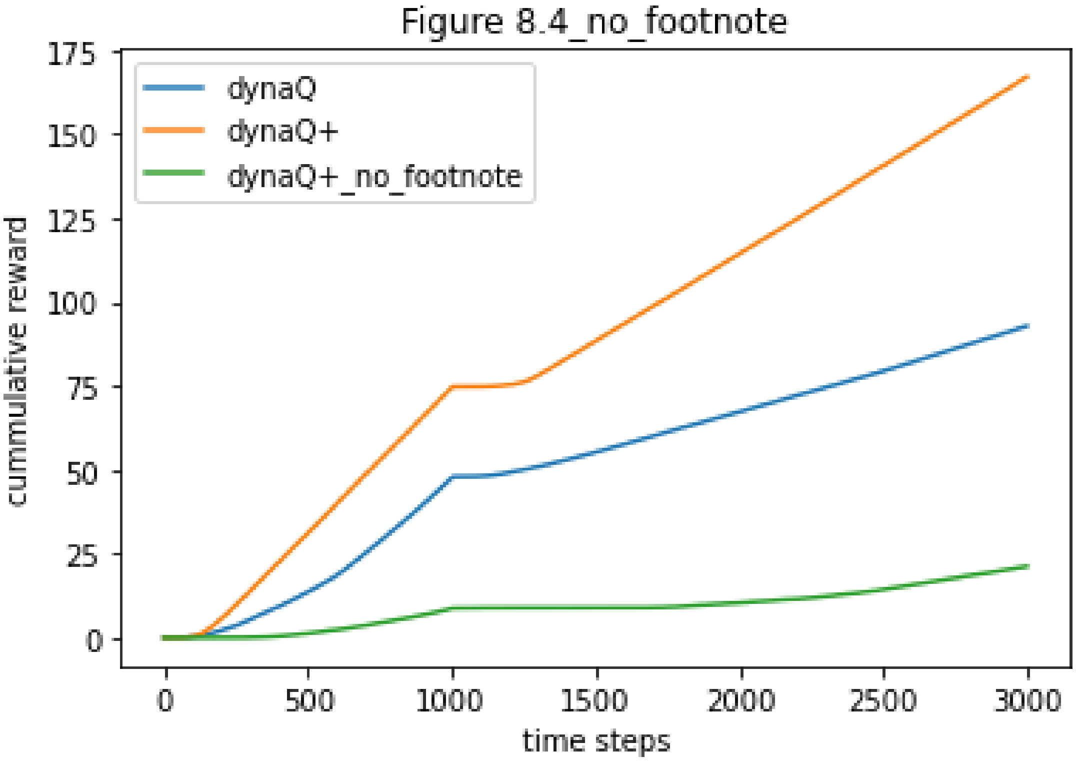
κ=10e-3 n=50



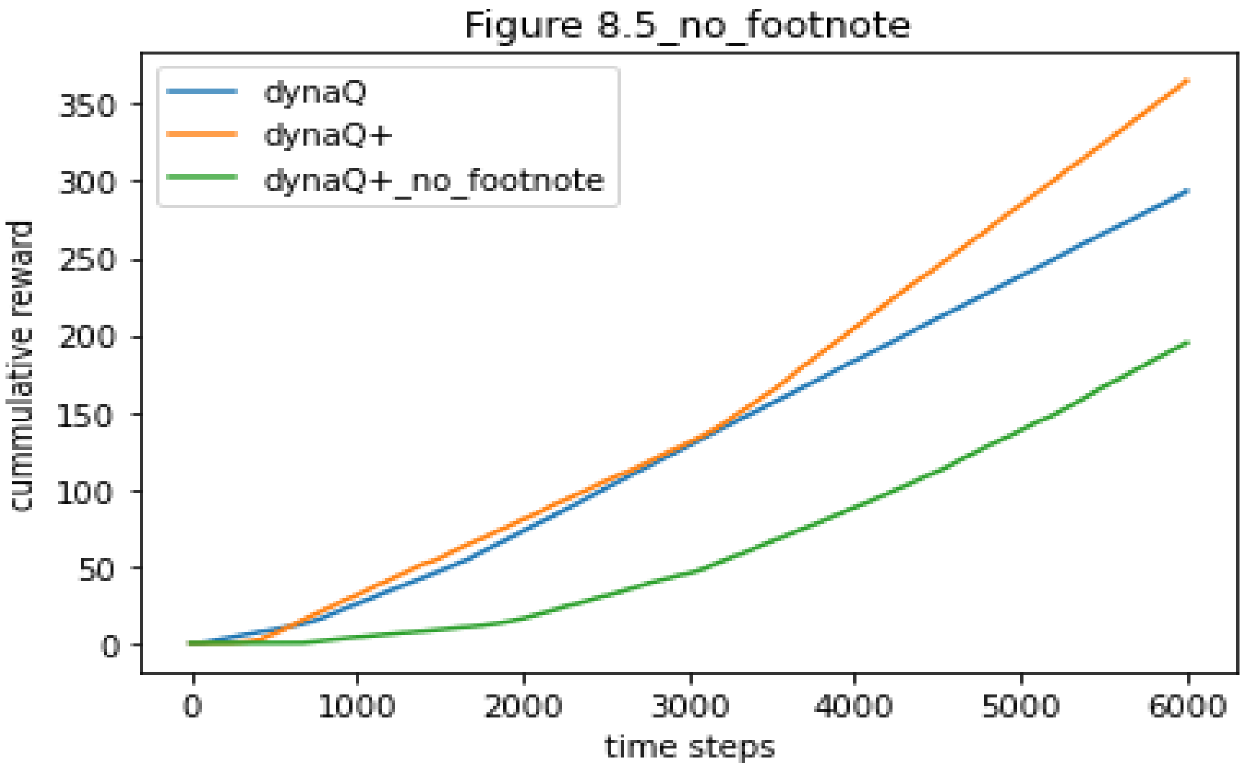
c)

Yes footnote is important because it is suggesting more exploration, exploring even not tried action in the state S. It is possible in an experience that an action in a state never being selected in ε -reedy selection, so by following footnote, it would be possible that his action be tried in planning and so we will have more exploration. Because this new action would be selected in the planning phase so we could not know its reward and transition (S') so we will consider its reward zero and also next state would be the current state. In the following, we could see that if we do not follow the footnote suggestion the result will not be a good one, because when we do not consider all actions in the state S and just give bonus to the actions which have been tied before in experiment, we are exploring too much in the areas which was not close to the optimal solution, so it causes that cumulative rewards growth slowly.

κ=10e-4   n=100



κ=10e-3   n=50



Q4)

In the case of applying bonuses to the reward function (model) we are changing the Q-value so if we have chosen a bad k it may affect the solution. In this case we may find not optimal policy and Q-value as the solution. But if we chose a good k then we could find changes in an environment because it would be able to map out areas unvisited for a long time. On the other hand, although using exploration bonus for actions resulted in faster learning in the initial phase, the exploration bonus might not stable enough to discover and exploit changes. The bonus affects only a single action selection and isn't able to map out areas unvisited for a long time.

Q 5)

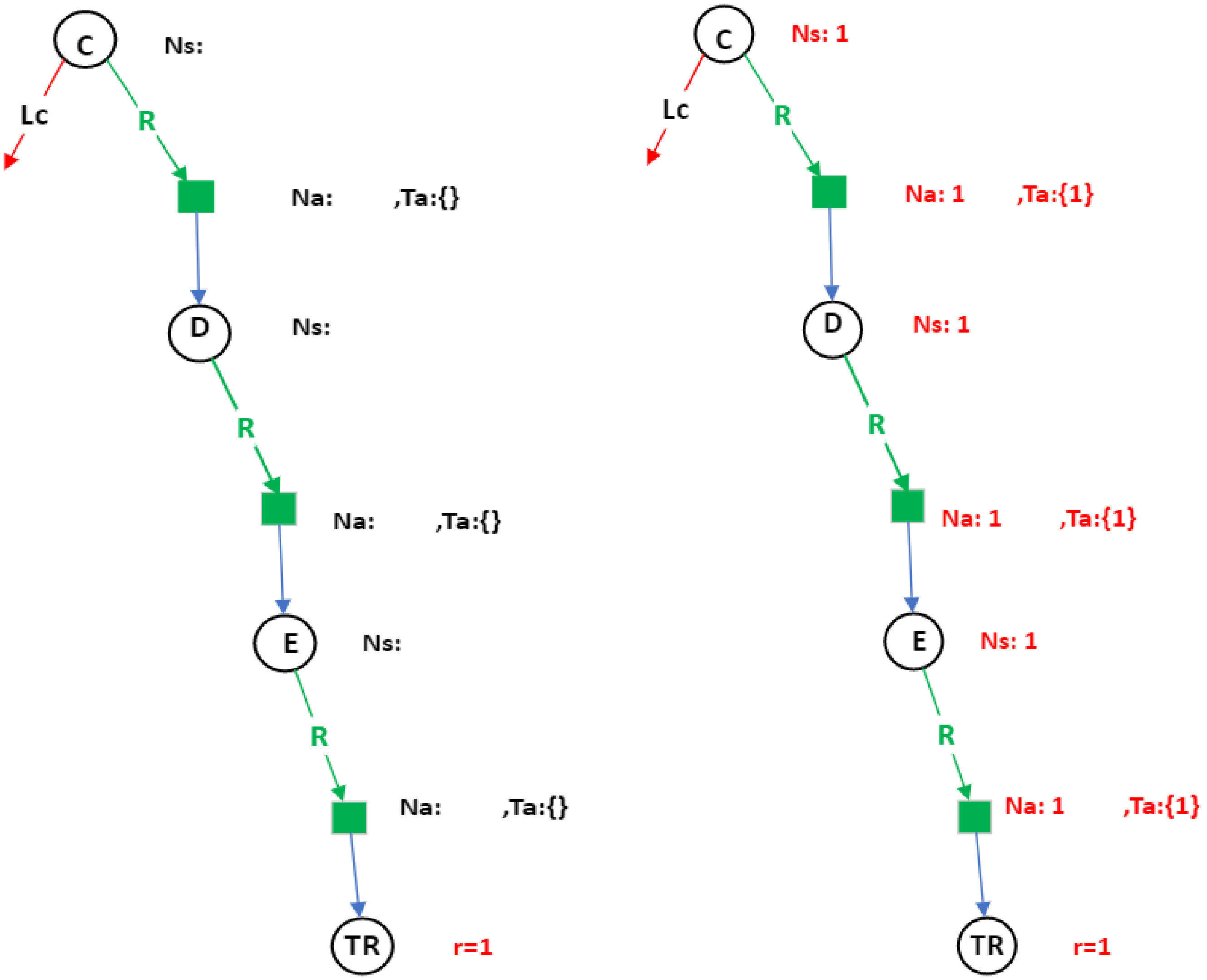
For stochastic environment, in part (e) of the algorithm, instead of updating R and S' directly, we should store samples of R and S' for each (S,A) pair. Then from which we can compute distributions and expectations. We will have something like P(R,S'|S,A) as a model.

However, it would perform poorly in changing environments because it would be bias toward earlier observations made in the unchanged environment. For changing environments, we could modify this by giving more weight to recent observations, or discounting past observations in the distribution. Also, we could use DynaQ+ algorithm for changing environments simultaneously.

Q6)

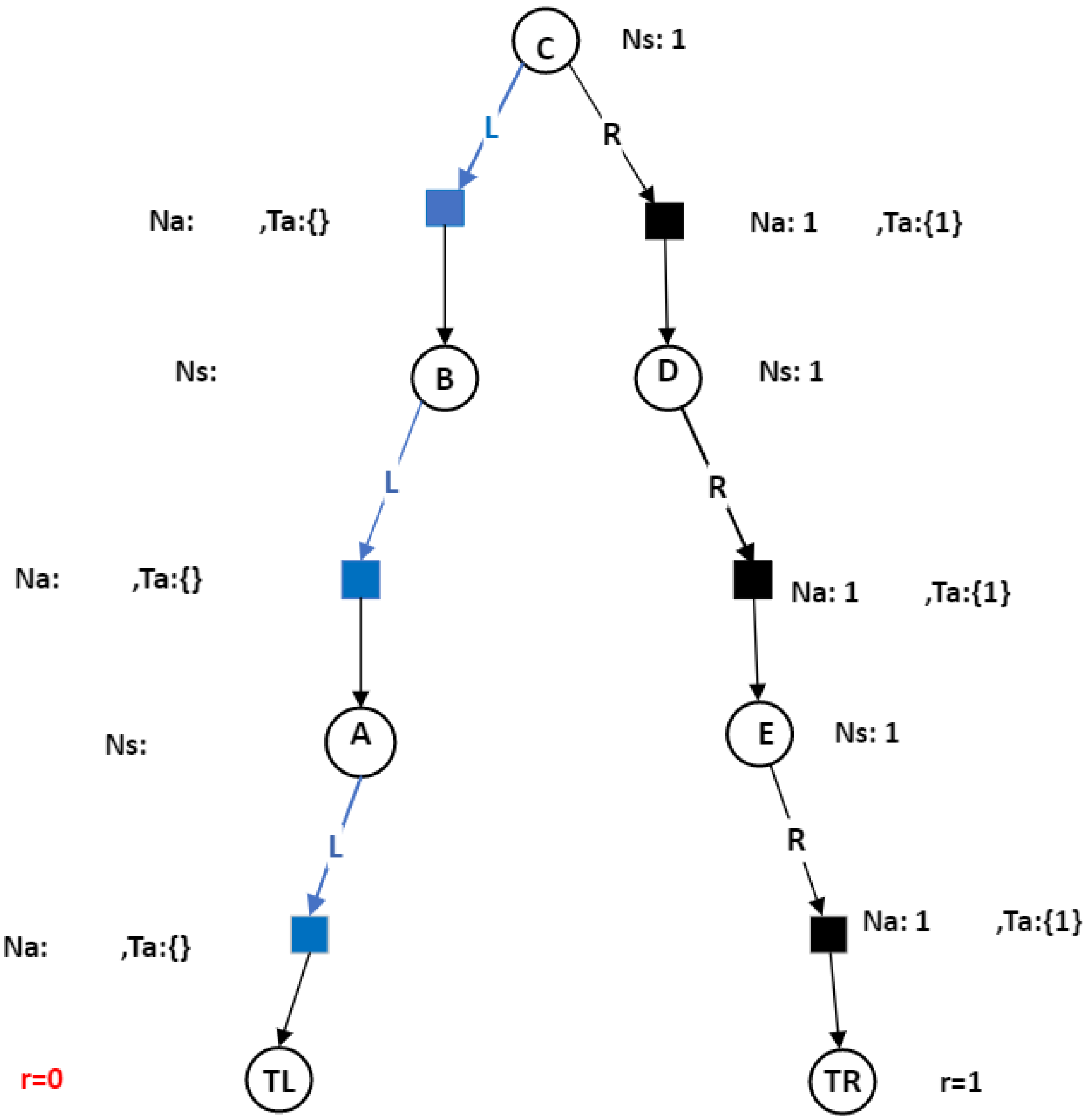
Episode 1:

Forward: **Backup:**

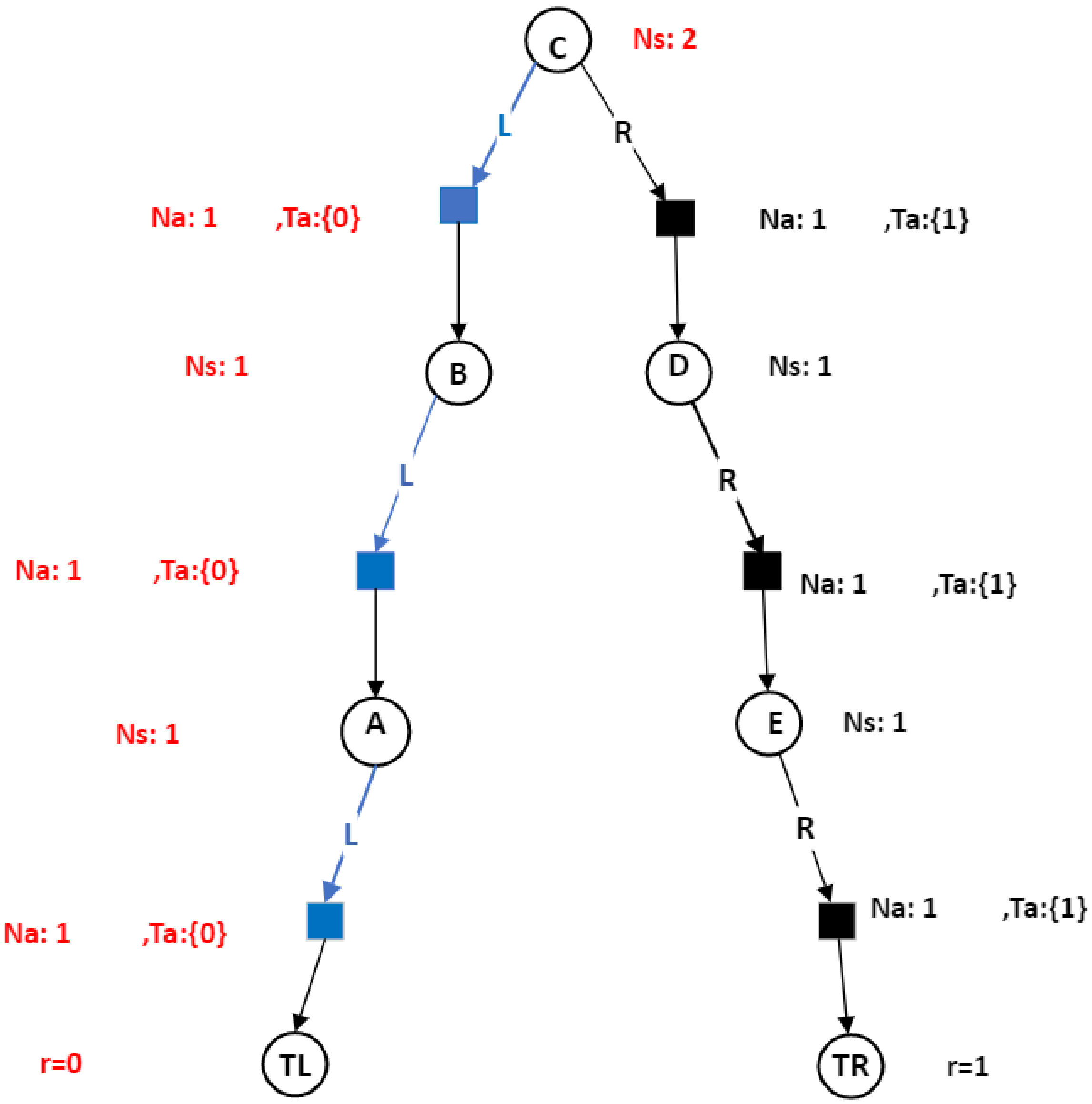


Episode 2:

Forward:

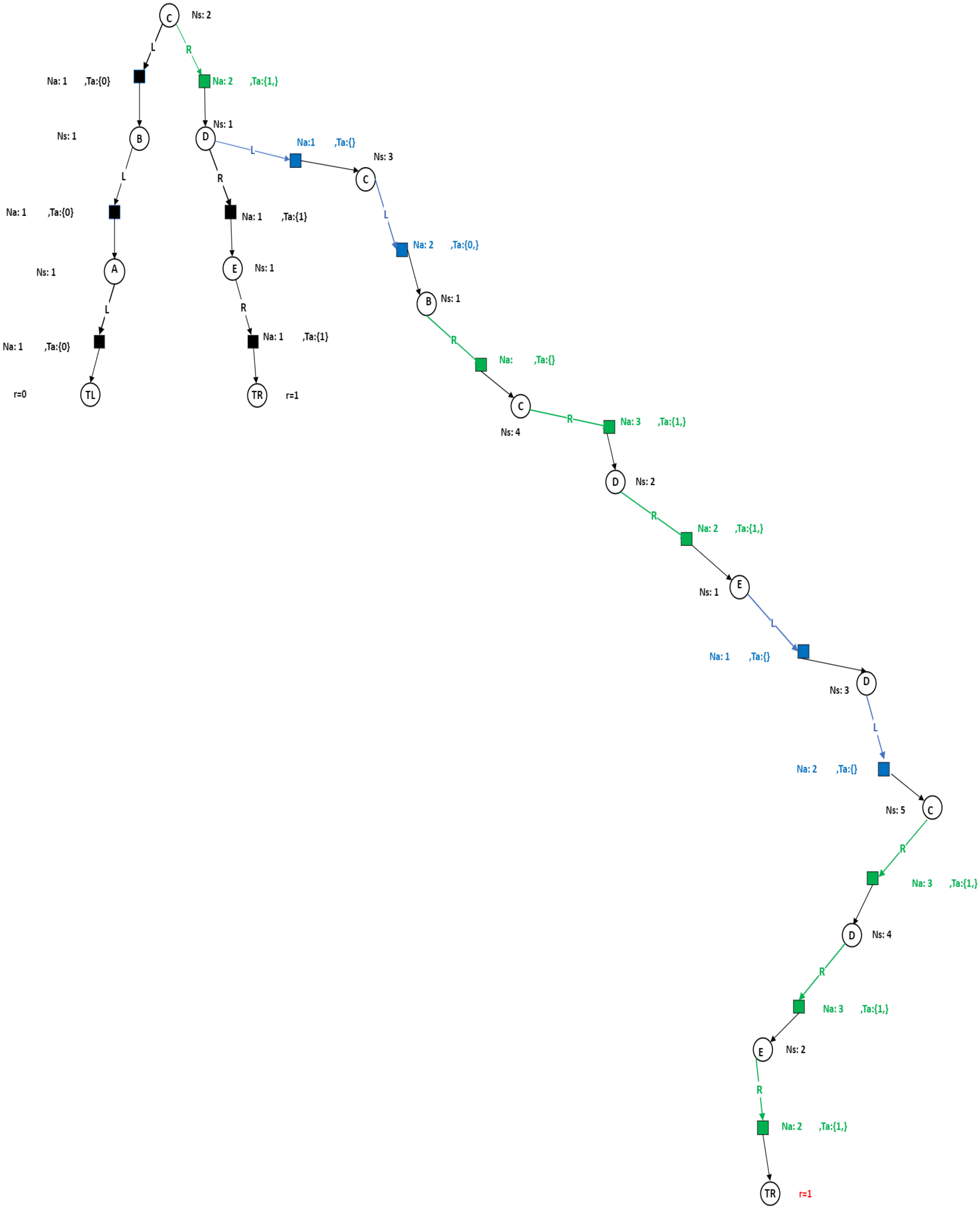

C
Ns: 1
L
R
Na: ,Ta:{}
Na: 1 ,Ta:{1}
B
Ns:
D
Ns: 1
L
R
Na: ,Ta:{}
Na: 1 ,Ta:{1}
A
Ns:
E
Ns: 1
L
R
Na: ,Ta:{}
Na: 1 ,Ta:{1}
r=0
TL
TR
r=1

Backup:

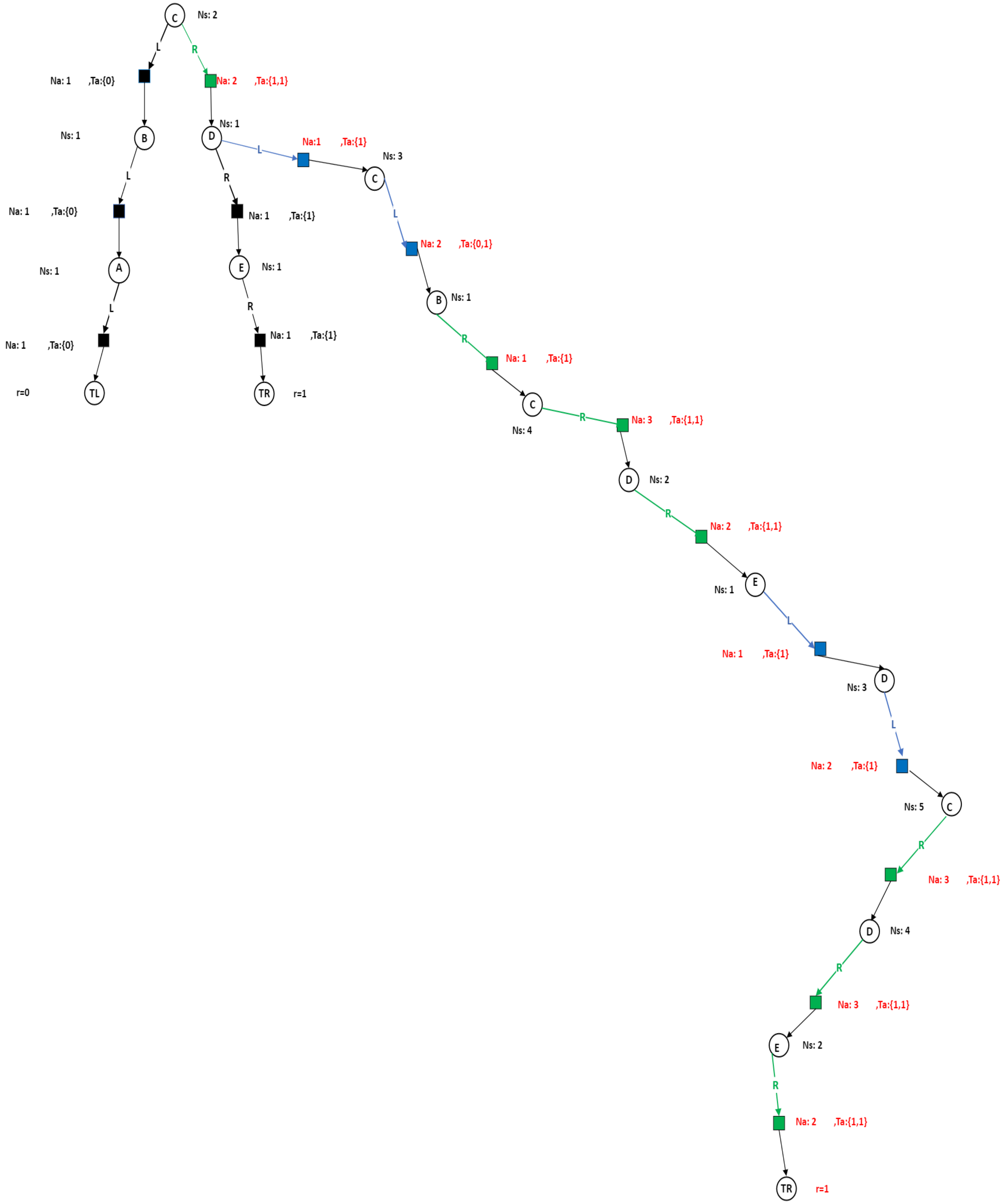


Episode 3:

Forward


C
Ns: 2
L
R
Na: 1
,Ta:{0}
Na: 2
,Ta:{1,}
Ns: 1
B
D
Ns: 1
Na:1
,Ta:{}
L
Ns: 3
C
L
R
Na: 1
,Ta:{0}
Na: 1
,Ta:{1}
L
Na: 2
,Ta:{0,}
Ns: 1
A
E
Ns: 1
B
Ns: 1
L
R
R
Na: 1
,Ta:{0}
Na: 1
,Ta:{1}
Na:
,Ta:{}
r=0
TL
TR
r=1
C
R
Na: 3
,Ta:{1,}
Ns: 4
D
Ns: 2
R
Na: 2
,Ta:{1,}
Ns: 1
E
L
Na: 1
,Ta:{}
D
Ns: 3
L
Na: 2
,Ta:{}
Ns: 5
C
R
Na: 3
,Ta:{1,}
D
Ns: 4
R
Na: 3
,Ta:{1,}
E
Ns: 2
R
Na: 2
,Ta:{1,}
TR
r=1

Backup:


C
Ns: 2
L
R
Na: 1
,Ta:{0}
Na: 2
,Ta:{1,1}
Ns: 1
B
D
Ns: 1
L
Na:1
,Ta:{1}
C
Ns: 3
L
R
Na: 1
,Ta:{0}
Na: 1
,Ta:{1}
L
Na: 2
,Ta:{0,1}
Ns: 1
A
E
Ns: 1
B
Ns: 1
L
R
R
Na: 1
,Ta:{0}
Na: 1
,Ta:{1}
Na: 1
,Ta:{1}
r=0
TL
TR
r=1
C
R
Na: 3
,Ta:{1,1}
Ns: 4
D
Ns: 2
R
Na: 2
,Ta:{1,1}
E
Ns: 1
L
Na: 1
,Ta:{1}
D
Ns: 3
L
Na: 2
,Ta:{1}
Ns: 5
C
R
Na: 3
,Ta:{1,1}
D
Ns: 4
R
Na: 3
,Ta:{1,1}
E
Ns: 2
R
Na: 2
,Ta:{1,1}
TR
r=1

If I wanted to continue to do 8 episodes, it would be after deadline 11:59 PM EST Mar 23, 2022 so I just send these three episodes. The other episode would be like this. Please let me know if I need to complete all 8 episode.