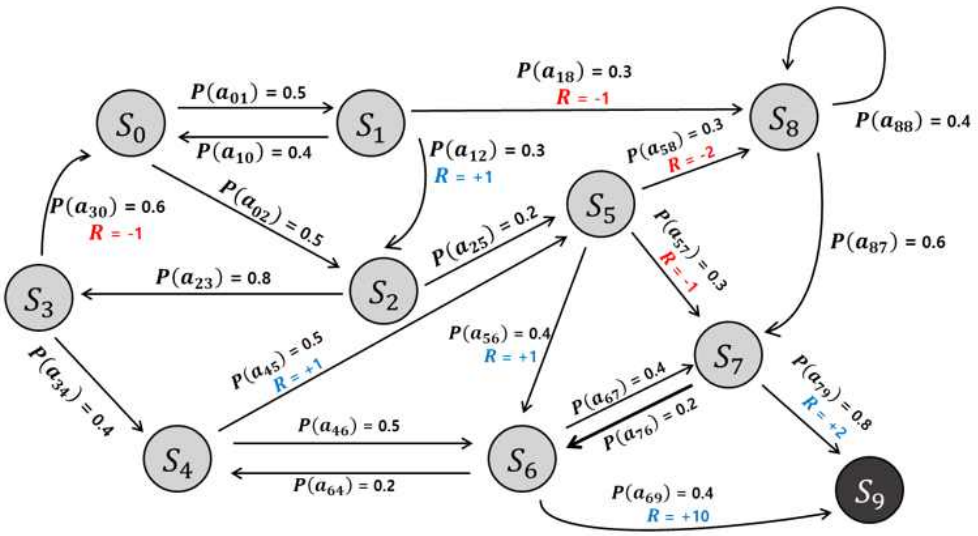
REPORT

“ 다이나믹 프로그래밍 - 정책 이터레이션 ”

과 목 명	지능 시스템
담당교수	조영완 교수님
학 과	컴퓨터공학과
학 번	2016305078
이 름	최영환
제 출 일	2021.03.24

● 다음의 State diagram으로 나타난 MDP에 대해 정책 이터레이션을 이용하여 각 상태의 가치함수 및 최적 정책을 구하는 프로그램을 작성하고 결과를 제시하시오.



● 아래의 알고리즘을 참고하여 프로그램을 작성하였음.

Policy iteration (using iterative policy evaluation)

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
      $\Delta \leftarrow 0$
      For each $s \in \mathcal{S}$:
         $v \leftarrow V(s)$
         $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\left[r + \gamma V(s')\right]$
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      old-action $\leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
      If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

1. 초기화 단계

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

가치함수 V(s) : 실수 집합 R에 포함
정책 π(s) : 각각의 행동 집합 A(s)에 포함
상태집합 S 의 모든 상태 s 에 대하여 임의로 V(s) 와 π(s)를 초기화함

```
# 상태 집합 S
S = [0, 1, 2, 3, 4, 5, 6, 7, 8]
# 각 상태의 첫번째 행동
a0 = [0, 1, 0, 0, 0, 0, 0, 0, 0]
a1 = [1, 0, 0, 0, 0, 0, 0, 0, 0]
a2 = [0, 0, 0, 1, 0, 0, 0, 0, 0]
a3 = [1, 0, 0, 0, 0, 0, 0, 0, 0]
a4 = [0, 0, 0, 0, 0, 1, 0, 0, 0]
a5 = [0, 0, 0, 0, 0, 0, 1, 0, 0]
a6 = [0, 0, 0, 0, 1, 0, 0, 0, 0]
a7 = [0, 0, 0, 0, 0, 0, 1, 0, 0]
a8 = [0, 0, 0, 0, 0, 0, 0, 1, 0]
a9 = [0, 0, 0, 0, 0, 0, 0, 0, 0]
a = [a0, a1, a2, a3, a4, a5, a6, a7, a8]
# 가치함수 집합 V
V = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
# 각 상태의 상태변환확률
p0 = [0, 0.5, 0.5, 0, 0, 0, 0, 0, 0, 0]
p1 = [0.4, 0, 0.3, 0, 0, 0, 0, 0, 0.3, 0]
p2 = [0, 0, 0, 0.8, 0, 0.2, 0, 0, 0, 0]
p3 = [0.6, 0, 0, 0, 0.4, 0, 0, 0, 0, 0]
p4 = [0, 0, 0, 0, 0, 0.5, 0.5, 0, 0, 0]
p5 = [0, 0, 0, 0, 0, 0, 0.4, 0.3, 0.3, 0]
p6 = [0, 0, 0, 0, 0.2, 0, 0, 0.4, 0, 0.4]
p7 = [0, 0, 0, 0, 0, 0, 0.2, 0, 0, 0.8]
p8 = [0, 0, 0, 0, 0, 0, 0, 0.6, 0.4, 0]
p9 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
p = [p0, p1, p2, p3, p4, p5, p6, p7, p8]
# 보상 집합
r0 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
r1 = [0, 0, 1, 0, 0, 0, 0, 0, -1, 0]
r2 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
r3 = [-1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
r4 = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
r5 = [0, 0, 0, 0, 0, 0, 0, 1, -1, -2, 0]
r6 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 10]
r7 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 2]
r8 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
r9 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
r = [r0, r1, r2, r3, r4, r5, r6, r7, r8]
```

▶ 상태 집합 S는 MDP에 제시된 숫자를 사용하여 초기화하였음.

▶ 가치함수 집합 V는 0으로 초기화하였음.

▶ 정책 집합 a는 각 상태에서 설정할 수 있는 첫 번째 행동을 0과 1을 통해 표현하였음.
예를 들어 S_0 에서 S_1 로 가는 경우, a0에 저장된 리스트와 같이 해당 행동의 번호에 1을 넣어서 초기화하였음.

▶ 각 상태의 상태변환확률집합인 p는 MDP에 제시된 $P(a_{ss'})$ 의 값을 통해 초기화하였음.

▶ 각 상태의 보상 집합 r 역시 MDP에 제시된 R 값을 통해 초기화하였음.

2. 정책 평가

2. Policy Evaluation
Repeat
$\Delta \leftarrow 0$
For each $s \in \mathcal{S}$:
$v \leftarrow V(s)$
$V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))[r + \gamma V(s')]$
$\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

▶ Δ(델타, delta) : 값 비교를 위한 임의의 변수, 현재 상태 s와 다음 상태 s'의 가치함수 간의 차이의 최댓값이 저장되는 변수.

▶ θ(세타, theta) : 임의의 양의 실수.

▶ Δ(델타)가 θ(세타)의 값보다 작아질 때까지 평가를 반복한다.

```python
# 정책 평가 (Policy Evaluation)
while True:
    evaluation_step += 1
    print(f'\n정책 평가 스텝 수 : {evaluation_step} 스텝 ')
    delta = 0
    for s in S:
        v = V[s]
        value = 0
        i = 0
        for percentage, reward in zip(p[s], r[s]):
            value += percentage * (reward + 0.9 * V[i])
            i += 1
        V[s] = value
        delta = max(delta, abs(v - V[s]))
        print(f'V[s{s}] = {V[s]}')

    print(f'Delta = max(v- V[s]) = {delta}')
    if delta < theta:
        break
```

▶ i는 다음 상태 s'을 의미함.

▶ p[s] 와 r[s] 는 현재 상태 s 에서 다음 상태 s'으로 이동할 때의 확률과 보상.

▶ 감가율은 0.9로, θ(세타)의 값은 0.001로 설정하였음.

3. 정책 발전

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      old-action $\leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$
      If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

▶ **policy-stable : 정책 발전의 반복 여부를 결정함. 이전의 정책과 새로운 정책이 같으면 정책 발전 단계를 종료하고, 그렇지 않으면 정책 평가 단계로 되돌아간다.**

▶ **old-action : 정책 발전 이전의 상태 s에서 다음 상태 s'으로 가는 행동이 저장된 정책 π(s)**

▶ **큐 함수를 통해 상태 s에서 선택 가능한 각각의 행동에 대하여, 큐 함수의 값이 가장 큰 행동을 정책 π(s)에 저장함.**

```
# 정책 발전 (Policy Improvement)
print(f'\n정책 발전 ')
policy_stable = True
for s in S:
    print(f'a[s{s}] = {a[s]}', end=' ---> ')
    old_action = a[s].index(1)
    q_list = []
    i = 0
    for percentage, reward in zip(p[s], r[s]):
        q_value = percentage * (reward + 0.9 * V[i])
        q_list.append(q_value)
        i += 1

    index = q_list.index(max(q_list))
    a[s][old_action] = 0
    a[s][index] = 1

    print(f'a[s{s}] = {a[s]}')
    if old_action != index:
        policy_stable = False

if policy_stable == True:
    break
```

▶ **old_action에는 상태 s의 이전 정책 a[s]가 어떤 상태로 갈지의 행동이 저장됨.**

▶ **i는 다음 상태 s'를, p[s]와 r[s]는 현재 상태에서 다음 상태로 가는 확률과 보상이 저장됨.**

▶ **index는 큐 함수의 값이 큰 행동을 의미하며, 이전의 정책을 0으로, 새로운 정책을 1로 함.**

▶ **이전의 정책과 새로운 정책이 다르면 정책 평가 단계로 돌아가며,**
이전의 정책과 새로운 정책이 같으면 정책 이터레이션을 종료한다.

< 전체 소스 코드 >

```
# 상태 집합 S
S = [0, 1, 2, 3, 4, 5, 6, 7, 8]

# 각 상태의 첫번째 행동
a0 = [0, 1, 0, 0, 0, 0, 0, 0, 0]
a1 = [1, 0, 0, 0, 0, 0, 0, 0, 0]
a2 = [0, 0, 0, 1, 0, 0, 0, 0, 0]
a3 = [1, 0, 0, 0, 0, 0, 0, 0, 0]
a4 = [0, 0, 0, 0, 0, 1, 0, 0, 0]
a5 = [0, 0, 0, 0, 0, 0, 1, 0, 0]
a6 = [0, 0, 0, 0, 1, 0, 0, 0, 0]
a7 = [0, 0, 0, 0, 0, 0, 1, 0, 0]
a8 = [0, 0, 0, 0, 0, 0, 0, 1, 0]
a9 = [0, 0, 0, 0, 0, 0, 0, 0, 0]
a = [a0, a1, a2, a3, a4, a5, a6, a7, a8]

# 가치함수 집합 V
V = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

# 각 상태의 상태변환확률
p0 = [0, 0.5, 0.5, 0, 0, 0, 0, 0, 0, 0]
p1 = [0.4, 0, 0.3, 0, 0, 0, 0, 0, 0.3, 0]
p2 = [0, 0, 0, 0.8, 0, 0.2, 0, 0, 0, 0]
p3 = [0.6, 0, 0, 0, 0.4, 0, 0, 0, 0, 0]
p4 = [0, 0, 0, 0, 0, 0.5, 0.5, 0, 0, 0]
p5 = [0, 0, 0, 0, 0, 0, 0.4, 0.3, 0.3, 0]
p6 = [0, 0, 0, 0, 0.2, 0, 0, 0.4, 0, 0.4]
p7 = [0, 0, 0, 0, 0, 0, 0.2, 0, 0, 0.8]
p8 = [0, 0, 0, 0, 0, 0, 0, 0.6, 0.4, 0]
p9 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
p = [p0, p1, p2, p3, p4, p5, p6, p7, p8]

# 보상 집합
r0 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
r1 = [0, 0, 1, 0, 0, 0, 0, 0, -1, 0]
r2 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
r3 = [-1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
r4 = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
r5 = [0, 0, 0, 0, 0, 0, 1, -1, -2, 0]
r6 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 10]
r7 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 2]
r8 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
r9 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
r = [r0, r1, r2, r3, r4, r5, r6, r7, r8]
```

```
# 정책 이터레이션 (Policy Iteration)
policy_iteration_step = 0
evaluation_step = 0
theta = 0.001
while True:
    policy_iteration_step += 1
    print(f'정책 이터레이션 스텝: {policy_iteration_step}')

    # 정책 평가 (Policy Evaluation)
    while True:
        evaluation_step += 1
        print(f'\n정책 평가 스텝 수 : {evaluation_step} 스텝 ')
        delta = 0
        for s in S:
            v = V[s]
            value = 0
            i = 0
            for percentage, reward in zip(p[s], r[s]):
                value += percentage * (reward + 0.9 * V[i])
                i += 1
            V[s] = value
            delta = max(delta, abs(v - V[s]))
            print(f'V[s{s}] = {V[s]}')

        print(f'Delta = max(v- V[s]) = {delta}')
        if delta < theta:
            break
```

```
    # 정책 발전 (Policy Improvement)
    print(f'\n정책 발전 ')
    policy_stable = True
    for s in S:
        print(f'a[s{s}] = {a[s]}', end=' ---> ')
        old_action = a[s].index(1)
        q_list = []
        i = 0
        for percentage, reward in zip(p[s], r[s]):
            q_value = percentage * (reward + 0.9 * V[i])
            q_list.append(q_value)
            i += 1

        index = q_list.index(max(q_list))
        a[s][old_action] = 0
        a[s][index] = 1

        print(f'a[s{s}] = {a[s]}')
        if old_action != index:
            policy_stable = False

    if policy_stable == True:
        break
```

```python
print(f'==========================================================================================================')
print(f'정책 이터레이션 스텝 수 : {policy_iteration_step} 스텝')

print()
for s in S:
    print(f'p[s{s}] = {p[s]}')

print()
for s in S:
    print(f'Policy[s{s}] => {a[s]}')

print()
for s in S:
    print(f'Policy[s{s}] => s{a[s].index(max(a[s]))}')

print()
for i, s in enumerate(S):
    print(f'V[{i}] = {V[s]}')
```

< 실행 결과 >

```
정책 이터레이션 스텝: 1

정책 평가 스텝 수 : 1 스텝
V[s0] = 0.0
V[s1] = 0.0
V[s2] = 0.0
V[s3] = -0.6
V[s4] = 0.5
V[s5] = -0.49999999999999994
V[s6] = 4.09
V[s7] = 2.3362000000000003
V[s8] = 1.2615480000000001
Delta = max(v- V[s]) = 4.09
정책 평가 스텝 수 : 2 스텝
V[s0] = 0.0
V[s1] = 0.34061796000000005
V[s2] = -0.522
V[s3] = -0.41999999999999993
V[s4] = 2.1155
V[s5] = 1.9437919600000002
V[s6] = 5.221822
V[s7] = 2.5399279600000004
V[s8] = 1.8257183784000004
Delta = max(v- V[s]) = 2.44379196
정책 평가 스텝 수 : 3 스텝
V[s0] = -0.08162191799999999
V[s1] = 0.3226200716880001
V[s2] = 0.04748255280000002
V[s3] = 0.11750416427999999
V[s4] = 3.7245262820000002
V[s5] = 2.558580431368001
V[s6] = 5.58478879636
V[s7] = 2.6052619833448
V[s8] = 2.064100887230192
Delta = max(v- V[s]) = 1.6090262820000003
정책 평가 스텝 수 : 4 스텝
V[s0] = 0.16654618101960006
V[s1] = 0.6300839379752078
V[s2] = 0.5451474759278403
V[s3] = 0.8307643992705843
V[s4] = 4.164516152477601
V[s5] = 2.771251725744848
V[s6] = 5.687507221450096
V[s7] = 2.6237512998610173
V[s8] = 2.1599017333278185
Delta = max(v- V[s]) = 0.7132602349905843
정책 평가 스텝 수 : 5 스텝
V[s0] = 0.5288541362563717
V[s1] = 0.9207507755513218
V[s2] = 1.0969756781088935
V[s3] = 1.1848070484703772
V[s4] = 4.306441526237725
V[s5] = 2.8390889186830206
V[s6] = 5.719709942672757
V[s7] = 2.629547789681096
V[s8] = 2.1975204304258065
Delta = max(v- V[s]) = 0.5518282021810532
정책 평가 스텝 수 : 6 스텝
V[s0] = 0.9079769041470969
V[s1] = 1.2163856347973239
V[s2] = 1.3640970802616152
V[s3] = 1.4406264776850135
V[s4] = 4.3514594876101
V[s5] = 2.8624039987910566
V[s6] = 5.7298999120550125
V[s7] = 2.631381984169902
V[s8] = 2.2120536264050377
Delta = max(v- V[s]) = 0.3791227678907252
정책 평가 스텝 수 : 7 스텝
V[s0] = 1.1612172217765226
V[s1] = 1.3835988906395442
V[s2] = 1.5524837837156
V[s3] = 1.5935827152989581
V[s4] = 4.366536759880731
V[s5] = 2.8704915831950384
V[s6] = 5.733274131079696
V[s7] = 2.6319893435943458
V[s8] = 2.2176135510467603
Delta = max(v- V[s]) = 0.2532403176294258
정책 평가 스텝 수 : 8 스텝
V[s0] = 1.321237203459815
V[s1] = 1.4935716736313707
V[s2] = 1.6640680399903567
V[s3] = 1.6854213234253632
V[s4] = 4.371694571423631
V[s5] = 2.8733714687417895
V[s6] = 5.734421186550218
V[s7] = 2.632195813579039
V[s8] = 2.2197266177095147
Delta = max(v- V[s]) = 0.16081998168329234
정책 평가 스텝 수 : 9 스텝
V[s0] = 1.4209378711297773
V[s1] = 1.560162191185685
V[s2] = 1.730710217239784
V[s3] = 1.7411164961225871
V[s4] = 4.373506694881403
V[s5] = 2.874410683605988
V[s6] = 5.7348216979671865
V[s7] = 2.6322679056340794
V[s8] = 2.2205262514178283
Delta = max(v- V[s]) = 0.09970066766996233
```

< 실행 결과 >

```
정책 평가 스텝 수 : 10 스텝
V[s0] = 1.480892583791461
V[s1] = 1.599955176702481
V[s2] = 1.7709978002573405
V[s3] = 1.774144405404694
V[s4] = 4.3741545717078925
V[s5] = 2.874790233672174
V[s6] = 5.7349642689356894
V[s7] = 2.632293568408424
V[s8] = 2.2208279774509676
Delta = max(v- V[s]) = 0.05995471266168373
정책 평가 스텝 수 : 11 스텝
V[s0] = 1.5169288396319196
V[s1] = 1.6238873422487343
V[s2] = 1.794846213952371
V[s3] = 1.7938372192160779
V[s4] = 4.374389526173538
V[s5] = 2.874929954198884
V[s6] = 5.73501579933827
V[s7] = 2.632302843880889
V[s8] = 2.2209416075780286
Delta = max(v- V[s]) = 0.03603625584045855
정책 평가 스텝 수 : 12 스텝
V[s0] = 1.5384301002904976
V[s1] = 1.638097547917787
V[s2] = 1.8090501895913755
V[s3] = 1.8055324835793427
V[s4] = 4.37447558909172
V[s5] = 2.8749816896556855
V[s6] = 5.735034629833629
V[s7] = 2.6323062333700538
V[s8] = 2.2209843447479196
Delta = max(v- V[s]) = 0.02150126065857804
정책 평가 스텝 수 : 13 스텝
V[s0] = 1.5512164818791232
V[s1] = 1.6465472577480942
V[s2] = 1.8174800923151504
V[s3] = 1.8124681122877457
V[s4] = 4.374507343770192
V[s5] = 2.8750009228319597
V[s6] = 5.735041565891854
V[s7] = 2.6323074818605336
V[s8] = 2.2210004043139393
Delta = max(v- V[s]) = 0.01278638158862555
정책 평가 스텝 수 : 14 스텝
V[s0] = 1.5588123075284601
V[s1] = 1.6515621648000998
V[s2] = 1.82247720695693
V[s3] = 1.8165812898226377
V[s4] = 4.374519119925717
V[s5] = 2.8750080929881756
V[s6] = 5.735044135056421
V[s7] = 2.6323079443101562
V[s8] = 2.2210064354805024
Delta = max(v- V[s]) = 0.007595825649336918
정책 평가 스텝 수 : 15 스텝
V[s0] = 1.5633177172906634
V[s1] = 1.6545349616827454
V[s2] = 1.825439985410171
V[s3] = 1.8190184505102165
V[s4] = 4.374523502620069
V[s5] = 2.87501077116379
V[s6] = 5.735045090423268
V[s7] = 2.6323081162761888
V[s8] = 2.2210086995621228
Delta = max(v- V[s]) = 0.004505409762203261
정책 평가 스텝 수 : 16 스텝
V[s0] = 1.5659887261918124
V[s1] = 1.6562970863715718
V[s2] = 1.827195223176838
V[s3] = 1.8204623730868037
V[s4] = 4.374525137714176
V[s5] = 2.875011772828721
V[s6] = 5.73504544664798
V[s7] = 2.6323081803966364
V[s8] = 2.221009549256548
Delta = max(v- V[s]) = 0.002671008901149019
정책 평가 스텝 수 : 17 스텝
V[s0] = 1.5675715392967844
V[s1] = 1.6573410427038566
V[s2] = 1.8282350277316686
V[s3] = 1.8213176807973672
V[s4] = 4.3745257487645155
V[s5] = 2.8750121477996333
V[s6] = 5.735045579720402
V[s7] = 2.6323082043496724
V[s8] = 2.2210098680811803
Delta = max(v- V[s]) = 0.0015828131049719918
정책 평가 스텝 수 : 18 스텝
V[s0] = 1.5685092316959863
V[s1] = 1.657959445280024
V[s2] = 1.8288509167780385
V[s3] = 1.8218242546710584
V[s4] = 4.374525977384016
V[s5] = 2.8750122882556752
V[s6] = 5.735045629495005
V[s7] = 2.632308213309101
V[s8] = 2.2210099876961396
Delta = max(v- V[s]) = 0.0009376923992019393
```

```
정책 발전
a[s0] = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0] ---> a[s0] = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
a[s1] = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0] ---> a[s1] = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
a[s2] = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0] ---> a[s2] = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
a[s3] = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0] ---> a[s3] = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
a[s4] = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0] ---> a[s4] = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
a[s5] = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0] ---> a[s5] = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
a[s6] = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0] ---> a[s6] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
a[s7] = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0] ---> a[s7] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
a[s8] = [0, 0, 0, 0, 0, 0, 0, 1, 0, 0] ---> a[s8] = [0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
```

▶ **정책 이터레이션의 첫 스텝은 총 18번의 정책 평가 반복 이후에 정책 발전으로 넘어감.**

▶ **이 과정에서 Δ(델타)의 값이 점점 감소하는 것을 확인함.**

▶ **정책발전에서는 각 상태 s에서의 정책도 변화함을 확인함.**

< 실행 결과 >

```
정책 이터레이션 스텝: 2

정책 평가 스텝 수 : 19 스텝
V[s0] = 1.5690646629261282
V[s1] = 1.6583257228614343
V[s2] = 1.8292156752491837
V[s3] = 1.8221242698383553
V[s4] = 4.374526062987806
V[s5] = 2.8750123408896173
V[s6] = 5.735045648129082
V[s7] = 2.632308216663235
V[s8] = 2.221010032568757
Delta = max(v- V[s]) = 0.0005554312301419007

정책 발전
a[s0] = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0] ---> a[s0] = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
a[s1] = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0] ---> a[s1] = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
a[s2] = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0] ---> a[s2] = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
a[s3] = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0] ---> a[s3] = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
a[s4] = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0] ---> a[s4] = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
a[s5] = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0] ---> a[s5] = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
a[s6] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1] ---> a[s6] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
a[s7] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1] ---> a[s7] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
a[s8] = [0, 0, 0, 0, 0, 0, 0, 1, 0, 0] ---> a[s8] = [0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
```

▶ **정책 이터레이션은 두 번째 스텝에서 종료되었음을 확인함.**

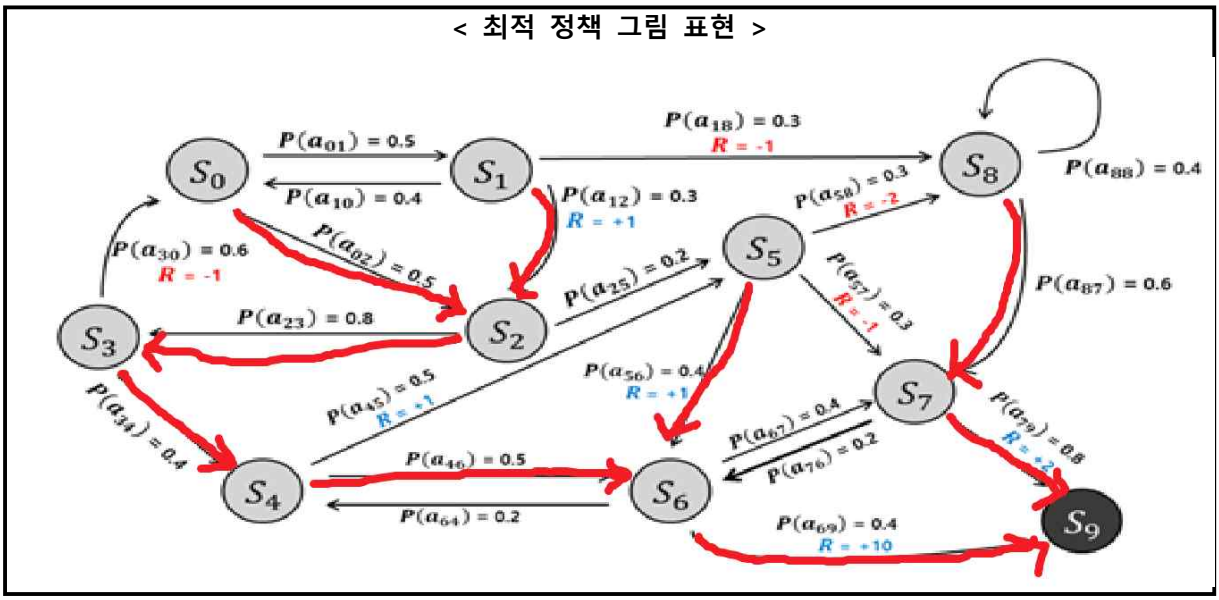
▶ **정책 이터레이션의 두 번째 스텝은 1번의 정책 평가 이후 바로 발전으로 넘어감.**

```
Policy[s0] => s2      V[0] = 1.5690646629261282
Policy[s1] => s2      V[1] = 1.6583257228614343
Policy[s2] => s3      V[2] = 1.8292156752491837
Policy[s3] => s4      V[3] = 1.8221242698383553
Policy[s4] => s6      V[4] = 4.374526062987806
Policy[s5] => s6      V[5] = 2.8750123408896173
Policy[s6] => s9      V[6] = 5.735045648129082
Policy[s7] => s9      V[7] = 2.632308216663235
Policy[s8] => s7      V[8] = 2.221010032568757
```

$V(S_0) = 1.56987137$
$V(S_1) = 1.65885766$
$V(S_2) = 1.82974539$
$V(S_3) = 1.82255994$
$V(S_4) = 4.37452611$
$V(S_5) = 2.87501237$
$V(S_6) = 5.73504566$
$V(S_7) = 2.63230822$
$V(S_8) = 2.22101006$
$V(S_9) = 0$

▶ **최적 정책이 구해졌으며, 이에 대한 가치함수의 값을 과제1의 가치함수 결과와 비교한 결과 두 값이 매우 비슷함을 확인하였음.**

< 최적 정책 그림 표현 >


S_0
S_1
S_2
S_3
S_4
S_5
S_6
S_7
S_8
S_9
$P(a_{01}) = 0.5$
$P(a_{10}) = 0.4$
$P(a_{18}) = 0.3$
$R = -1$
$P(a_{88}) = 0.4$
$P(a_{12}) = 0.3$
$R = +1$
$P(a_{02}) = 0.5$
$P(a_{30}) = 0.6$
$R = -1$
$P(a_{58}) = 0.3$
$R = -2$
$P(a_{25}) = 0.2$
$P(a_{87}) = 0.6$
$P(a_{57}) = 0.3$
$R = -1$
$P(a_{23}) = 0.8$
$P(a_{56}) = 0.4$
$R = +1$
$P(a_{34}) = 0.4$
$P(a_{45}) = 0.5$
$R = +1$
$P(a_{67}) = 0.4$
$P(a_{76}) = 0.2$
$P(a_{79}) = 0.8$
$R = +2$
$P(a_{46}) = 0.5$
$P(a_{64}) = 0.2$
$P(a_{69}) = 0.4$
$R = +10$