2. 마르코프 의사결정 프로세스 (Markov Decision Process)

순천향대학교 컴퓨터공학과

이 상 정

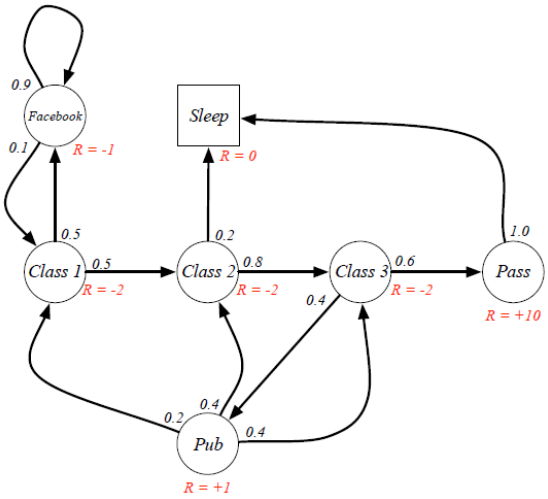
학습 내용

- 강화학습을 위해 수학적인 기초 소개
  - 마르코프 프로세스 (Markov Process)
  - 마르코프 보상 프로세스 (Markov Reward Processes)
  - 마르코프 의사결정 프로세스 (Markov Decision Processes)

마르코프 의사결정 프로세스

- 마르코프 의사결정 프로세스 (Markov Decision Process, MDP)은 강화학습에서 환경을 기술하는 수학 프레임워크
  - 에이전트가 환경에서 발생하는 모든 정보를 볼 수 있다고 가정 (완전한 관찰, *fully observable*)
  - 현재 상태는 진행이 되고 있는 프로세스 중에서 특정 시점이고, 모든 환경을 다 볼 수 있는 완전한 특성을 갖춤

Andrey Markov
(1856-1922)

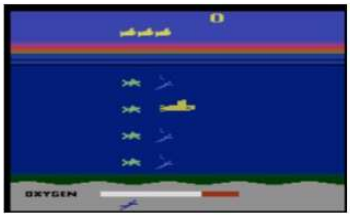


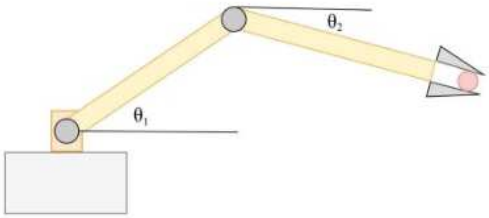
마르코프 프로세스 (Markov Process)

MDP 상태 (State)

- MDP의 상태(status)는 에이전트가 인식하고 관찰하는 상태

- 상태 예
  - 게임에서는 게임 이미지 자체(픽셀)가 상태
  - 로봇 제어에서는 센서가 측정한 조인트 각도, 속도 등





마르코프 프로퍼티 (Markov Property)

- 마르코프 프로퍼티 (Markov Property) 정의
  - 미래는 현재 상태에만 의존하고, 과거와는 독립적

Definition

A state S_t is Markov if and only if

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, \ldots, S_t]$$

  - 현재의 상태는 과거의 모든 관련 정보를 포함
  - 다가오는 미래를 예측하는데 충분한 정보를 포함
  - 강화학습은 현재의 시점에서 미래 가치를 예측하여 의사결정

상태 전이 행렬 (State Transition Matrix)

- 현재의 상태인 s 와 연속된 다음의 상태를 s' 라고 했을때 상태가 s에서 s'로 변경될 **상태 전이 확률 (State Transition Probability)**의 정의

$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

- **상태 전이 행렬 (State Transition Matrix)** P는 모든 상태 s 의 다음의 상태 s' 로 변경될 확률을 정의
  - 각 행의 합은 1

$$\mathcal{P} = \begin{matrix} \\ from \end{matrix} \begin{matrix} \begin{matrix} to \end{matrix} \\ \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix} \end{matrix}$$

마르코프 프로세스 (Markov Process)

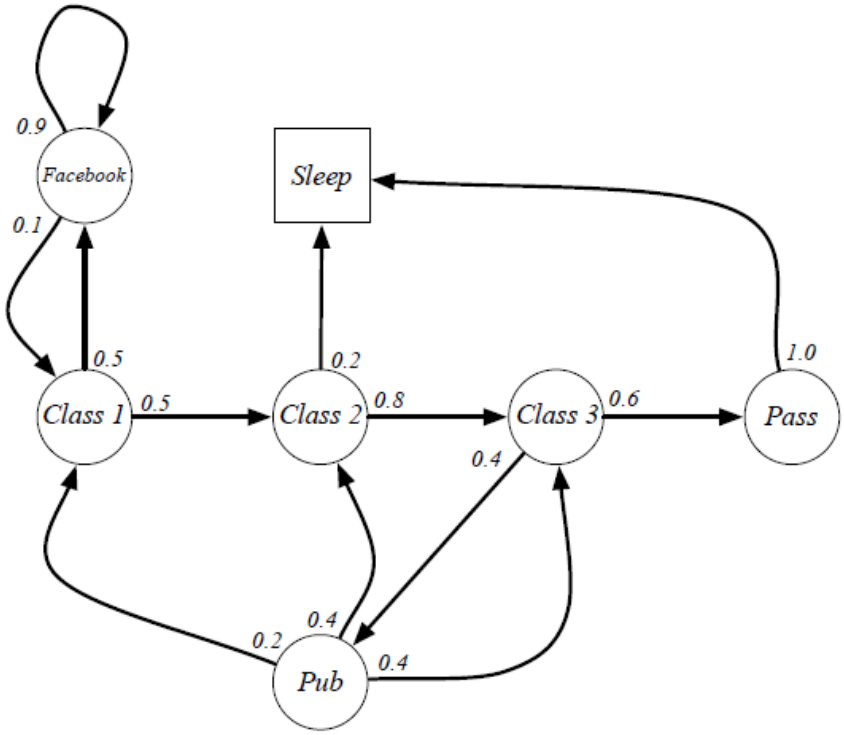
- **마르코프 프로세스 (Markov Process)**는 과거를 기억하지 않는 **마르코프 프로퍼티**를 갖는 **랜덤한 상태들의 시퀀스**와 **상태가 변경(전이)될 확률**로 표현되는 **랜덤 프로세스 (random process)**
  - 마르코프 체인 (Markov Chain)이라고도함

Definition

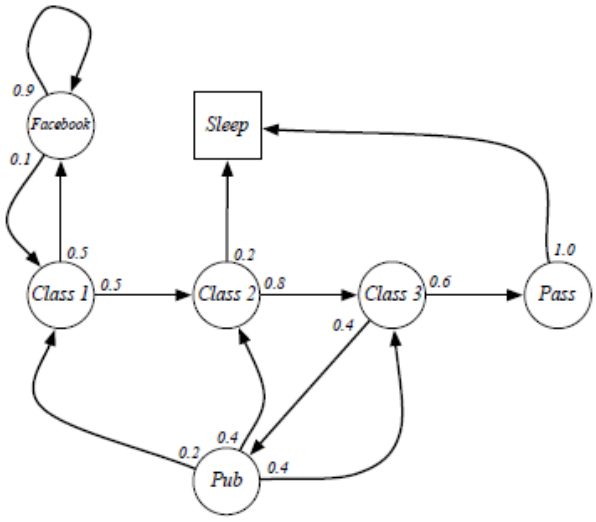
A *Markov Process* (or *Markov Chain*) is a tuple $\langle \mathcal{S}, \mathcal{P} \rangle$

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$

마르코프 프로세스 예: 학생 마르코프 체인



학생 마르코프 체인 예: 에피소드 예

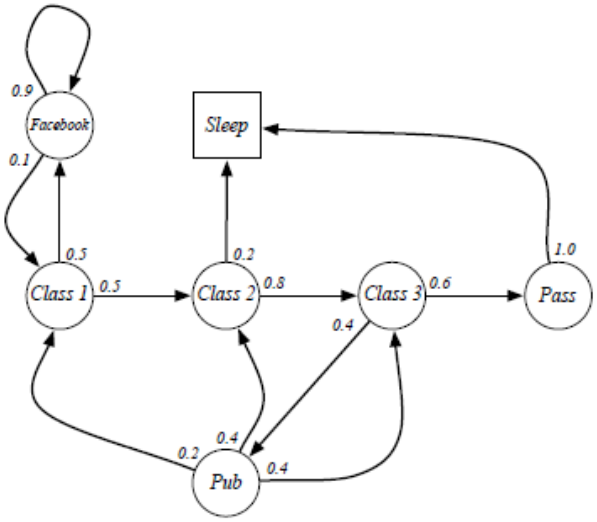


Sample episodes for Student Markov Chain starting from $S_1 = C1$

$$S_1, S_2, ..., S_T$$

- C1 C2 C3 Pass Sleep
- C1 FB FB C1 C2 Sleep
- C1 C2 C3 Pub C2 C3 Pass Sleep
- C1 FB FB C1 C2 C3 Pub C1 FB FB FB C1 C2 C3 Pub C2 Sleep

학생 마르코프 체인 예: 상태 전이 행렬



$$\mathcal{P} = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array} \begin{array}{c} \begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\ \begin{bmatrix} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{bmatrix} \end{array}$$

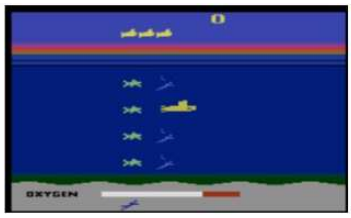
마르코프 보상 프로세스
(Markov Reward Process)

보상 (Reward)

- 에이전트가 취한 행동에 따라 환경이 알려주는 보상 (reward)

$$\mathcal{R}_s^a = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

  - 보상 예
    - 게임에서는 점수
    - 바둑에서는 승패
    - 궤도 제어에서는 의도한 궤도에 얼마나 가깝게 움직였는지 여부



마르코프 보상 프로세스 (Markov Reward Process)

- 마르코프 보상 프로세스 (Markov Reward Process, MRP)는 마르코프 체인에 가치(value) 개념을 추가
  - 현재 상태에서 다음 상태로 변경 시 받게 될 보상 (reward)
  - 0과 1사이의 값을 갖는 할인율 (discount factor)
    - 미래에 받게 될 보상은 할인율을 적용하여 현재 즉시 받게 될 보상과 다른 가치를 적용

Definition

A *Markov Reward Process* is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}\left[R_{t+1} \mid S_t = s\right]$
- γ is a discount factor, $\gamma \in [0, 1]$

할인율 (Discount Factor)

- 현재 얻게 되는 보상이 미래에 얻게 될 보상보다 얼마나 더 중요한지를 나타내는 값으로 0과 1사이의 값

$$\gamma \in [0, 1]$$

  - 스텝 t에서 미래를 포함한 전체 보상

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

1

Worth Now

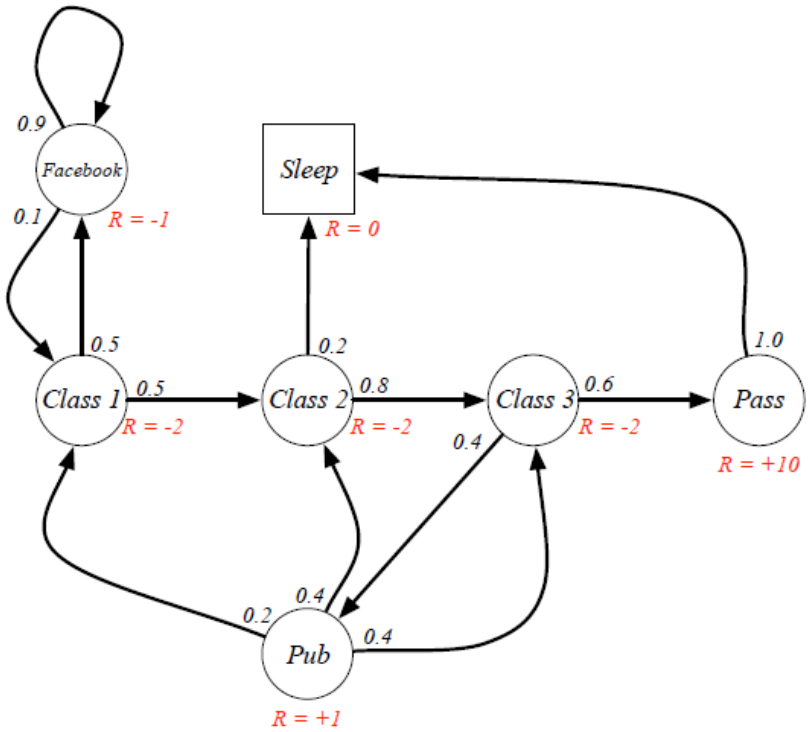
γ

Worth Next Step

γ^2

Worth In Two Steps

학생 마르코프 체인 예: MRP



마르코프 의사결정 프로세스 (Markov Decision Process)

마르코프 결정 프로세스 (Markov Decision Process, MDP)

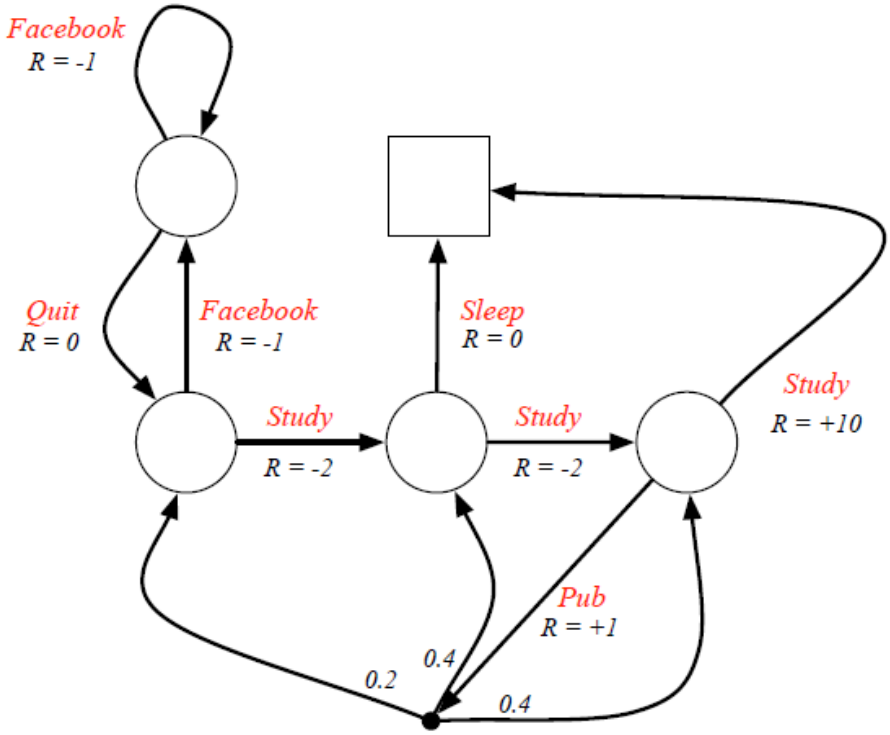
- 마르코프 의사결정 프로세스 (Markov Decision Process, MDP)는 MRP(Markov Reward Process)에 의사결정의 개념을 추가
  - 행동(action)이 추가

Definition

A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^{a}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^{a}_{s} = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- γ is a discount factor $\gamma \in [0, 1]$.

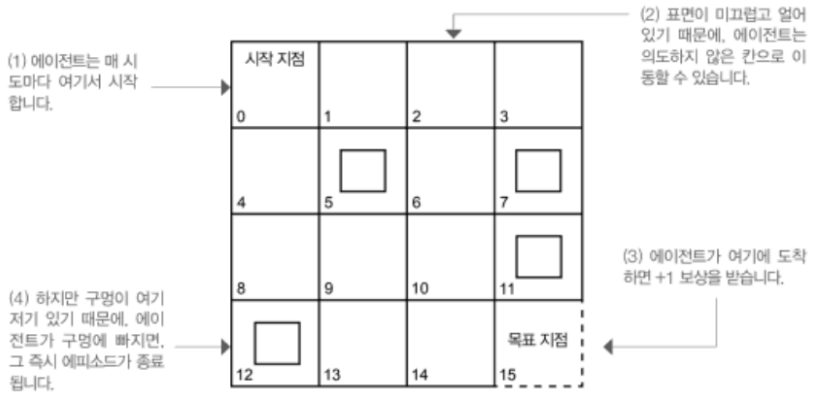
학생 MDP 예



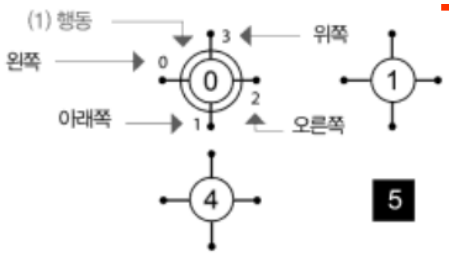
프로즌 레이크 (Frozen Lake) 예

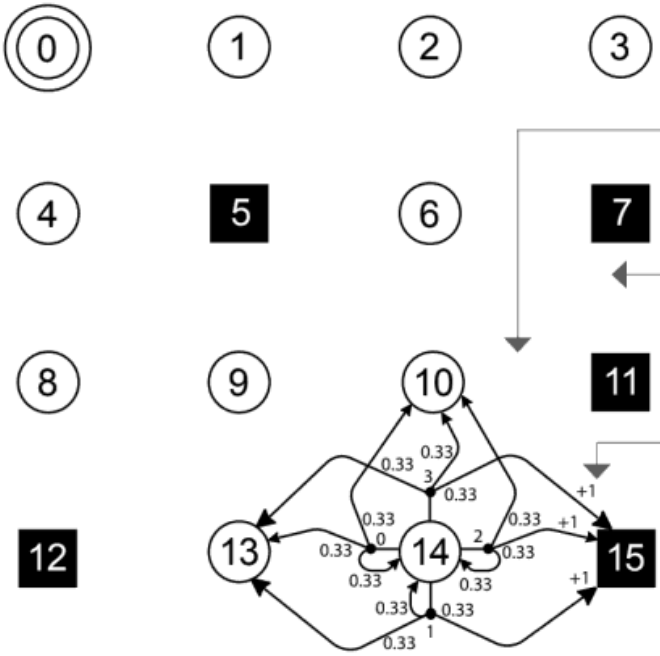
❑ 4 x 4 격자(grid)

- 상,하,좌,우 4개의 행동
  - 행동 시 표면이 미끄러워 1/3 확률로 의도한 행동으로 움직임
    - 나머지는 각각 1/3 확률로 의도한 것과 수직인 방향으로 이동
      - » 아래로 이동하는 행동인 경우 33.3%의 확률로 아래로 이동, 33.3%의 확률로 왼쪽으로 이동, 33.3%의 확률로 오른쪽으로 이동
- 호수(lake)에는 총 4개의 구멍이 있고, 에이전트가 이 구멍으로 빠지면 게임은 종료
- 격자 공간 밖으로 이동 시 현재 칸으로 되튕김
- 상태 15에 도달하면 보상 +1, 나머지는 0



프로즌 레이크 (Frozen Lake) 예: 행동, 전이 함수, 보상





(1) 상태 14에서의 행동 전이 함수와 보상 신호입니다.

(2) 이 환경에서 정의되지 않은 조건에 대한 보상은 모두 0이기 때문에 상태 14에서만 빼고는 모두 생략시켰습니다.

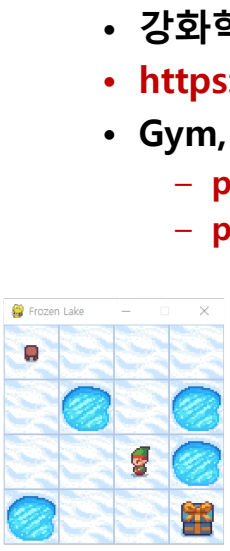
(3) 여기서는 가장 명백한 형태인 완전 전이 $R(s,a,s')$를 사용하고 있습니다.

2. 마르코프 의사결정 프로세스

프로즌 레이크 (Frozen Lake) 예: OpenAI Gym

❑ 실습 환경

- Anaconda3-2023.03-1-Windows-x86_64.exe (2023년 5월 기준)
  - 아나콘다 버전 23.3.1, 파이썬 버전 3.10.9
  - Gym-0.26.2, pygame-2.4.0
- OpenAI Gym
  - 강화학습 알고리즘 수행을 위한 환경 제공
  - https://gym.openai.com/
  - Gym, pygame 파이썬 라이브러리 설치
    - pip install gym
    - pip install pygame




```
import gym

# 프로즌 레이크 환경 생성
env = gym.make('FrozenLake-v1', render_mode='human')
# 상태 출력
env.env.P
```

프로즈 레이크 (Frozen Lake) 실행 예

{상태: {행동: [(전이확률, 다음상태, 보상, 완료여부), (전이확률, 다음상태, 보상, 완료여부)], 행동: [(전이확률, …)]}}

프로즌 레이크(Frozen Lake) 예

```
1  import gym
2
3  # 프로즌 레이크 환경 생성
4  env = gym.make('FrozenLake-v1', render_mode='human')
5  # 상태 출력
6  env.env.P
```

```
{0: {0: [(0.3333333333333333, 0, 0.0, False),
         (0.3333333333333333, 0, 0.0, False),
         (0.3333333333333333, 4, 0.0, False)],
     1: [(0.3333333333333333, 0, 0.0, False),
         (0.3333333333333333, 4, 0.0, False),
         (0.3333333333333333, 1, 0.0, False)],
     2: [(0.3333333333333333, 4, 0.0, False),
         (0.3333333333333333, 1, 0.0, False),
         (0.3333333333333333, 0, 0.0, False)],
     3: [(0.3333333333333333, 1, 0.0, False),
         (0.3333333333333333, 0, 0.0, False),
         (0.3333333333333333, 0, 0.0, False)]},
 1: {0: [(0.3333333333333333, 1, 0.0, False),
         (0.3333333333333333, 0, 0.0, False),
         (0.3333333333333333, 5, 0.0, True)],
     1: [(0.3333333333333333, 0, 0.0, False),
         (0.3333333333333333, 5, 0.0, True),
         (0.3333333333333333, 2, 0.0, False)],
     2: [(0.3333333333333333, 5, 0.0, True),
         (0.3333333333333333, 2, 0.0, False),
         (0.3333333333333333, 1, 0.0, False)],
     3: [(0.3333333333333333, 2, 0.0, False),
         (0.3333333333333333, 1, 0.0, False),
         (0.3333333333333333, 0, 0.0, False)]},
 2: {0: [(0.3333333333333333, 2, 0.0, False),
         (0.3333333333333333, 1, 0.0, False),
```

상태 / 행동 / 전이확률 / 다음상태 / 보상 / 최종상태 여부

```
 12: {0: [(1.0, 12, 0, True)],
      1: [(1.0, 12, 0, True)],
      2: [(1.0, 12, 0, True)],
      3: [(1.0, 12, 0, True)]},
 13: {0: [(0.3333333333333333, 9, 0.0, False),
          (0.3333333333333333, 12, 0.0, True),
          (0.3333333333333333, 13, 0.0, False)],
      1: [(0.3333333333333333, 12, 0.0, True),
          (0.3333333333333333, 13, 0.0, False),
          (0.3333333333333333, 14, 0.0, False)],
      2: [(0.3333333333333333, 13, 0.0, False),
          (0.3333333333333333, 14, 0.0, False),
          (0.3333333333333333, 9, 0.0, False)],
      3: [(0.3333333333333333, 14, 0.0, False),
          (0.3333333333333333, 9, 0.0, False),
          (0.3333333333333333, 12, 0.0, True)]},
 14: {0: [(0.3333333333333333, 10, 0.0, False),
          (0.3333333333333333, 13, 0.0, False),
          (0.3333333333333333, 14, 0.0, False)],
      1: [(0.3333333333333333, 13, 0.0, False),
          (0.3333333333333333, 14, 0.0, False),
          (0.3333333333333333, 15, 1.0, True)],
      2: [(0.3333333333333333, 14, 0.0, False),
          (0.3333333333333333, 15, 1.0, True),
          (0.3333333333333333, 10, 0.0, False)],
      3: [(0.3333333333333333, 15, 1.0, True),
          (0.3333333333333333, 10, 0.0, False),
          (0.3333333333333333, 13, 0.0, False)]},
 15: {0: [(1.0, 15, 0, True)],
      1: [(1.0, 15, 0, True)],
      2: [(1.0, 15, 0, True)],
      3: [(1.0, 15, 0, True)]}}
```

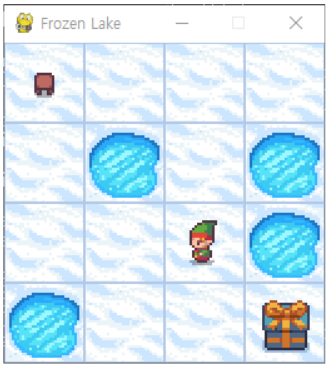

프로즌 레이크 임의의 동작 예

```
goal_steps = 500

# 에피소드 반복
for e in range(10):
    state = env.reset()          # 환경 초기화
    # 최대 에피소드까지 반복
    for t in range(goal_steps):
        # 임의의 액션 지정
        action = env.action_space.sample()
        # 액션을 수행하고 상태, 보상, 종료 여부, 부가정보 리턴
        state, reward, done, truncated, info = env.step(action)
        print(state, ',', reward, ',', done, ',', truncated, ',', info)
        # 종료된 경우 타임 스텝 루프 빠져나와 반복
        if done:
            print("Episode finished after {} timesteps".format(t+1))
            break
        # 격자 디스플레이
        env.render()
```

프로즌 레이크 임의의 동작 실행 예

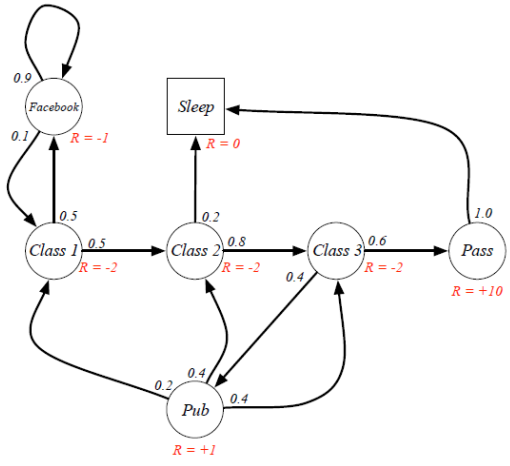
```python
goal_steps =50

# 에피소드 반복
for e in range(10):
    state = env.reset()            # 환경 초기화
    # 최대 타임 스텝까지 반복
    for t in range(goal_steps):
        # 임의의 액션 지정
        action = env.action_space.sample()
        # 액션을 수행하고 상태, 보상, 종료 여부, 부가정보 리턴
        state, reward, done, truncated, info = env.step(action)
        print(state, ',', reward, ',', done, ',', truncated, ',', info)
        # 종료 시 새로운 에피소드
        if done:
            print("Episode finished after {} timesteps".format(t+1))
            break
        # 프로즌레이크 디스플레이
        env.render()
```

```
4 , 0.0 , False , False , {'prob': 0.3333333333333333}
0 , 0.0 , False , False , {'prob': 0.3333333333333333}
0 , 0.0 , False , False , {'prob': 0.3333333333333333}
0 , 0.0 , False , False , {'prob': 0.3333333333333333}
4 , 0.0 , False , False , {'prob': 0.3333333333333333}
4 , 0.0 , False , False , {'prob': 0.3333333333333333}
5 , 0.0 , True , False , {'prob': 0.3333333333333333}
Episode finished after 7 timesteps
0 , 0.0 , False , False , {'prob': 0.3333333333333333}
4 , 0.0 , False , False , {'prob': 0.3333333333333333}
5 , 0.0 , True , False , {'prob': 0.3333333333333333}
Episode finished after 3 timesteps
1 , 0.0 , False , False , {'prob': 0.3333333333333333}
2 , 0.0 , False , False , {'prob': 0.3333333333333333}
3 , 0.0 , False , False , {'prob': 0.3333333333333333}
2 , 0.0 , False , False , {'prob': 0.3333333333333333}
1 , 0.0 , False , False , {'prob': 0.3333333333333333}
0 , 0.0 , False , False , {'prob': 0.3333333333333333}
4 , 0.0 , False , False , {'prob': 0.3333333333333333}
5 , 0.0 , True , False , {'prob': 0.3333333333333333}
Episode finished after 8 timesteps
1 , 0.0 , False , False , {'prob': 0.3333333333333333}
0 , 0.0 , False , False , {'prob': 0.3333333333333333}
1 , 0.0 , False , False , {'prob': 0.3333333333333333}
1 , 0.0 , False , False , {'prob': 0.3333333333333333}
```



과제 2-1: MDP 상태

- 앞의 프로즌 레이크 예에 대해 실행
- 앞의 학생 MDP 예에 대해 상태 정의
  - 앞의 Open AI Gym의 프로즌 레이크 예 참조하여 딕셔너리로 상태 정의
  - 정의된 다음 행동 확률로 행동 선택 후 실행



참고 자료

- David Silver - UCL Course on RL, 2015
  - https://www.davidsilver.uk/teaching/
  - Lecture 2: Markov Decision Processes
- Miguel Morales, Grokking Deep Reinforcement Learning
  - https://livebook.manning.com/book/grokking-deep-reinforcement-learning
  - 그로킹 심층 강화학습, 강찬석 옮김, 한빛미디어
  - 2장. 강화학습의 수학적 기초