

## 확률 관련 문제 해결

관심 사건에 대한 확률을 프로그램을 사용해 효율적으로 계산하는 방법을 게임 예제를 통해 이해

01. 문제에서 사용하는 정의와 조건 이해: Monopoly 게임의 규칙

02. 작은 게임에 대한 확률 계산해 보기

03. 트리 구성과 확률 계산 자동화 (iterative method)

```
for(i=0;i<n;i++) {  
    // 확률 계산 iteration  
}
```

04. 예외 규칙 적용해 확률 계산해 보기

05. 실습 문제 풀이 & 질의 응답

- 상태(state): 게임을 구성하는 여러가지 상황의 조합 (예: 지도에서의 현재 위치, player의 건강치, 갖고 있는 아이템 등)
- 컴퓨터 프로그램(게임 포함)은 프로그램이 가질 수 있는 상태를 정점으로, 이들 간의 전이 가능성을 간선으로 하는 그래프로 표현할 수 있으며, 간선에는 전이 확률이 있음
- 이러한 모델을 사용해 여러 다른 상태에 도달할 확률을 계산해 보면 프로그램의 특성을 더 정확히 분석하고 분석 결과에 따른 조정 가능 (예: 게임의 난이도 확인해 규칙 조정)



## 문제 정의: Monopoly 게임의 확률

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

목표: Monopoly 게임에서 **보드의 각 칸에 도달할 확률**을 효율적으로 계산하는 방법 설계. 특히 충분히 긴 시간 게임을 진행했을 때 각 칸에 도달할 **평균 확률**을 계산하고자 함. 이러한 확률을 알면 게임의 속성(예: 난이도)을 파악하고 이에 따라 규칙을 수정할 수 있음

이 게임의 진행 규칙은 다음과 같다.

- 각 players는 **GO 칸에서 시작**하며, **2개의 N-sided dice**를 던져 나온 수의 합만큼 **시계방향**으로 움직인다. 단 **세 종류의 칸 (G2J, CC, 혹은 CH)에 도달**하면 이어지는 페이지에 설명한 action을 취한다.



## Monopoly 게임 이동 규칙 (1) - G2J

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP


목표: Monopoly 게임에서 **보드의 각 칸에 도달할 확률** 계산

- 각 players는 **GO 칸에서 시작**하며, **2개의 N-sided dice**를 던져 나온 수의 합만큼 시계방향으로 움직인다. 단 **세 종류의 칸 (G2J, CC, 혹은 CH)에 도달**하면 다음과 같은 action을 취한다.

① **G2J(Go To Jail)**: 이 칸에 도달한 player는 JAIL 칸으로 이동한다. 그리고 다음에 같은 player가 주사위를 던질 차례가 되면 바로 JAIL에서 벗어난다. 원래 monopoly 게임의 규칙은 주사위를 던져 double이 나오면 JAIL에서 벗어날 수 있으며 또는 돈을 내고 벗어날 수도 있다. 그렇지 않다면 JAIL에 계속 머무른다. 우리는 확률 계산을 좀 더 간단히 하기 위해 둘 중 한 방법을 사용해 다음 차례에 JAIL을 벗어난다고 가정한다.



## Monopoly 게임 이동 규칙 (2) - CC

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

목표: Monopoly 게임에서 **보드의 각 칸에 도달할 확률** 계산


- 각 players는 **GO 칸에서 시작**하며, **2개의 N-sided dice**를 던져 나온 수의 합만큼 시계방향으로 움직인다. 단 **세 종류의 칸 (G2J, CC, 혹은 CH)에 도달**하면 다음과 같은 action을 취한다.

② **CC(Community Chest)**: CC 칸에 도달하면 16장의 CC 카드 중 가장 위 카드를 뽑아 뒷면의 지시대로 행동한 후, 이를 가장 아래에 넣는다. 이들은 임의의 순서로 섞고 게임을 시작한다. 16개의 CC 카드 중 2장이 아래와 같이 다시 이동하도록 지시하며, 그 외 14개의 카드를 뽑으면 CC 칸에 그대로 머무른다.

- GO로 이동 (1장)
- JAIL로 이동 (1장)



# Monopoly 게임 이동 규칙 (3) - CH

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

③ **CH(CHance)**: CH 칸에 도달하면 16장의 CH 카드 중 가장 위 카드를 뽑아 뒷면의 지시대로 행동한 후, 이를 다시 가장 아래에 넣는다. 이들은 임의의 순서로 섞고 게임을 시작한다. 16개의 CH 카드 중 10장이 아래와 같이 다시 이동하도록 지시하며, 그 외 6개 카드를 뽑으면 CH 칸에 그대로 머무른다.

- Go로 이동 (1장)
- JAIL로 이동 (1장)
- C1으로 이동 (1장)
- E3으로 이동 (1장)
- H2로 이동 (1장)
- R1로 이동 (1장)
- 다음 R 칸으로 이동 (2장) - R은 'Railway company'를 의미
- 다음 U 칸으로 이동 (1장) - U는 'Utility company'를 의미
- 3칸 뒤로(반시계 방향) 이동 (1장)



## 문제 정의: Monopoly 게임의 확률

G0	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

목표: Monopoly 게임에서 **보드의 각 칸에 도달할 확률**을 효율적으로 계산하는 방법 설계. 특히 **충분히 긴 시간** 게임을 진행했을 때 각 칸에 도달할 **평균 확률**을 계산하고자 함. 이러한 확률을 알면 게임의 속성(예: 난이도)을 파악하고 이에 따라 규칙을 수정할 수 있음

각 players는 G0 칸에서 시작하며, 2개의 N-sided dice를 던져 나온 수의 합만큼 시계방향으로 움직인다. 단 세 종류의 칸(G2J, CC, 혹은 CH)에 도달하면 앞에서 설명한 action을 취한다.

※ 어떤 칸에 도달할 확률은 **주사위를 한 번 던지고 이동을 마쳤을 때 그 칸에 도달할 확률**을 말한다. 예를 들어 주사위를 한 번 던져 U2→G2J→JAIL에서 이동을 마쳤다면 JAIL에 도달한 것으로 보며, G2J에 도달한 것으로 보지 않는다.

※ 어떤 칸에 도달할 확률은 **주사위를 한 번 던지고 이동을 마쳤을 때 그 칸에 도달할 확률**을 말한다.  
 예를 들어 주사위를 한 번 던져  $U2 \rightarrow G2J \rightarrow JAIL$ 에서 이동을 마쳤다면 JAIL에 도달한 것으로 보며,  
 G2J에 도달한 것으로 보지 않는다.

(Q) 게임의 마지막 규칙(※ 부분)부터 이해해 보자.  
 이 규칙에 따르면 G2J(Go To Jail)에 도달할 확률은 무엇인가?

(Q) 앞 문제의 답을 고려할 때, G2J 외에 **또 어떤 칸에 도달할 확률이 낮을까?**

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

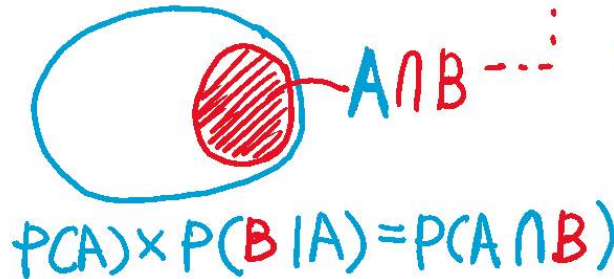
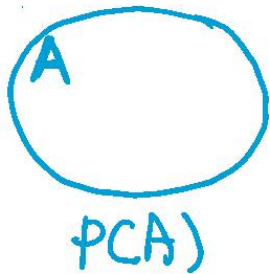




\* 확률거리) 곱하는 경우:

부분집합을 구할 때

"그리고(and)"의 의미 표현할 때



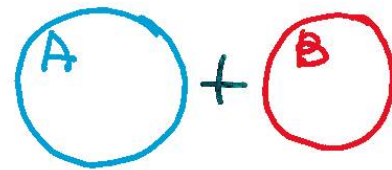
A라는 작은 세상에서  
표본공간보다  
B가 발생할 확률

→ A뿐 아니라 B도 발생한 상황  
즉 "A and B"가 발생함  
이 사건은 A의 "부분집합"  
따라서 곱함으로써  
확률이 더 작아짐.





\* 확률이 더하는 경우  
합집합을 구할 때  
"또는(or)"의 의미 표현할 때



:  $A \cup B$

→ A 또는 B가 발생한 상황  
즉 "A or B"가 발생함  
이 사건은 A보다 큰 사건  
따라서 더함으로써  
확률이 더 커짐

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

이벤트 시간에는 이 부분은 0.

A와 B가 배타적인(exclusive)  
경우만 다루므로.



## 문제 풀이 과정

- ① 이번 시간 문제를 잘 이해하기 위해 간단한 경우부터 시작해 문제 이해하고 답 이끌어내 보기



- ② 이번 시간 문제에 대한 풀이 방법 생각



- ③ 생각한 풀이 방법을 보다 효율적으로 개선

문제 명확히 이해하는데 도움

이 과정에서 문제에 대한 해법도  
생각해 낼 수 있음

이 게임에서는 각 player가 자신의 차례에 2개의 N-sided dice를 던지며 진행한다. 따라서 이로부터 나올 수 있는 경우들과 이들의 확률부터 구해 보자. 작은 값에서 시작해 확률을 구한 후, 임의의 N에 대해 일반화해 보자.

(Q)  $N = 2$ 인 경우에 대해 생각해 보자. **2개의 2-sided dice**를 던졌을 때 나올 수 있는 경우를 모두 나열해 보시오.

(1, 1), (1, 2), (2, 1), (2, 2)

(Q) 앞 문제의 각 경우에 대한 확률은 무엇인가?

(1, 1), (1, 2), (2, 1), (2, 2)  
 $1/4, 1/4, 1/4, 1/4$

(Q)  $N = 6$ 인 경우에 대해 생각해 보자. 주사위를 던졌을 때 나올 수 있는 모든 가능한 경우 및 이들의 확률은 무엇인가?

36가지

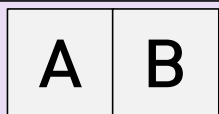
(Q) 지금까지 구한 결과를 **임의의 N에 대해 일반화**해 보자. 2 개의 N-sided dice를 던졌을 때 나올 수 있는 모든 가능한 경우 및 이들의 확률은 무엇인가?

$$1 \sim M / \text{pow}(M, 2)$$

$$M = M - 1 / \text{pow}(M, 2)$$

$$M + 1 = M / \text{pow}(M, 2)$$

$$2M = 1 / \text{pow}(M, 2)$$



### <Monopoly 게임을 2개 칸으로 단순화한 게임>

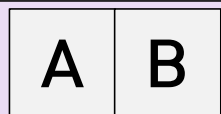
각 player는 A 칸에서 시작하며 하나의 2-sided die를 던져 나온 수만큼 시계방향으로 움직인다. 이 외에 추가적인 action을 취하는 때는 없다.

(Q) 주사위를 **처음 한 번** 던졌을 때 player가 갈 수 있는 칸은 무엇이며, 이들에 갈 확률은 무엇인가?

A: 0.5  
B: 0.5

(Q) 주사위를 **두 번째로** 던졌을 때 player가 갈 수 있는 칸은 무엇이며, 이들에 갈 확률은 무엇인가?

A:  $0.5 * 0.5 + 0.5 * 0.5$   
B:  $0.5 * 0.5 + 0.5 * 0.5$

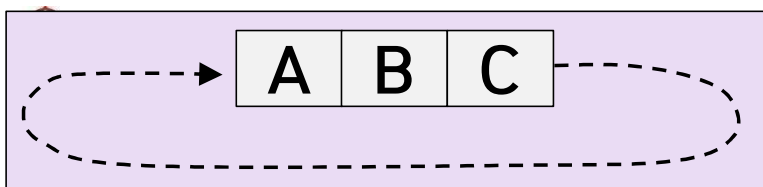


### <Monopoly 게임을 2개 칸으로 단순화한 게임>

각 player는 A 칸에서 시작하며 하나의 2-sided die를 던져 나온 수만큼 시계방향으로 움직인다. 이 외에 추가적인 action을 취하는 때는 없다.

(Q) 앞에서 구한 해를 일반화 해보자. 주사위를 M 번째로 던졌을 때 player가 갈 수 있는 칸은 무엇이며, 이들에 갈 확률은 무엇인가?



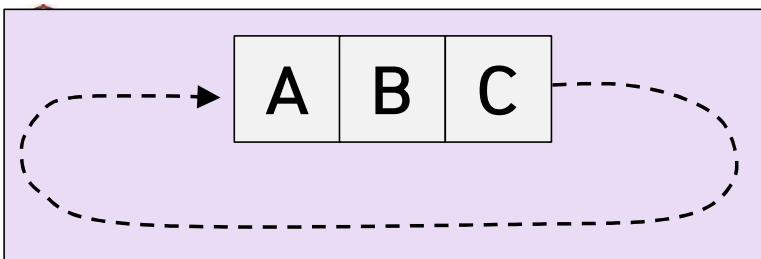


### <Monopoly 게임을 3개 칸으로 단순화한 게임>

각 player는 A 칸에서 시작해 2-sided die 하나를 던져 나온 수만큼 시계방향으로 움직이며, 이 외에 예외 규칙은 없다.

(Q) 주사위를 **4번째로 던졌을 때** player가 갈 수 있는 칸은 무엇이며, 이들에 갈 확률은 무엇인가?  
가능성 트리를 그린 후 1번째, 2번째, 3번째, 4번째 roll 차례로 확률을 계산해 보자.

1번째	2번째
A: 1/2	A: 1/2
B: 1/2	B: 1/2
C: 0	C: 0



### <Monopoly 게임을 3개 칸으로 단순화한 게임>

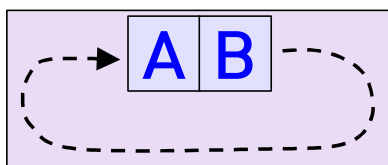
각 player는 A 칸에서 시작하며 하나의 2-sided die를 던져 나온 수만큼 시계방향으로 움직인다. 이 외에 추가적인 action을 취하는 때는 없다.

(Q) 앞에서 구한 해를 일반화 해보자.

주사위를  $M$  번째로 던졌을 때 player가 갈 수 있는 칸은 무엇이며, 이들에 갈 확률은 무엇인가?

(Q) 이제 앞에서 본 두 게임에 대한 확률을 **iterative method**로 계산하는 프로그램을 실행해 보자.  
칸의 수를 2개에서 시작해 점차 증가시켜 보며 확률이 어떻게 변하는지 확인해 보자.

```
if __name__ == "__main__":  
    # Run a simple game with no exceptional rules  
    # simpleGame(칸 수, 주사위 면 수 N, 주사위 던진 횟수, 계산 과정 출력 여부 True or False)  
    simpleGame(2, 2, 10, True)
```



(Q) 이제 Monopoly 게임에 대해 생각해 보자. **2개의 6-sided dice**를 던져 이동하며, **이동 후 추가적인 action은 취하지 않는다고 가정**하자. 각 칸에 도달할 확률은 어떻게 될 것으로 예상하는가?



## 지정된 iteration 후 확률의 변화를 계산하는 방법

$P_i[s]$ :  $i$ 번째 iteration 후에 state  $s$ 에 도달할 확률

초기 state  $s_0$ 에 대해서는  $P_0[s_0] = 1$ 이며 그 외 state  $s_k$ 에 대해서는  $P_0[s_k] = 0$

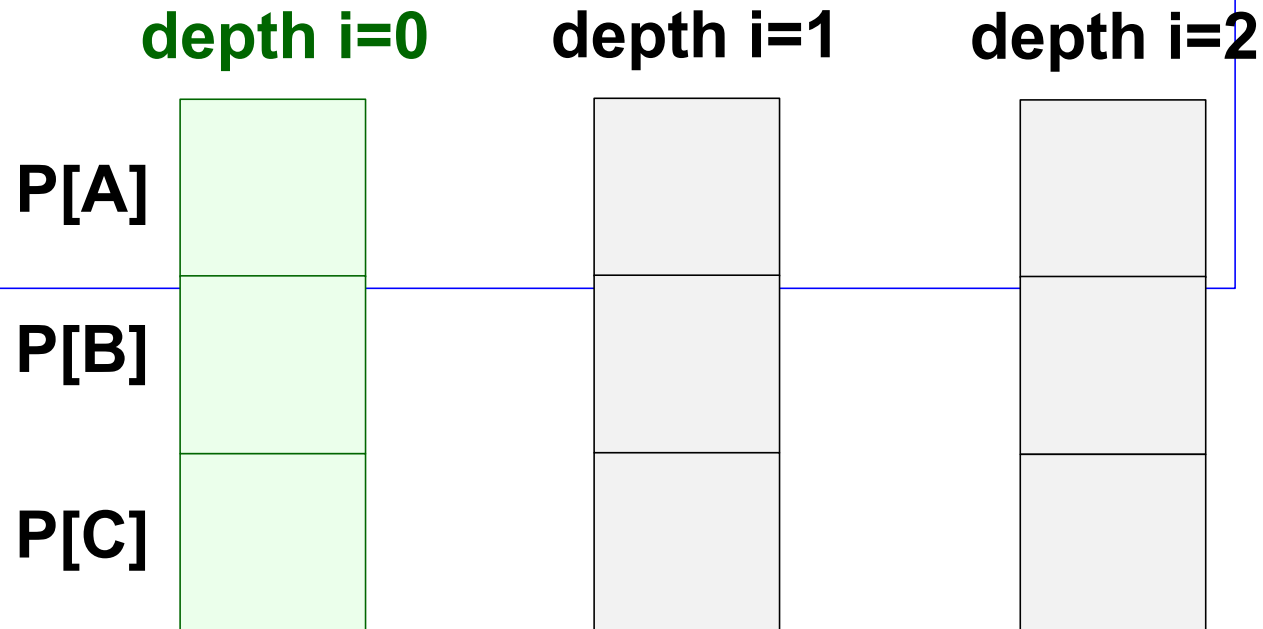
$P_{\text{transition}}[sx][sy]$ : 한 iteration 후에 state  $s$ 에서  $sy$ 로 이동할 확률

for each iteration  $i (\geq 1)$

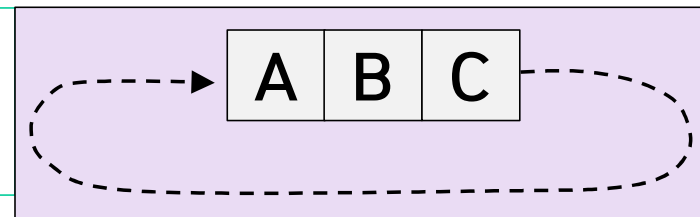
$P_i[sy] = 0$  for all  $sy$ 로 초기화

for each state pair  $(sx, sy)$

$$P_i[sy] = P_i[sy] + P_{\text{transition}}[sx][sy] \times P_{i-1}[sx]$$



(Q) 3칸 게임에 iterative method를 적용해 보자. 어떤 state가 존재하는가? 각 state에 대한 초기 확률값  $P_0[s_x]$ 는 무엇인가? 또한, state 간 이동 확률  $P_{\text{transition}}[s_x][s_y]$ 는 무엇인가?



Iteration  $i = 0$  (initial stage)

$P_i[A]$	
$P_i[B]$	
$P_i[C]$	

Iteration I    Iteration I+1

	A
A	B
	C

Iteration I    Iteration I+1

	A
B	B
	C

Iteration I    Iteration I+1

	A
C	B
	C

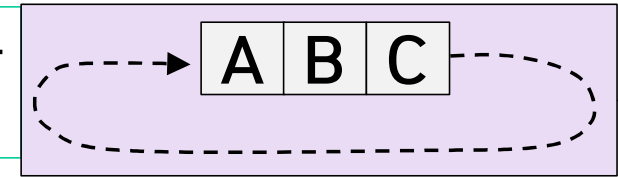
$P_i[s_x]$ :  $i$ 번째 iteration 후에 state  $s_x$ 에 도달할 확률

초기 state  $s_0$ 에 대해서는  $P_0[s_0] = 1$ 이며 그 외 state  $s_i$ 에 대해서는  $P_0[s_i] = 0$

$P_{\text{transition}}[s_x][s_y]$ : 한 iteration 후에 state  $s_x$ 에서  $s_y$ 로 이동할 확률



(Q) 앞에서 구한  $P_0[s_x]$ 와  $P_{\text{transition}}[s_x][s_y]$ 을 사용해 3번의 iteration을 거치며  $P_i[s_x]$ 가 어떻게 변해가는지 보이시오.



21

Iteration i	0	1	2	3
$P_i[A]$	1	0	$1/4 + 1/4$	$1/8 + 1/8$
$P_i[B]$	0	$1/2$	$1/4$	$1/4 + 1/8$
$P_i[C]$	0	$1/2$	$1/4$	$1/4 + 1/8$
Sum of prob. 1				

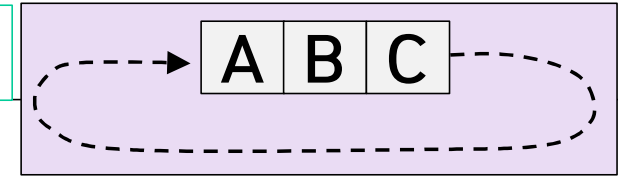
for each iteration  $i (\geq 1)$

$P_i[sy] = 0$  for all  $sy$ 로 초기화

for each state pair  $(sx, sy)$

$$P_i[sy] = P_i[sy] + P_{\text{transition}}[sx][sy] \times P_{i-1}[sx]$$

(Q) 4번째와 5번째 iteration 후에는  $P_i[s_x]$ 가 어떻게 변하는가?



22

Iteration #	3	4	5
$P_i[A]$	$1/8 + 1/8$	$3/16 + 3/16$	
$P_i[B]$	$1/4 + 1/8$	$1/8 + 3/16$	
$P_i[C]$	$1/4 + 1/8$	$1/8 + 3/16$	
Sum of prob.	1		

for each iteration  $i (\geq 1)$

$P_i[s_y] = 0$  for all  $s_y$ 로 초기화

for each state pair  $(s_x, s_y)$

$$P_i[s_y] = P_i[s_y] + P_{\text{transition}}[s_x][s_y] \times P_{i-1}[s_x]$$



## 정리: 지정된 iteration 후 확률의 변화를 계산하는 방법

$P_i[sx]$ :  $i$ 번째 iteration 후에 state  $sx$ 에 도달할 확률

초기 state  $s0$ 에 대해서는  $P_0[s0] = 1$ 이며 그 외 state  $si$ 에 대해서는  $P_0[si] = 0$

$P_{\text{transition}}[sx][sy]$ : 한 iteration 후에 state  $sx$ 에서  $sy$ 로 이동할 확률

for each iteration  $i (\geq 1)$

$P_i[sy] = 0$  for all  $sy$ 로 초기화

for each state pair  $(sx, sy)$

$$P_i[sy] = P_i[sy] + P_{\text{transition}}[sx][sy] \times P_{i-1}[sx]$$

- 가능성 트리를 따라 내려가며 확률을 계산하는 과정
- 가능성 트리의 depth = iteration  $i$
- 각 depth에서 한 state  $sx$ 를 여러 노드로 분리하지 않고 하나로 병합함으로써 node 개수가 빠르게 증가하는 것을 막고 항상 일정한 개수의 node가 존재하도록 함

$P_i[sx]$ :  $i$ 번째 iteration 후에 state  $sx$ 에 도달할 확률

초기 state  $s0$ 에 대해서는  $P_0[s0] = 1$ 이며 그 외 state  $si$ 에 대해서는  $P_0[si] = 0$

$P_{\text{transition}}[sx][sy]$ : 한 iteration 후에 state  $sx$ 에서  $sy$ 로 이동할 확률

for each iteration  $i (\geq 1)$

$P_i[sy] = 0$  for all  $sy$ 로 초기화

for each state pair  $(sx, sy)$

$$P_i[sy] = P_{i-1}[sy] + P_{\text{transition}}[sx][sy] \times P_{i-1}[sx]$$

# simpleGame(칸 수, 주사위 면 수  $N$ , 주사위 던진 횟수, 계산 과정 출력 여부)

def simpleGame(numSquares, numSides, numIterations, debug):

    probCurrent = [0 for \_ in range(numSquares)]

    probCurrent[0] = 1 # game begins from square 0

    probTransition = []

    for  $sx$  in range(numSquares):

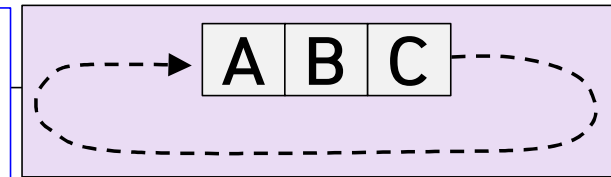
        probTransitionSx = [0 for \_ in range(numSquares)]

        for  $n$  in range(1, numSides + 1):

            probTransitionSx[(sx + n) % numSquares] += 1/numSides

        probTransition.append(probTransitionSx)

# 다음 페이지에 이어짐



$P_0[A]$

$P_0[B]$

$P_0[C]$

Iteration I	Iteration I+1
	A
A	B
	C

Iteration I	Iteration I+1
	A
B	B
	C

Iteration I	Iteration I+1
	A
C	B
	C

$P_i[sx]$ :  $i$ 번째 iteration 후에 state  $sx$ 에 도달할 확률

초기 state  $s0$ 에 대해서는  $P_0[s0] = 1$ 이며 그 외 state  $si$ 에 대해서는  $P_0[si] = 0$

$P_{\text{transition}}[sx][sy]$ : 한 iteration 후에 state  $sx$ 에서  $sy$ 로 이동할 확률

for each iteration  $i (\geq 1)$

$P_i[sy] = 0$  for all  $sy$ 로 초기화

for each state pair  $(sx, sy)$

$$P_i[sy] = P_i[sy] + P_{\text{transition}}[sx][sy] \times P_{i-1}[sx]$$

```
def simpleGame(numSquares, numSides, numIterations, debug):
```

```
    # 앞 페이지에서 이어짐
```

```
    for i in range(numIterations):
```

```
        # probCurrent, probNext:  $P_{i-1}[]$ ,  $P_i[]$ 
```

```
        probNext = [0 for _ in range(len(probCurrent))]
```

```
        for sx in range(len(probTransition)):
```

```
            for sy in range(len(probTransition[sx])):
```

```
                probNext[sy] += probTransition[sx][sy] * probCurrent[sx]
```

```
        probCurrent = probNext
```

```
        if debug: print(probCurrent)
```

```
    return probCurrent
```

Iteration i	0	1	2	3
$P_i[A]$				
$P_i[B]$				
$P_i[C]$				
Sum of prob.	1			

게임 규칙에 맞게  
초기 확률 probCurrent와  
전이 확률 probTransition이 잘 준비되었다면  
재활용 가능한 코드

이제 iterative method를 Monopoly 게임에 적용해 보자. 이를 위해 ① 어떤 state  $s_x$ 가 존재하고, ② 각 state의 초기 확률  $P_0[s_x]$ 는 무엇이며, ③ state 간 이동 확률  $P_{\text{transition}}[s_x][s_y]$ 은 무엇인지를 확인하면 될 것이다. ①~③을 결정하였다면 앞에서 본 방법대로 진행하며 확률을 계산하면 된다.

(Q) Monopoly 게임에서 확률을 계산하고자 하는 각 state  $s_x$ 는 무엇인가? 또한, 이들에 대한 초기 확률  $P_0[s_x]$ 는 무엇인가?

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP



(Q) G2J, CC, CH 칸에 대한 **예외 규칙이 (추가 이동 규칙) 없다**고 가정하자. 또한, **앞으로의 문제에서는 계속 2개의 6-sided dice**를 사용한다고 가정하자. 이럴 때 GO 칸에서 각 state sy로 이동할 확률  $P_{\text{transition}}[\text{Go}][\text{sy}]$ 는 무엇인가?

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP



# Monopoly 게임 이동 규칙 (1) - G2J

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

목표: Monopoly 게임에서 **보드의 각 칸에 도달할 확률** 계산

- 각 players는 **GO 칸에서 시작**하며, **2개의 N-sided dice**를 던져 나온 수의 합만큼 시계방향으로 움직인다. 단 **세 종류의 칸 (G2J, CC, 혹은 CH)에 도달**하면 다음과 같은 action을 취한다.

① **G2J(Go To Jail)**: 이 칸에 도달한 player는 JAIL 칸으로 이동한다. 그리고 다음에 같은 player가 주사위를 던질 차례가 되면 바로 JAIL에서 벗어난다. 원래 monopoly 게임의 규칙은 주사위를 던져 double이 나오면 JAIL에서 벗어날 수 있으며 또는 돈을 내고 벗어날 수도 있다. 그렇지 않다면 JAIL에 계속 머무른다. 우리는 확률 계산을 좀 더 간단히 하기 위해 둘 중 한 방법을 사용해 다음 차례에 JAIL을 벗어난다고 가정한다.

(Q) 이제 **G2J에 대한 예외 규칙을 추가**해 보자. 즉 주사위를 던져 **G2J로 가게 되면 JAIL로 이동**한다고 하자. 이럴 때 **E2**에서 각 state  $s_y$ 로 이동할 확률  $P_{\text{transition}}[\mathbf{E2}][s_y]$ 는 무엇인가?

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

※ 어떤 칸에 도달할 확률은 **주사위를 한 번 던지고 이동을 마쳤을 때 그 칸에 도달할 확률**을 말한다. 예를 들어 주사위를 한 번 던져  $U2 \rightarrow G2J \rightarrow JAIL$ 에서 이동을 마쳤다면 G2J가 아닌 JAIL에 도달한 것으로 본다.


(Q) 이제 **G2J에 대한 예외 규칙을 추가**해 보자. 즉 주사위를 던져 **G2J로 가게 되면 JAIL로 이동**한다고 하자. 이럴 때 **U2**에서 각 state  $s_y$ 로 이동할 확률  $P_{\text{transition}}[\mathbf{U2}][s_y]$ 는 무엇인가?

1	GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL	1
2	H2										C1	
3	T2										U1	
4	H1										C2	
5	CH3										C3	
6	R4										R2	
5	G3										D1	
4	CC3										CC2	
3	G2										D2	
2	G1										D3	
0	G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP	

※ 어떤 칸에 도달할 확률은 **주사위를 한 번 던지고 이동을 마쳤을 때 그 칸에 도달할 확률**을 말한다. 예를 들어 주사위를 한 번 던져  $U2 \rightarrow G2J \rightarrow JAIL$ 에서 이동을 마쳤다면 G2J가 아닌 JAIL에 도달한 것으로 본다.



## Monopoly 게임 이동 규칙 (2) - CC

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

목표: Monopoly 게임에서 **보드의 각 칸에 도달할 확률** 계산

- 각 players는 **GO 칸에서 시작**하며, **2개의 N-sided dice**를 던져 나온 수의 합만큼 시계방향으로 움직인다. 단 **세 종류의 칸 (G2J, CC, 혹은 CH)에 도달**하면 다음과 같은 action을 취한다.

② **CC(Community Chest)**: CC 칸에 도달하면 16장의 CC 카드 중 가장 위 카드를 뽑아 뒷면의 지시대로 행동한 후, 이를 가장 아래에 넣는다. 이들은 임의의 순서로 섞고 게임을 시작한다. 16개의 CC 카드 중 2장이 아래와 같이 다시 이동하도록 지시하며, 그 외 14개의 카드를 뽑으면 CC 칸에 그대로 머무른다.

- GO로 이동 (1장)
- JAIL로 이동 (1장)

(Q) CC에 대한 예외 규칙을 추가하자. C1에서 각 state  $s_y$ 로 이동할 확률  $P_{\text{transition}}[\text{C1}][s_y]$ 는?

② CC(Community Chest):

-GO로 이동 (1/16 카드)

-JAIL로 이동 (1/16 카드)

$$5/36 * 1/16$$

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

$$5/36 * 1/16$$

1

2

3

4

$$5/36 * 14/16$$

6

5

4

1 2 3

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP



(Q) CC에 대한 예외 규칙이 있다고 할 때, GO에서 각 state  $s_y$ 로 이동할 확률  $P_{\text{transition}}[\text{GO}][s_y]$ 는?

② CC(Community Chest):

-GO로 이동 (1/16 카드)

-JAIL로 이동 (1/16 카드)

$$2/36 * 14/16$$

$$2/36 * 1/16$$

3 4 5

$$2/36 * 1/16$$

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

(Q) 이제 **G2J**와 **CC** 모두에 대한 예외 규칙이 추가되었다고 하자 (즉 CH에 대한 예외 규칙을 제외하고 모두 추가). 이럴 때 **U2**에서 각 state  $s_y$ 로 이동할 확률  $P_{\text{transition}}[\mathbf{U2}][s_y]$ 는?

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

① G2J 도달하면 JAIL로 이동

② CC 도달하면

-GO로 이동 (1/16)

-JAIL로 이동 (1/16)

-CC에 머무름 (14/16)

(Q) G2J와 CC 모두에 대한 예외 규칙이 추가되었다고 하자 (CH에 대한 예외 규칙을 제외하고 모두 추가). 이럴 때 R3에서 각 state  $s_y$ 로 이동할 확률  $P_{\text{transition}}[\mathbf{R3}][s_y]$ 는?

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

① G2J 도달하면 JAIL로 이동

② CC 도달하면


-GO로 이동 (1/16)

-JAIL로 이동 (1/16)

-CC에 머무름 (14/16)



## Monopoly 게임 이동 규칙 (3) - CH

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

③ **CH(CHance)**: CH 칸에 도달하면 16장의 CH 카드 중 가장 위 카드를 뽑아 뒷면의 지시대로 행동한 후, 이를 다시 가장 아래에 넣는다. 이들은 임의의 순서로 섞고 게임을 시작한다. 16개의 CH 카드 중 10장이 아래와 같이 다시 이동하도록 지시하며, 그 외 6개 카드를 뽑으면 CH 칸에 그대로 머무른다.

- Go로 이동 (1장)
- JAIL로 이동 (1장)
- C1으로 이동 (1장)
- E3으로 이동 (1장)
- H2로 이동 (1장)
- R1로 이동 (1장)
- 다음 R 칸으로 이동 (2장) - R은 'Railway company'를 의미
- 다음 U 칸으로 이동 (1장) - U는 'Utility company'를 의미
- 3칸 뒤로(반시계 방향) 이동 (1장)

(Q) 이제 ① G2J, ② CC에 대한 규칙에 더해 ③ CH에 대한 규칙도 포함하자.

GO에서 각 state  $s_y$ 로 이동할 확률  $P_{\text{transition}}[\text{GO}][s_y]$ 는?

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

① G2J 도달하면 JAIL로 이동

② CC 도달하면

-GO로 이동 (1/16)

-JAIL로 이동 (1/16)

③ CH 도달하면:

-GO로 이동 (1/16)

-JAIL로 이동 (1/16)

-C1으로 이동 (1/16)

-E3으로 이동 (1/16)

-H2로 이동 (1/16)

-R1로 이동 (1/16)

-다음 R 칸으로 이동 (2/16)

-다음 U 칸으로 이동 (1/16)

-3칸 뒤로 이동 (1/16)

(Q) ① G2J, ② CC ③ CH에 대한 규칙을 모두 포함했다면  
**G1**에서 각 state  $s_y$ 로 이동할 확률  $P_{\text{transition}}[\mathbf{G1}][s_y]$ 는?

$1/36 * 14/16$

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
<b>G1</b>										D3
G2J	F3	U2	F2	F1	R3	E3	E2	CH2	E1	FP

- ① G2J 도달하면 JAIL로 이동
- ② CC 도달하면
  - GO로 이동 (1/16)
  - JAIL로 이동 (1/16)
- ③ CH 도달하면:
  - GO로 이동 (1/16)
  - JAIL로 이동 (1/16)
  - C1으로 이동 (1/16)
  - E3으로 이동 (1/16)
  - H2로 이동 (1/16)
  - R1로 이동 (1/16)
  - 다음 R 칸**으로 이동 (**2/16**)
  - 다음 U 칸**으로 이동 (1/16)
  - 3칸 뒤**로 이동 (1/16)

$P_i[sx]$ :  $i$ 번째 iteration 후에 state  $sx$ 에 도달할 확률

초기 state  $s0$ 에 대해서는  $P_0[s0] = 1$ 이며 그 외 state  $si$ 에 대해서는  $P_0[si] = 0$

$P_{\text{transition}}[sx][sy]$ : 한 iteration 후에 state  $sx$ 에서  $sy$ 로 이동할 확률

for each iteration  $i (\geq 1)$

for each state pair  $(sx, sy)$

$$P_i[sy] = P_i[sy] + P_{\text{transition}}[sx][sy] \times P_{i-1}[sx]$$

(Q) 지금까지 계산한 확률을 사용해 구현한 iterative method를 실행해서 결과를 확인해 보자.

G2J 외에 가장 확률이 낮은 3개의 state는 무엇인가? 왜 그러한지 이해가 되는가?

(Q) 앞 문제에 이어서, JAIL과 GO에 대한 확률은 왜 그러한지 생각해 보자.



(Q) 2개의 6-sided dice 대신에 2개의 **N-sided dice**를 사용했다면 지금까지 사용한 어떤 값을 바꾸어 사용해야 하는가? 또한, N값에 따라 iterative method가 계산한 확률이 크게 달라질 것으로 예상하는가?

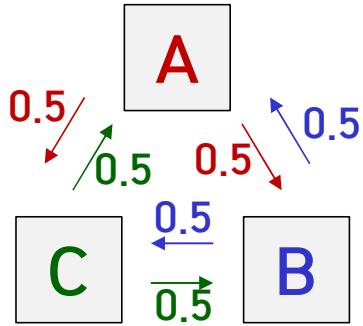
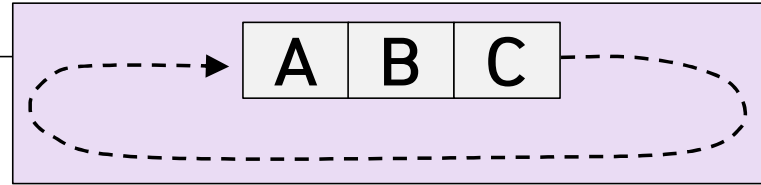


## <정리>

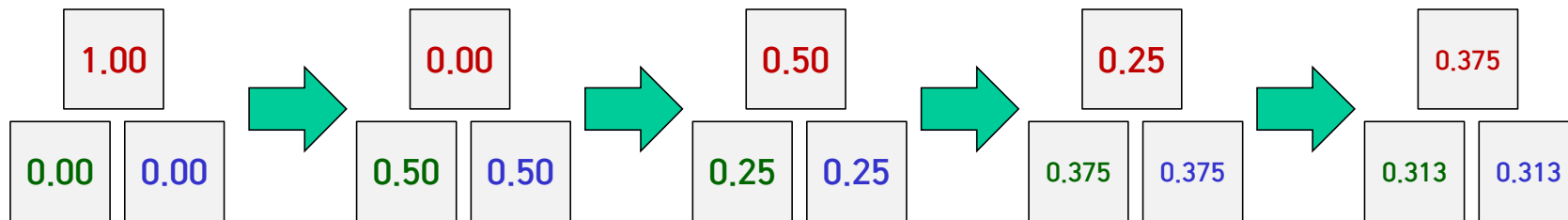
- 이번 시간에는 게임을 진행하면서 각 state에 도달할 확률을 계산해 보았음
- 계산한 확률은 게임의 규칙과 난이도 조정에 활용할 수 있음
- 게임의 규칙과 state가 복잡한 경우에는 직접 확률값을 계산해 내기가 어려우며, 이럴 때 **iterative method**를 사용함
- Iterative method는 다음과 같이 생각할 수 있음: ① **state-transition diagram** + 각 state에 있을 **초기 확률** + **state 간 이동 확률** 계산 ② 게임이 진행함에 따라 이러한 확률 값이 어떻게 여러 다른 state 간에 분배되고 이동하는지 관찰



<Summary>

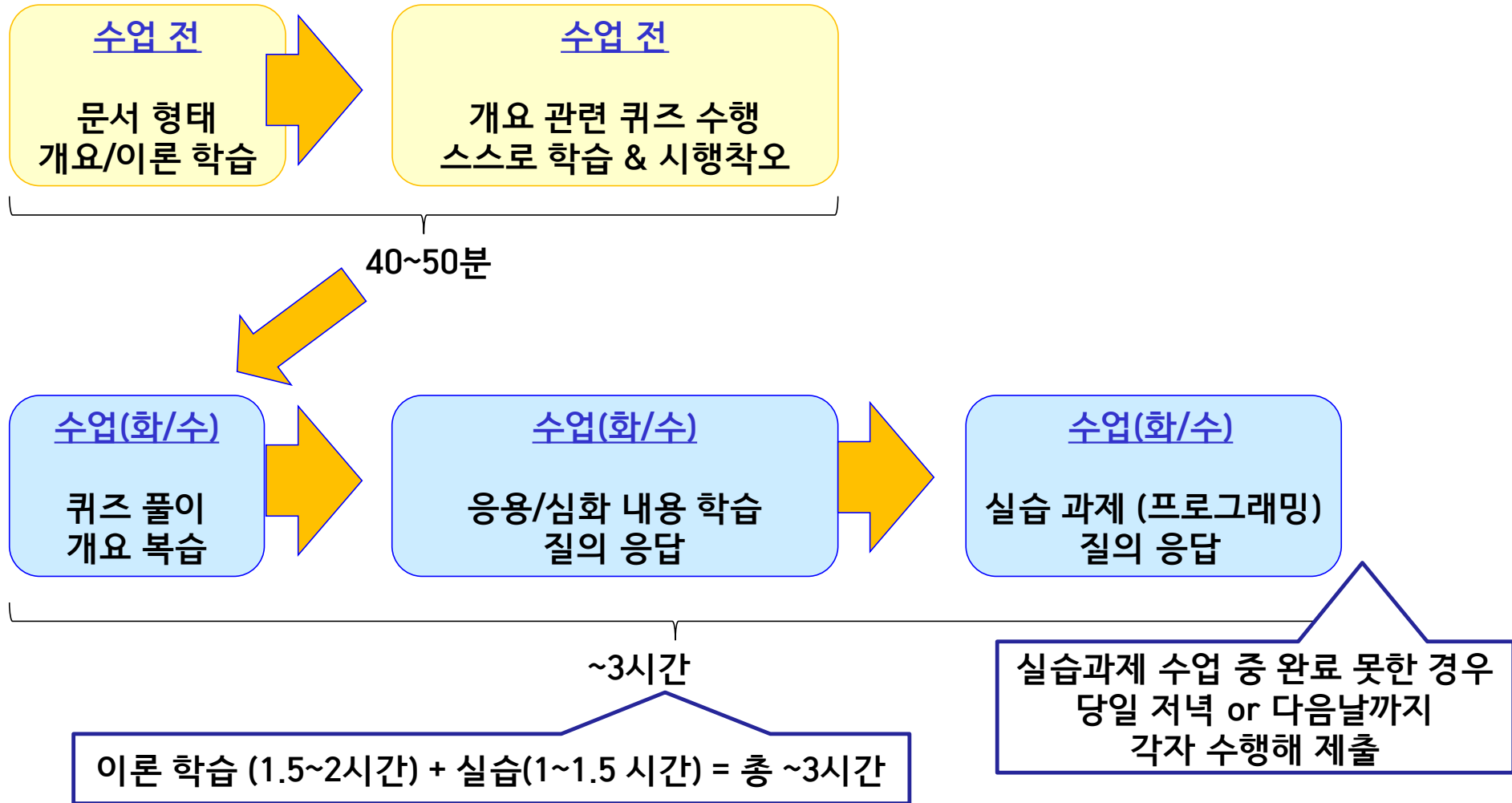


<State transition diagram>





# 스마트 출결





## 05. 실습문제풀이 & 질의응답

- 이번 시간에 배운 내용에 대한 실습 문제 풀이 & 질의 응답
- 채점 방식은 지난 시간 문제 풀이때와 유사함
- 왜 중요한가?
- 이번 주 배운 내용을 총괄하는 문제 풀이 통해 배운 내용 활용 & 복습
- 문제 풀이 점수는 이번 주 과제 점수에 포함됨

## monopoly (numSides, numIterations) 함수 구현 조건: Monopoly 게임에서 각 칸에 도달할 확률 계산

- 입력으로 주사위 면의 수와 (numSides) iteration 수 (numIterations)가 주어졌을 때
- 각 칸에 도달할 확률을 계산해 반환하는 함수 구현  

```
def monopoly (numSides, numIterations):
```
- 입력 **numSides**: 주사위 면의 수를 의미하며, 이러한 주사위 **2개**를 사용한다고 가정
- $2 \leq \text{numSides} \leq 30$  범위의 정수이며, 이 조건에 맞는 값만 입력으로 들어온다고 가정
- 입력 **numIterations**: iteration 횟수로, 몇 번의 dice roll을 했는지를 의미
- $1 \leq \text{numIterations} \leq 1000$  범위의 정수이며, 이 조건에 맞는 값만 입력으로 들어온다고 가정
- 반환 값: 각 칸에 도달할 확률을 2-tuple의 리스트로 반환하며, 각 2-tuple은 (**문자열 형식의 칸 이름**, **float 형식의 확률**). 예: `[('GO', 0.03), ('A1', 0.024), ('CC1', 0.0182), ...]`
- 이번 시간에 제공한 코드 Probability.py의 monopoly () 함수 내에 코드 작성해 제출

입출력 예: 2개의 2면 주사위를 한 번 던졌을 때 각 칸에 도달할 확률

```
print(monopoly(2, 1))
```

```
[('GO', 0.015625), ('A1', 0.0), ('CC1', 0.21875), ('A2', 0.5), ('T1', 0.25), ('R1', 0.0), ('B1', 0.0), ('CH1', 0.0), ('B2', 0.0), ('B3', 0.0), ('JAIL', 0.015625), ('C1', 0.0), ('U1', 0.0), ('C2', 0.0), ('C3', 0.0), ('R2', 0.0), ('D1', 0.0), ('CC2', 0.0), ('D2', 0.0), ('D3', 0.0), ('FP', 0.0), ('E1', 0.0), ('CH2', 0.0), ('E2', 0.0), ('E3', 0.0), ('R3', 0.0), ('F1', 0.0), ('F2', 0.0), ('U2', 0.0), ('F3', 0.0), ('G2J', 0), ('G1', 0.0), ('G2', 0.0), ('CC3', 0.0), ('G3', 0.0), ('R4', 0.0), ('CH3', 0.0), ('H1', 0.0), ('T2', 0.0), ('H2', 0.0)]
```

GO	A1	CC1	A2	T1	R1	B1	CH1	B2	B3	JAIL
H2										C1
T2										U1
H1										C2
CH3										C3
R4										R2
G3										D1
CC3										CC2
G2										D2
G1										D3

$$\begin{aligned}
 &Go \quad \frac{1}{4} \times \frac{1}{16} \\
 &CC1 \quad \frac{1}{4} \times \frac{14}{16} \\
 &A2 \quad \frac{2}{4} \\
 &T1 \quad \frac{1}{4} \\
 &JAIL \quad \frac{1}{4} \times \frac{1}{16}
 \end{aligned}$$

입출력 예: 2개의 6면 주사위를 한 번 던졌을 때 각 칸에 도달할 확률

```
print(monopoly(6, 1))
```

```
[('GO', 0.012152777777777776), ('A1', 0.0), ('CC1', 0.024305555555555552), ('A2',
0.05555555555555555), ('T1', 0.09375), ('R1', 0.12152777777777778), ('B1', 0.13888888888888889),
('CH1', 0.0625), ('B2', 0.13888888888888889), ('B3', 0.11111111111111111), ('JAIL',
0.09548611111111111), ('C1', 0.06597222222222222), ('U1', 0.03819444444444444), ('C2', 0.0), ('C3',
0.0), ('R2', 0.020833333333333332), ('D1', 0.0), ('CC2', 0.0), ('D2', 0.0), ('D3', 0.0), ('FP',
0.0), ('E1', 0.0), ('CH2', 0.0), ('E2', 0.0), ('E3', 0.010416666666666666), ('R3', 0.0), ('F1',
0.0), ('F2', 0.0), ('U2', 0.0), ('F3', 0.0), ('G2J', 0), ('G1', 0.0), ('G2', 0.0), ('CC3', 0.0),
('G3', 0.0), ('R4', 0.0), ('CH3', 0.0), ('H1', 0.0), ('T2', 0.0), ('H2', 0.010416666666666666)]
```

# 위 결과에 대한 설명: 수업에서 풀었던 GO state로부터의 전이확률 Ptransition[Go][s]와 같은 결과임

```
[("GO", 6/36*1/16+1/36*1/16), ("A1", 0), ("CC1", 1/36*14/16), ("A2", 2/36), ("T1", 3/36+6/36*1/16),
("R1", 4/36+6/36*1/16), ("B1", 5/36), ("CH1", 6/36*6/16), ("B2", 5/36), ("B3", 4/36), ("JAIL",
3/36+1/36*1/16+6/36*1/16), ("C1", 2/36+6/36*1/16), ("U1", 1/36+6/36*1/16), ("C2", 0), ("C3", 0), ("R2",
6/36*2/16), ("D1", 0), ("CC2", 0), ("D2", 0), ("D3", 0), ("FP", 0), ("E1", 0), ("CH2", 0), ("E2", 0),
("E3", 6/36*1/16), ("R3", 0), ("F1", 0), ("F2", 0), ("U2", 0), ("F3", 0), ("G2J", 0), ("G1", 0), ("G2",
0), ("CC3", 0), ("G3", 0), ("R4", 0), ("CH3", 0), ("H1", 0), ("T2", 0), ("H2", 6/36*1/16)]
```

그 외 예제는 \_\_main\_\_ 아래 테스트 코드를 참조하세요.

## 그 외 프로그램 구현 조건

- 최종 결과물로 Probability.py 파일 하나만 제출하며, 이 파일만으로 코드가 동작해야 함
- import는 사용할 수 없음
- `__main__` 아래의 코드는 작성한 함수가 올바른지 확인하는 코드로
- 코드 작성 후 스스로 채점해 보는데 활용하세요.
- 각 테스트 케이스에 대해 P(or Pass) 혹은 F(or Fail)이 출력됩니다.
- 코드를 제출할 때는 `__main__` 아래 코드는 제거하거나 수정하지 말고 제출합니다. (채점에 사용되기 때문)
- 정확도 테스트는 케이스별로 2초 이상 기다려야 하는 경우 fail로 봅니다.
- 정확도 검사 방법: (칸 이름, 확률) tuple이 모두 맞아야 하며, tuple이 리스트에 저장된 순서는 (확률 순으로 정렬 후 검사하므로) 임의의 순서여도 괜찮음. **확률은 소수점 아래 6자리까지 올바른지 확인함**
- **속도 테스트는 거의 항상 pass 해야만 통과입니다.**





## 이번 시간 제공 코드 함께 보기

- Probability.py
- Monopoly 게임의 칸 이름과 순서는 리스트 squares에 미리 작성된 대문자 이름 그대로 사용

```
def monopoly(numSides, numIterations):  
    squares = ["GO", "A1", "CC1", "A2", "T1", "R1", "B1", "CH1", "B2", "B3", \  
               "JAIL", "C1", "U1", "C2", "C3", "R2", "D1", "CC2", "D2", "D3", \  
               "FP", "E1", "CH2", "E2", "E3", "R3", "F1", "F2", "U2", "F3", \  
               "G2J", "G1", "G2", "CC3", "G3", "R4", "CH3", "H1", "T2", "H2"]
```



## 실습 문제 풀이 & 질의 응답

- 종료 시간(17:00) 이전에 **일찍 모든 문제를 통과한 경우** 각자 퇴실 가능
- 실습 문제에 대한 코드는 lms 과제함에 **다음 날 23:59까지 제출** 가능합니다.
- 마감 시간까지 작성을 다 못한 경우는 (제출하지 않으면 0점이므로) 그때까지 작성한 코드를 꼭 제출해 부분점수를 받으세요.
- 실습 문제는 **개별 평가**입니다. 제출한 코드에 대한 유사도 검사를 실습 문제마다 진행하며, 코드를 건네 준 사례가 발견되면 0점 처리됩니다.
- 로직에 대해 서로 의견을 나누는 것은 괜찮지만, 코드를 직접 건네 주지는 마세요.
- **채점 시 정확도/속도 비교에 사용하는 함수를 그대로 복사해 제출하거나 채점 코드에 대해서만 올바른 답을 반환하도록 작성한 경우에도 점수가 없습니다.**
- 본인 실력 향상을 위해서도 **코드는 꼭 각자 직접 작성**해 주세요.