Ch 7. n-step Bootstrapping

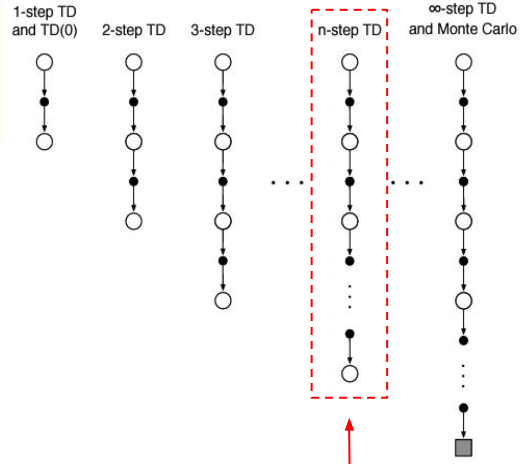
Outline

Motivation

one-step temporal-difference (TD)

(Chapter 6)

- Lower variance
- Some bias
- Online
- Incomplete sequences



Monte Carlo (MC)

(Chapter 5)

- High variance
- Zero bias
- Complete sequences

“ TD(0)와 MC 두 extreme 사이의 중간 지점을 택하면 양쪽의 장점을 활용할 수 있지 않을까? ”

1-step TD vs n-step TD

- 시간 단계의 억압(tyranny)으로부터 자유롭게 해줌

  (많은 경우, 무언가 변한 것을 고려하기 위해 행동을 빠르게 갱신하는 것을 원하지만, Bootstrap은

  중요하고 식별 가능한 상태 변화가 발생한 구간에서 가장 잘 작동함)

- $1 < n < T \quad s.t \quad n \in N$

n-step TD Prediction

one-step

$$G_{t:t+1} \doteq R_{t+1} + \gamma V_t(S_{t+1})$$

two-step

$$G_{t:t+2} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+2})$$

n-step

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}) \longrightarrow$$

state-value learning algorithm

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha\left[G_{t:t+n} - V_{t+n-1}(S_t)\right]$$

Monte Carlo (MC)

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T$$

n-step TD estimating V

n-step TD for estimating $V \approx v_\pi$

Input: a policy π
Algorithm parameters: step size $\alpha \in (0, 1]$, a positive integer n
Initialize $V(s)$ arbitrarily, for all $s \in \mathcal{S}$
All store and access operations (for S_t and R_t) can take their index mod $n+1$

Loop for each episode:
    Initialize and store $S_0 \neq$ terminal
    $T \leftarrow \infty$
    Loop for each step of episode, $t = 0, 1, 2, \ldots$:
    | If $t < T$, then:
    | Take an action according to $\pi(\cdot|S_t)$
    | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
    | If S_{t+1} is terminal, then $T \leftarrow t+1$
    | $\tau \leftarrow t - n + 1$ (τ is the time whose state's estimate is being updated)
    | If $\tau \geq 0$:
    | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
    | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
    | $V(S_\tau) \leftarrow V(S_\tau) + \alpha \left[G - V(S_\tau)\right]$
    Until $\tau = T - 1$

Random Walk

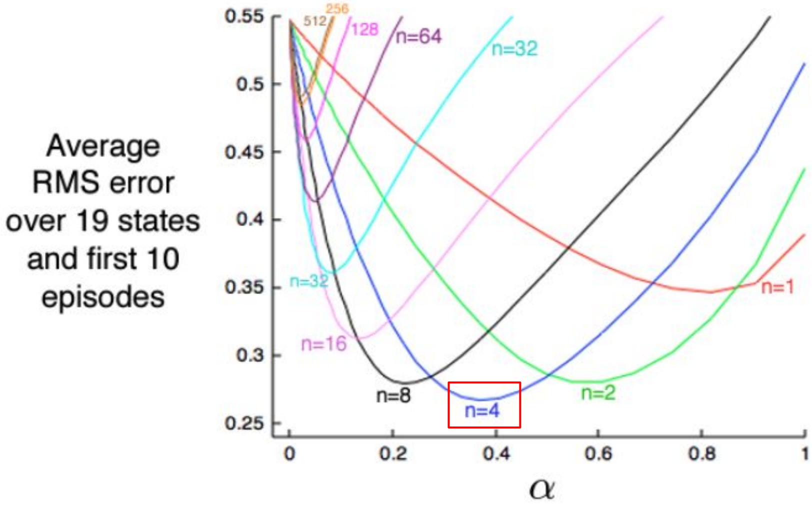
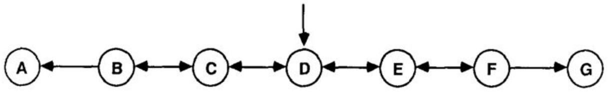
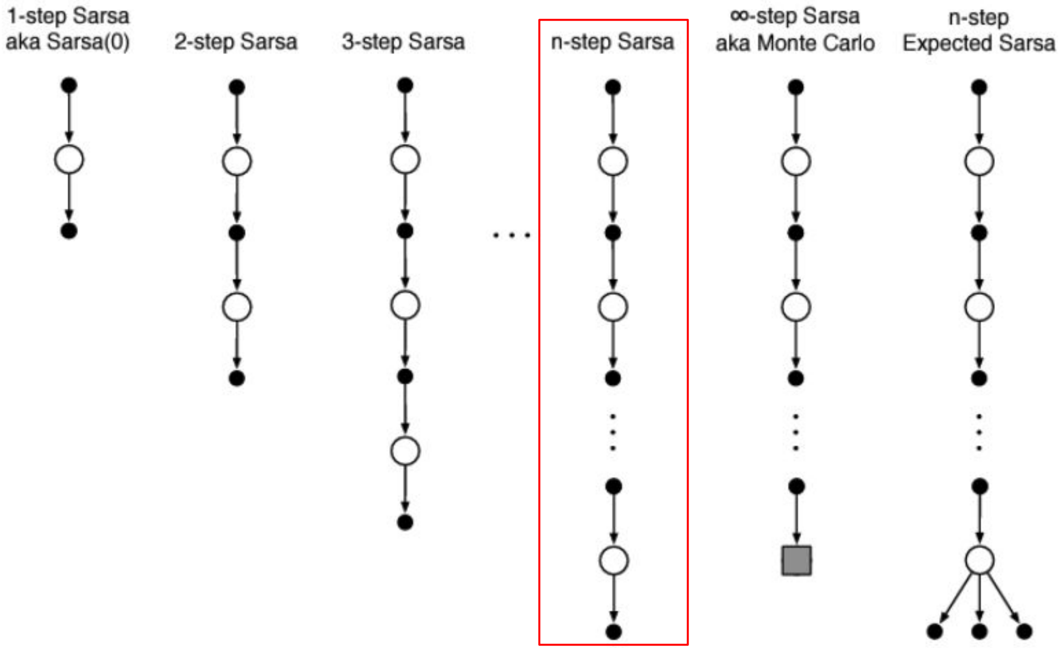


Figure 7.2: Performance of n-step TD methods as a function of α, for various values of n, on a 19-state random walk task (Example 7.1).



“n의 값은 어떻게 선택될 수 있을까 ?”

n-step SARSA



1-step Sarsa
aka Sarsa(0)
2-step Sarsa
3-step Sarsa
n-step Sarsa
∞-step Sarsa
aka Monte Carlo
n-step
Expected Sarsa

n-step SARSA

n-step

$$G_{t:t+n} \doteq R_{t+1}+\gamma R_{t+2}+\cdots+\gamma^{n-1}R_{t+n}+\gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n}), \quad n \geq 1, 0 \leq t < T-n$$

action-value learning algorithm

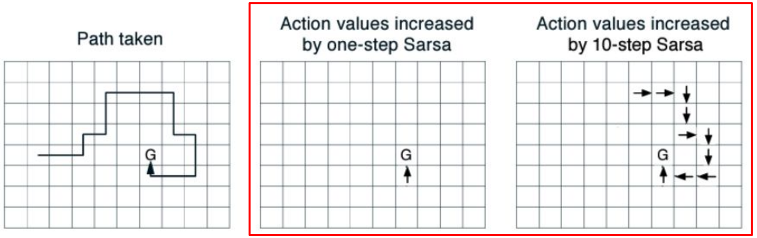
$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha \left[G_{t:t+n} - Q_{t+n-1}(S_t, A_t)\right]$$

n-step SARSA estimating Q

n-step Sarsa for estimating $Q \approx q_*$ or q_π

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ε-greedy with respect to Q, or to a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  Loop for each step of episode, $t = 0, 1, 2, \ldots$:
    If $t < T$, then:
      Take action A_t
      Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
      If S_{t+1} is terminal, then:
        $T \leftarrow t+1$
      else:
        Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
    $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
    If $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
      $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\,[G - Q(S_\tau, A_\tau)]$
      If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
  Until $\tau = T - 1$



n-step Off-policy Learning

Recall ...

- Off-policy에서는 data로부터 behavior policy를 학습하고, 이와 구분되게 target policy가 존재하였음
- 두 policy로 구분되기 때문에 두 분포의 차이를 보정하기 위해 Importance Sampling을 적용

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha \boxed{\rho_{t:t+n-1}} \left[G_{t:t+n} - V_{t+n-1}(S_t)\right], \qquad 0 \leq t < T$$

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha \boxed{\rho_{t+1:t+n}} \left[G_{t:t+n} - Q_{t+n-1}(S_t, A_t)\right]$$

where, $\rho_{t:h} \doteq \prod_{k=t}^{\min(h,T-1)} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$

n-step Off-policy Learning

Off-policy n-step Sarsa for estimating $Q \approx q_*$ or q_π

Input: an arbitrary behavior policy b such that $b(a|s) > 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $Q(s, a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be greedy with respect to Q, or as a fixed given policy
Algorithm parameters: step size $\alpha \in (0, 1]$, a positive integer n
All store and access operations (for S_t, A_t, and R_t) can take their index mod $n + 1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim b(\cdot|S_0)$
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
  | If $t < T$, then:
  | Take action A_t
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then:
  | $T \leftarrow t + 1$
  | else:
  | Select and store an action $A_{t+1} \sim b(\cdot|S_{t+1})$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | $\rho \leftarrow \prod_{i=\tau+1}^{\min(\tau+n-1, T-1)} \frac{\pi(A_i|S_i)}{b(A_i|S_i)}$ $(\rho_{\tau+1:t+n-1})$
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\rho\,[G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is greedy wrt Q
  Until $\tau = T - 1$

Per-decision Methods with Control Variates

$$G_{t:h} = R_{t+1} + \gamma G_{t+1:h}, \qquad t < h < T$$

$$G_{t:h} \doteq \rho_t \left(R_{t+1} + \gamma G_{t+1:h}\right) + \underbrace{\boxed{(1 - \rho_t)V_{h-1}(S_t)}}_{\text{Control variate}}, \qquad t < h < T \qquad\qquad G_{h:h} \doteq V_{h-1}(S_h)$$

$$\begin{aligned} G_{t:h} &\doteq R_{t+1} + \gamma\Big(\rho_{t+1}G_{t+1:h} + \bar{V}_{h-1}(S_{t+1}) - \rho_{t+1}Q_{h-1}(S_{t+1}, A_{t+1})\Big), \\ &= R_{t+1} + \gamma\rho_{t+1}\Big(G_{t+1:h} - Q_{h-1}(S_{t+1}, A_{t+1})\Big) + \gamma\bar{V}_{h-1}(S_{t+1}), \quad t < h \leq T. \end{aligned}$$

n-step Tree-Backup Algorithm

“ Importance Sampling을 사용하지 않고도 off-policy learning이 가능할까 ? ”

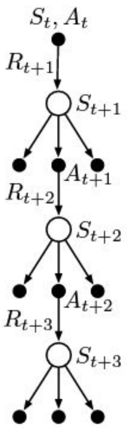
Recall …

- One-step의 경우, chapter 6에서 다뤘듯이 **Q-learning**과 **expected SARSA**를 사용하면 가능했음
- n-step에서는 Tree-Backup Algorithm을 제시함

$$G_{t:t+1} \doteq R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q_t(S_{t+1}, a)$$

$$\begin{aligned} G_{t:t+2} &\doteq R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+1}(S_{t+1}, a) \\ &\quad + \gamma \boxed{\pi(A_{t+1}|S_{t+1})} \Big(R_{t+2} + \gamma \sum_a \pi(a|S_{t+2}) Q_{t+1}(S_{t+2}, a) \Big) \\ &= R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+1}(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:t+2}, \end{aligned}$$

$$G_{t:t+n} \doteq R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+n-1}(S_{t+1}, a) \ + \ \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:t+n}$$



the 3-step tree-backup update

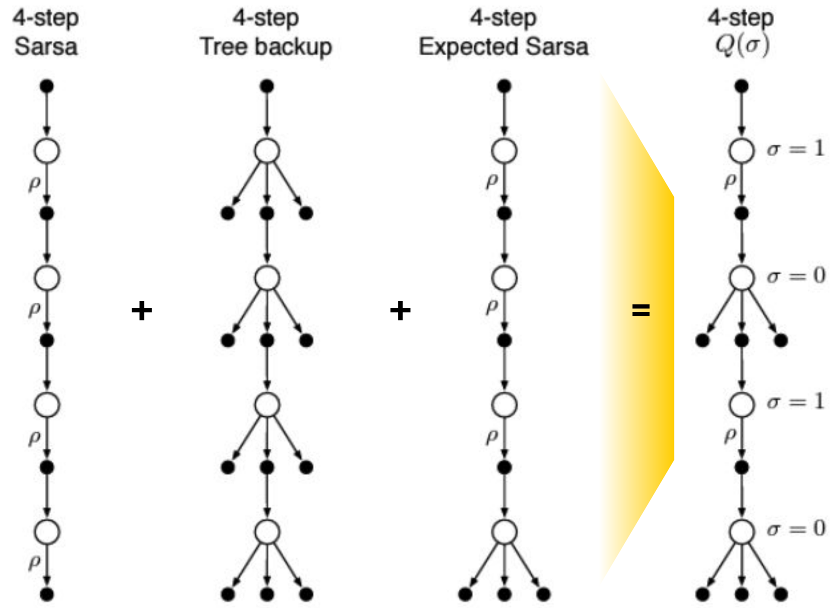
n-step Tree-Backup Algorithm

n-step Tree Backup for estimating $Q \approx q_*$ or q_π

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be greedy with respect to Q, or as a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, a positive integer n
All store and access operations can take their index mod $n+1$

Loop for each episode:
    Initialize and store $S_0 \neq$ terminal
    Choose an action A_0 arbitrarily as a function of S_0; Store A_0
    $T \leftarrow \infty$
    Loop for $t = 0, 1, 2, \ldots$:
    | If $t < T$:
    | Take action A_t; observe and store the next reward and state as R_{t+1}, S_{t+1}
    | If S_{t+1} is terminal:
    | $T \leftarrow t+1$
    | else:
    | Choose an action A_{t+1} arbitrarily as a function of S_{t+1}; Store A_{t+1}
    | $\tau \leftarrow t+1-n$ (τ is the time whose estimate is being updated)
    | If $\tau \geq 0$:
    | If $t+1 \geq T$:
    | $G \leftarrow R_T$
    | else
    | $G \leftarrow R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$
    | Loop for $k = \min(t, T-1)$ down through $\tau+1$:
    | $G \leftarrow R_k + \gamma \sum_{a \neq A_k} \pi(a|S_k) Q(S_k, a) + \gamma \pi(A_k|S_k) G$
    | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha \left[G - Q(S_\tau, A_\tau)\right]$
    | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is greedy wrt Q
    Until $\tau = T-1$

n-step Q(σ)



ϱ : sample transition

if (**σ** = 1)
: full sampling

else
: a pure expectation with no sampling

n-step Q(σ)

Off-policy n-step $Q(\sigma)$ for estimating $Q \approx q_*$ or q_π

Input: an arbitrary behavior policy b such that $b(a|s) > 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ε-greedy with respect to Q, or as a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n
All store and access operations can take their index mod $n+1$

```
Loop for each episode:
    Initialize and store S_0 ≠ terminal
    Choose and store an action A_0 ~ b(·|S_0)
    T ← ∞
    Loop for t = 0, 1, 2, ... :
    |   If t < T:
    |       Take action A_t; observe and store the next reward and state as R_{t+1}, S_{t+1}
    |       If S_{t+1} is terminal:
    |           T ← t + 1
    |       else:
    |           Choose and store an action A_{t+1} ~ b(·|S_{t+1})
    |           Select and store σ_{t+1}
    |           Store π(A_{t+1}|S_{t+1}) / b(A_{t+1}|S_{t+1}) as ρ_{t+1}
    |   τ ← t − n + 1   (τ is the time whose estimate is being updated)
    |   If τ ≥ 0:
    |       G ← 0:
    |       Loop for k = min(t + 1, T) down through τ + 1:
    |           if k = T:
    |               G ← R_T
    |           else:
    |               V̄ ← Σ_a π(a|S_k) Q(S_k, a)
    |               G ← R_k + γ(σ_k ρ_k + (1 − σ_k) π(A_k|S_k))(G − Q(S_k, A_k)) + γV̄
    |       Q(S_τ, A_τ) ← Q(S_τ, A_τ) + α[G − Q(S_τ, A_τ)]
    |       If π is being learned, then ensure that π(·|S_τ) is greedy wrt Q
    Until τ = T − 1
```

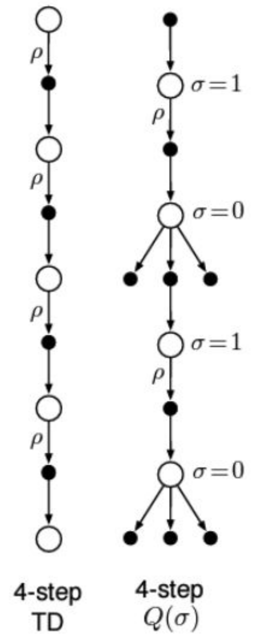
$$\bar{V}_t(s) \doteq \sum_a \pi(a|s) Q_t(s,a), \qquad \text{for all } s \in \mathcal{S}.$$

$$\begin{aligned}
G_{t:h} &= R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{h-1}(S_{t+1}, a) \;+\; \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:h} \\
&= R_{t+1} + \gamma \bar{V}_{h-1}(S_{t+1}) - \gamma \pi(A_{t+1}|S_{t+1}) Q_{h-1}(S_{t+1}, A_{t+1}) + \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:h} \\
&= R_{t+1} + \gamma \pi(A_{t+1}|S_{t+1}) \Big(G_{t+1:h} - Q_{h-1}(S_{t+1}, A_{t+1}) \Big) + \gamma \bar{V}_{h-1}(S_{t+1}),
\end{aligned}$$

$$\begin{aligned}
G_{t:h} \doteq R_{t+1} &+ \gamma \Big(\sigma_{t+1}\rho_{t+1} + (1 - \sigma_{t+1}) \pi(A_{t+1}|S_{t+1}) \Big) \Big(G_{t+1:h} - Q_{h-1}(S_{t+1}, A_{t+1}) \Big) \\
&+ \gamma \bar{V}_{h-1}(S_{t+1}),
\end{aligned}$$

Summary

- ❖ MC와 one-step TD의 중간에 위치하는 n-step methods에 대한 아이디어가 소개됨
- ❖ 비록 n-step 방법이 두 extreme의 장점을 활용한다는 것에 기인했지만, 더 많은 computation을 요구하며, 더 많은 memory를 필요로 한다는 단점도 존재하긴 함
- ❖ Chapter 12. 에서는 multi-step TD method에 대해서 소개를 하는데, eligibility trace라는 방법을 도입해 최소한의 메모리와 계산량을 사용해 이를 구현함
- ❖ 결과적으로, n-step 방식은 개념적으로도 명료하고, 상황에 따라 효율적일 수 있는 방법론이라고 할 수 있음



Thank you for your attention !