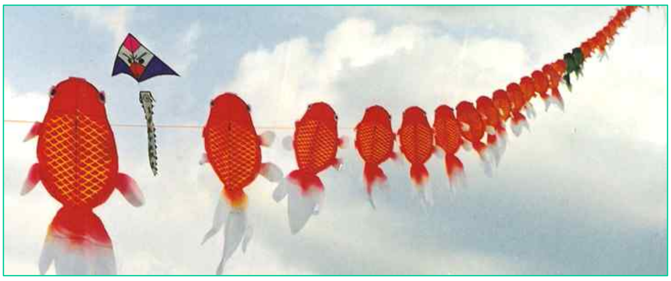
Dynamic Programming (DP)

01. Dynamic Programming(DP, 동적 프로그래밍)이 무엇인가?

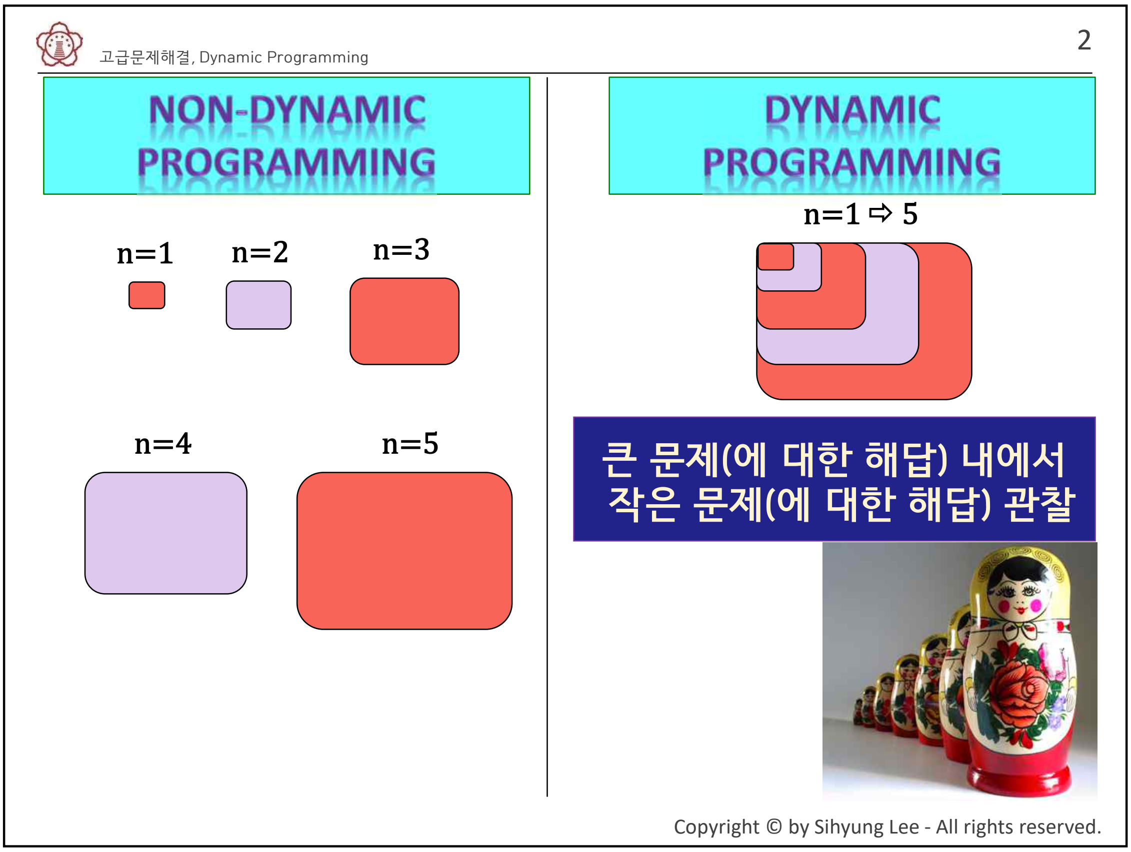
02. P(N)을 구하기 위해 P(N-1)을 활용하는 예

03. P(N)을 구하기 위해 P(N-i), i>1,를 활용하는 예



이번주에는 많이 사용되는 알고리즘 기법 중 하나인 Dynamic Programming에 대해 배워보겠습니다.

특히 다양한 예를 보며 Dynamic Programming의 개념과 특성에 익숙해 지는 것을 목표로 하겠습니다.

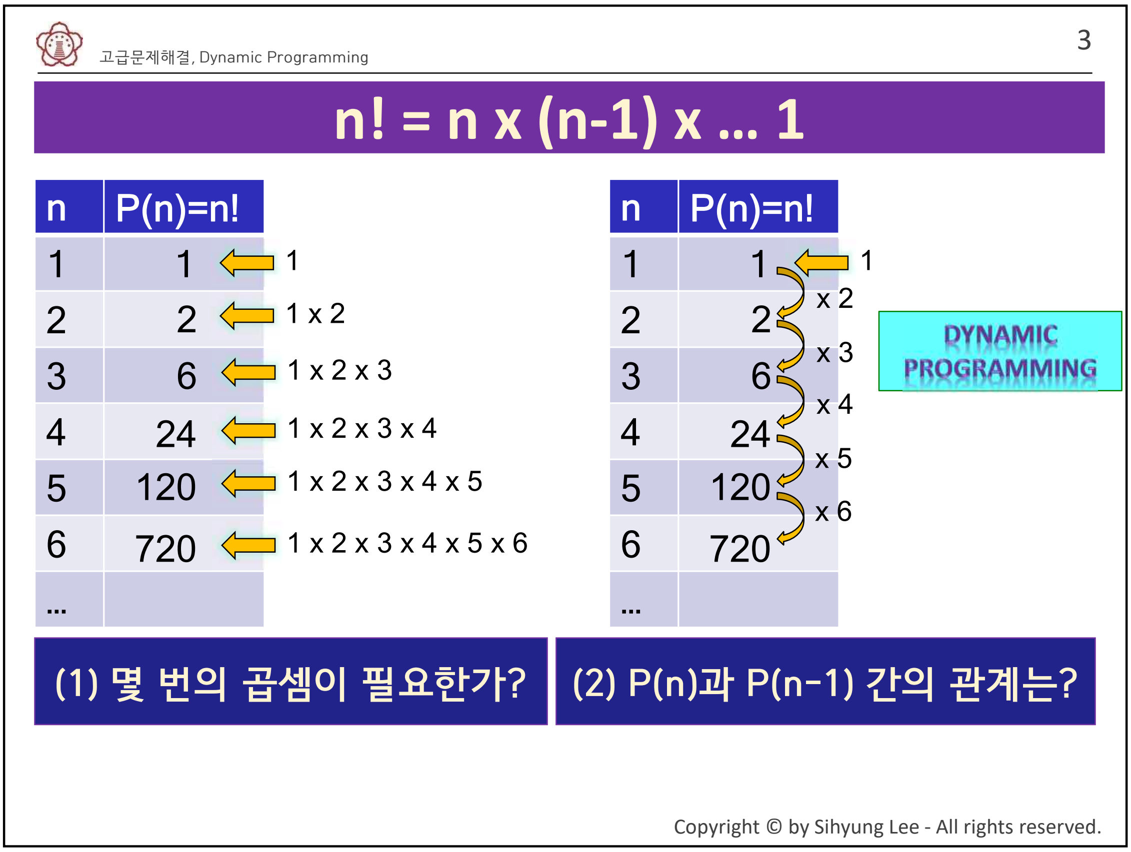


'Dynamic programming'은 위 오른쪽 그림과 같이 작은 문제에 대한 답을 기억해 두었다가 더 큰 문제를 풀기 위해 재사용함으로써 문제를 보다 효율적으로 풀어내는 알고리즘을 통칭합니다. 앞으로 dynamic programming을 줄여서 DP라 하겠습니다.

DP는 보통 하나의 문제를 풀어야 하는 경우보다는 **여러 다른 크기의 문제를 함께 풀어야 하는 경우에 적용**됩니다 (예: 크기 1~N까지의 문제를 모두 풀어야 하는 경우). 이러한 경우 작은 문제와 큰 문제 간의 연관성을 파악할 수 있다면 작은 문제에 대한 해를 재사용함으로써 다음으로 풀어야 할 보다 큰 문제를 더 빠르게 풀어낼 수 있습니다. 예를 들어 크기 1인 문제를 풀어 이에 대한 해 P(1)을 얻었다면 이를 사용해 P(2)를 빨리 계산해 냅니다. 이번에는 P(2)를 사용해 다음으로 풀어야 할 P(3)을 빨리 계산해 냅니다.

만약 같은 상황에서 DP를 사용하지 않는다면 위 왼쪽 그림과 같이 기존 문제의 해를 재사용하지 않고 각 문제를 독립적으로 풀어야 하므로 시간이 더 걸리게 됩니다.

DP를 적용하기 위해서는 서로 다른 크기의 문제 간의 연관성을 잘 발견할 수 있어야 합니다. 그래서 앞으로는 몇 가지 문제에 대해 이러한 연관성의 예를 보도록 하겠습니다.

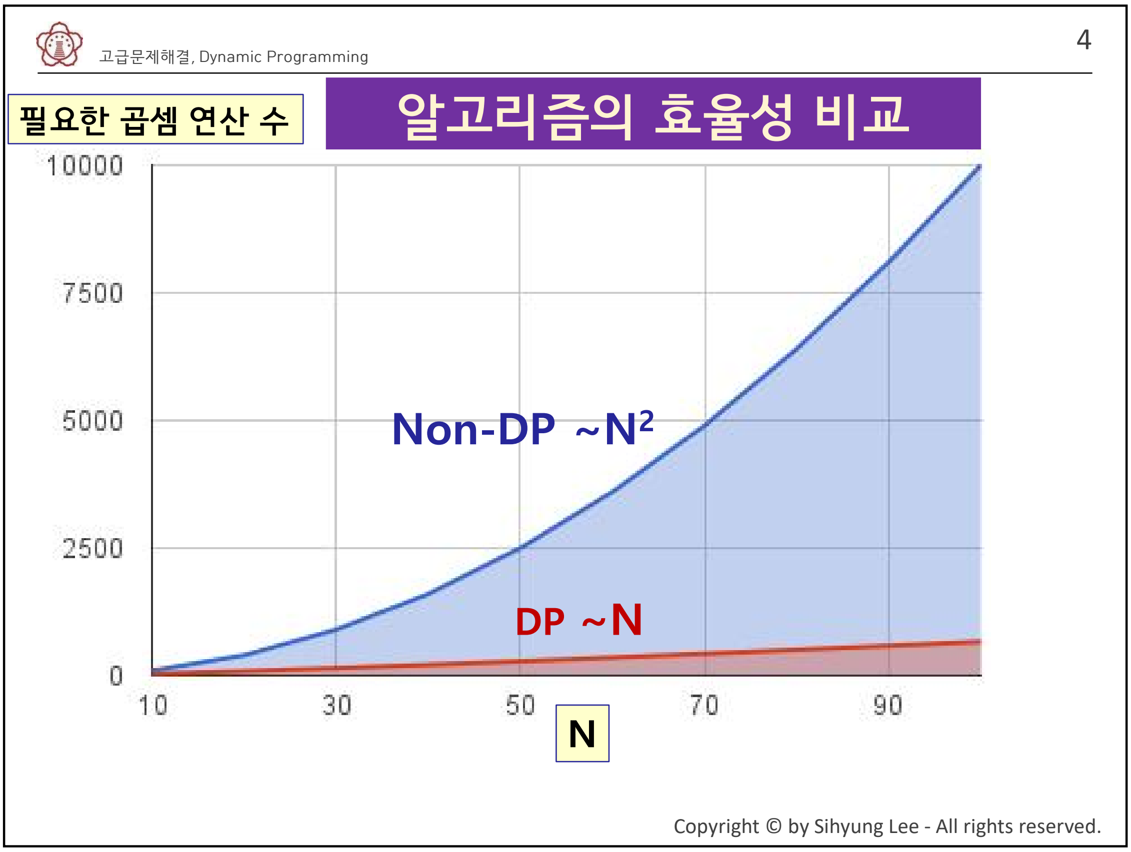


DP가 도움이 되는 첫 번째 예로 n-factorial 계산을 보겠습니다. 특히 n=1~N 까지 N개의 n-factorial을 구해야 한다고 가정하겠습니다.

(1) DP를 사용하지 않는 경우(위 왼쪽 그림): 각 factorial을 독립적으로 계산합니다. 각 n-factorial을 구하기 위해서는 n-1번의 곱셈이 필요합니다. 예를 들어 2! = 1×2를 계산하려면 (2-1)=1번의 곱셈이 필요하고 3!=1×2×3을 계산하려면 (3-1)=2번의 곱셈이 필요합니다. 따라서 n=1~N까지 N개의 factorial을 구하기 위해서는 0+1+2+⋯+(N-1)=(N-1)×N/2 번의 곱셈이 필요하며 이는 N^2에 비례합니다.

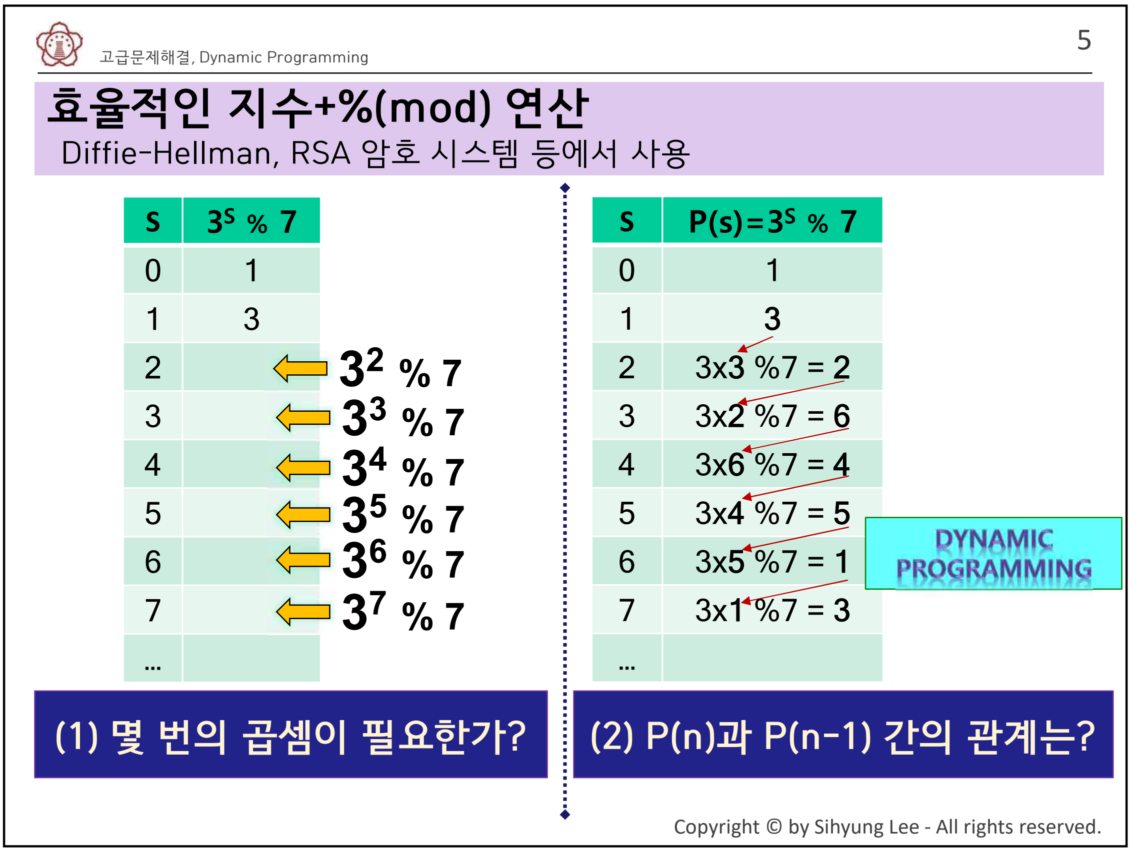
앞으로 P(n)을 크기 n인 문제에 대해 우리가 구하고자 하는 답이라 가정하겠습니다. 따라서 factorial 문제에서는 P(n)=n! 입니다.

(2) DP를 사용하는 경우(위 오른쪽 그림): P(n)과 P(n-1) 간의 관계, 즉 P(n)=n×P(n-1) (예: P(4) = 4×3×2×1 = 4×P(3))를 활용합니다. 이 관계를 활용하면 P(n-1)에 n을 곱해 P(n)을 얻을 수 있으므로 각 P(n)을 얻는데 1번의 곱셈만 필요합니다. 단 초기값인 P(1)=1을 구할 때는 곱셈이 필요하지 않습니다. 따라서 n=1~N까지 N개의 factorial을 구하기 위해서는 N-1번의 곱셈이 필요하며 이는 N에 비례합니다.



앞 페이지에서 n-factorial을 계산할 때 DP를 사용하지 않는다면 N^2에 비례한 수의 곱셈이 필요하며 DP를 사용한다면 N에 비례한 수의 곱셈이 필요함을 보았습니다. 이 두 값은 위 그래프에서와 같이 N이 증가할수록 그 차이가 더 커집니다.

따라서 풀어야 할 문제의 크기가 커질수록 처리 속도를 높이기 위해서는 효율적인 알고리즘을 찾는 것이 중요함을 알 수 있습니다.



다음으로 factorial 계산과 유사한 문제를 하나 더 보겠습니다. 특히 현대 암호 시스템에서 자주 사용되는 지수+mod 연산이 사용되는 경우를 보겠습니다. mod 연산은 주어진 수로 나누었을 때 나머지를 구하는 연산으로 %로 자주 표기합니다.

S=1~N에 대해 3^s%7을 계산해야 한다고 가정하겠습니다.

이번에도 DP를 사용하지 않는 경우와 DP를 사용하는 경우에 곱셈 연산의 수를 비교해 보겠습니다.

(1) DP를 사용하지 않는 경우(위 왼쪽 그림): s=1~N까지의 각 s에 대해 거듭제곱 3^s (3의 s승)를 계산한 후 mod 연산을 수행합니다. 3^s을 구하기 위해서는 s-1번의 곱셈이 필요합니다. 예를 들어 3^2 = 3×3을 계산하려면 (2-1)=1번의 곱셈이 필요하고 3^3 = 3×3×3을 계산하려면 (3-1)=2번의 곱셈이 필요합니다. 따라서 n=1~N까지 N개의 factorial을 구하기 위해서는 0+1+2+⋯+(N-1)=(N-1)×N/2 번의 곱셈이 필요하며 이는 **N^2에 비례**합니다.

(2) DP를 사용하는 경우(위 오른쪽 그림): P(n)과 P(n-1) 간의 관계, 즉 **P(n)=3×P(n-1) %7**을 활용합니다. 예를 들어 **P(2)** = 3^2 % 7 = (3×3) % 7 = (3%7)×(3%7) %7 = **3×P(1) % 7** 입니다. 이 관계를 활용하면 P(n-1)에 3을 곱하고 %7 하면 바로 P(n)을 얻을 수 있으므로 각 P(n)을 얻는데 1번의 곱셈만 필요합니다. 단 초기값인 P(1)=3을 구할 때는 곱셈이 필요하지 않습니다. 따라서 s=1~N까지 N개의 3^s%7을 구하기 위해서는 N-1번의 곱셈이 필요하며 이는 N에 비례합니다.

정리하면 이번에도 DP를 사용하면 DP를 사용하지 않는 경우보다 더 빠르게 계산을 완료할 수 있음을 볼 수 있습니다.

※ 위 (2)의 예제에 사용된 아래 등식은 두 양의 정수 a와 b의 곱에 대해 일반적으로 성립하는 등식입니다. a=cp+q로 치환하고 b=cp'+q'로 치환하면 증명할 수 있습니다.

(a×b) % c = (a%c) × (b%c) % c

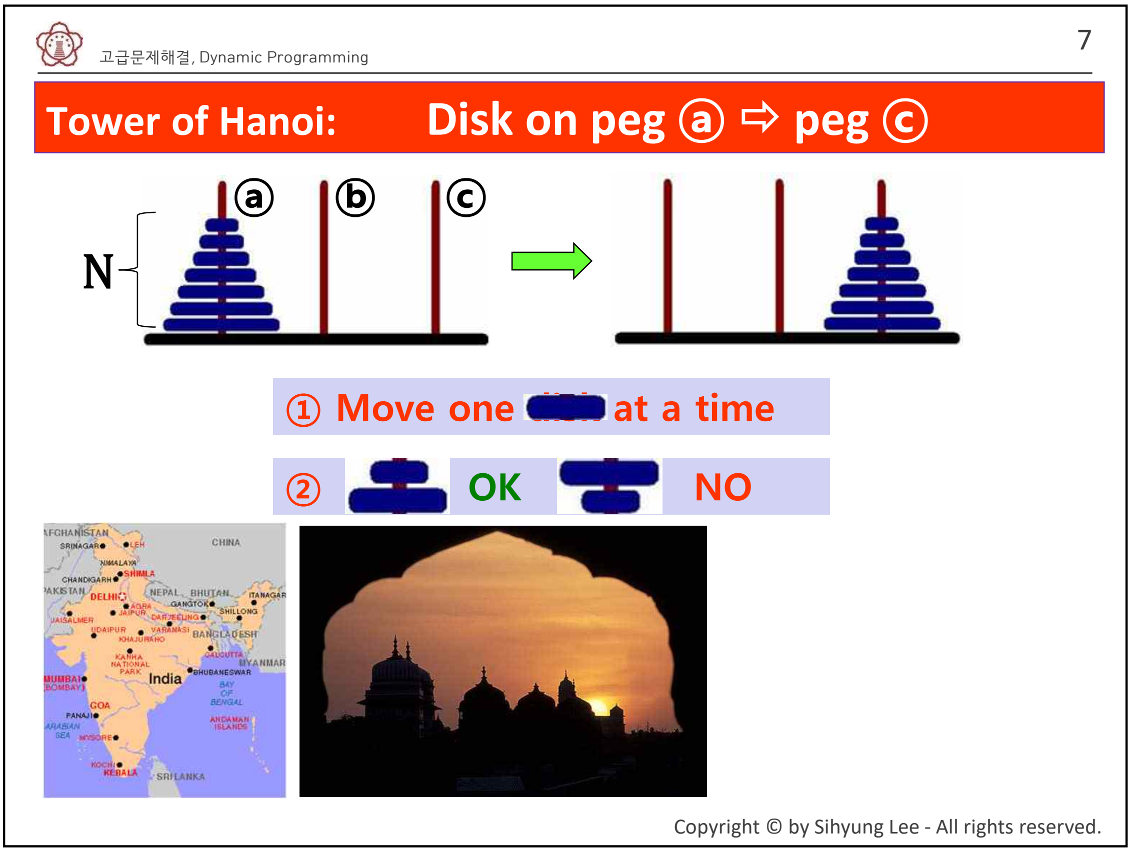
※ 위 예에서는 (1) DP를 사용하지 않는 경우와 (2) DP를 사용한 경우에 대해 **곱셈 연산의 수만을 비교**하였고 **mod 연산의 수는 비교하지 않았습**니다. 곱셈 연산은 (1)과 (2)에서 수행하는 횟수가 다르므로 비교의 의미가 있지만 mod 연산은 (1)과 (2)에서 같은 횟수만큼 수행하므로(각 s 값에 대해 %7을 한 번씩 동일하게 수행) 비교하지 않았습니다.

[Q] s=1~N에 대해 P(s) = 5^s % 7를 계산해야 한다. P(1)과 P(2)를 구했다고 가정하자. P(3)을 구하는 방법 중 정확한 답을 내면서 가장 효율적인 방법은?

- 5 x P(2)
- (5 x P(2)) % 7
- 5^3 % 7
- 5^3

이 자료에서 [Q]로 제시된 문제는 학습 후 풀이하는 온라인 퀴즈에 그대로 나오니 학습하면서 그때그때 문제를 미리 풀어 두세요.

온라인 퀴즈에서 보기의 순서는 바뀔 수 있으니 유의하세요. (예: 보기 1이 보기 2가 되고, 보기 2가 보기 1이 될 수도 있음)



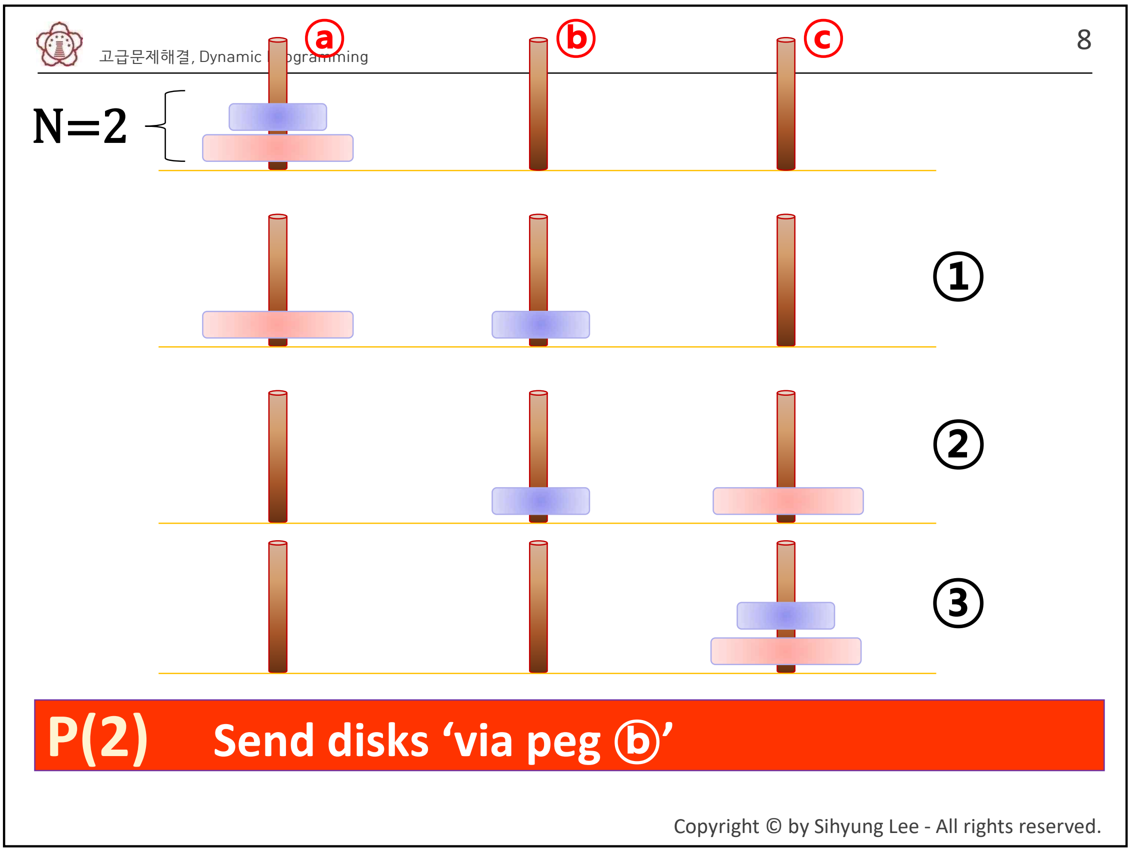
이번에 볼 DP 문제는 '하노이의 탑' 문제입니다.

이 문제의 목표는 왼쪽의 막대 ⓐ에 있는 N개의 디스크를 오른쪽의 막대 ⓒ로 모두 옮기는 것입니다.

각 디스크는 크기가 서로 다른데, 이들을 옮기는 과정 중에는 다음 **규칙 ①~②를 반드시 지켜야 합니다**:

① 한 번에 한 디스크만 옮길 수 있으며 가운데의 막대 ⓑ로 옮기는 것도 가능함
② 작은 디스크는 큰 디스크 위에 올 수 있지만 큰 디스크는 작은 디스크 위에 올 수 없음

하노이의 탑 문제는 인도의 사원에서 N=64개의 디스크를 사용해 실제로 수행되고 있다고 하며, 이들을 모두 옮길 때 세상의 종말이 온다는 전설이 있습니다.



하노이의 탑 문제에 DP를 적용하게 위해 작은 문제(디스크 수 N 값이 작은 문제)와 큰 문제(N 값이 큰 문제) 간의 연관성을 찾아보겠습니다.

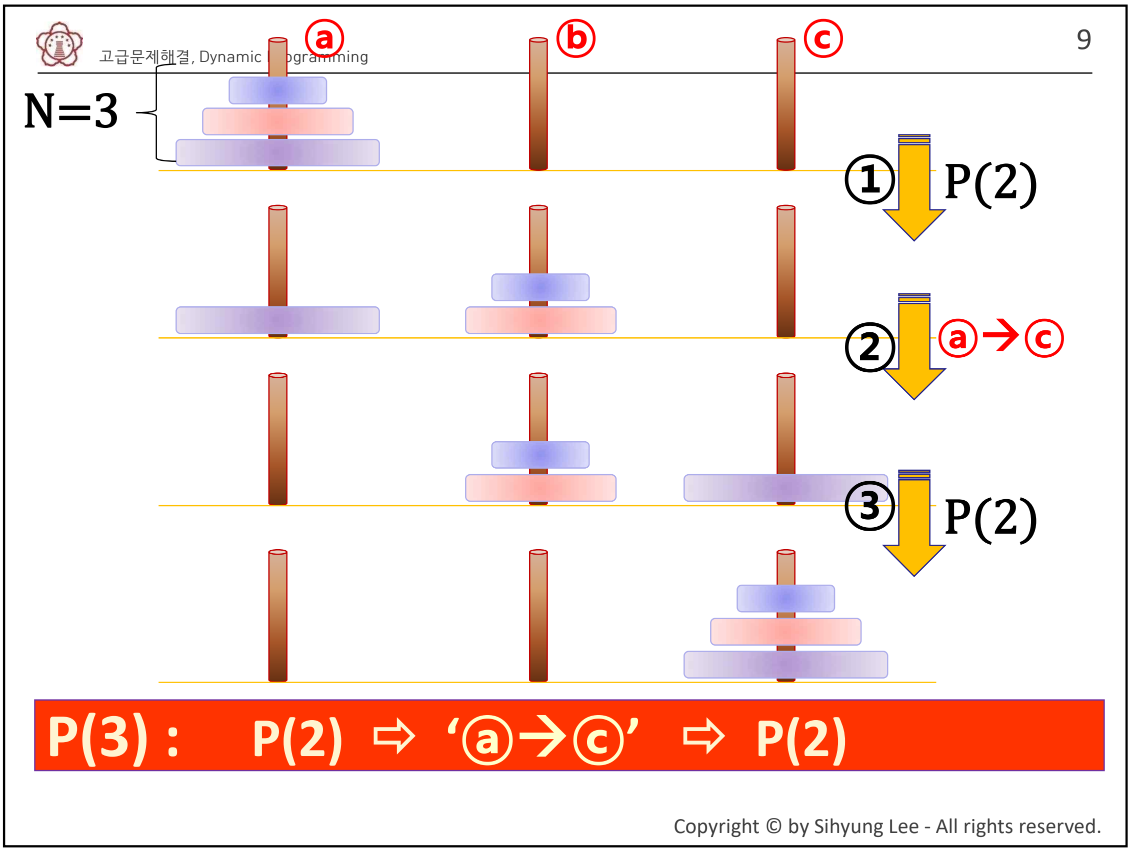
N=2인 간단한 경우(디스크가 2개인 경우)에서 분석을 시작해 보겠습니다. 큰 디스크(그림에서 붉은 색 디스크)가 최종적으로 막대 ⓒ 아래에 와야 하므로

① 먼저 작은 디스크(그림에서 푸른 색 디스크)를 가운데 막대 ⓑ로 옮기고

② 큰 디스크를 막대 ⓒ로 옮긴 후

③ 마지막으로 작은 디스크를 막대 ⓑ에서 막대 ⓒ로 옮기면 됩니다.

여기서 중요한 부분은 '**규칙에 따라 큰 디스크를 ⓐ에서 ⓒ로 옮기기 위해서 작은 디스크를 임시로 두기 위한 경유지가 필요**'했으며, '**가운데 막대 ⓑ를 이러한 경유지로 활용**'했다는 것입니다.



이제 N을 1 증가시킨 N=3인 경우(디스크가 3개인 경우)를 보겠습니다. 이 경우도 해법은 N=2인 경우와 유사합니다.

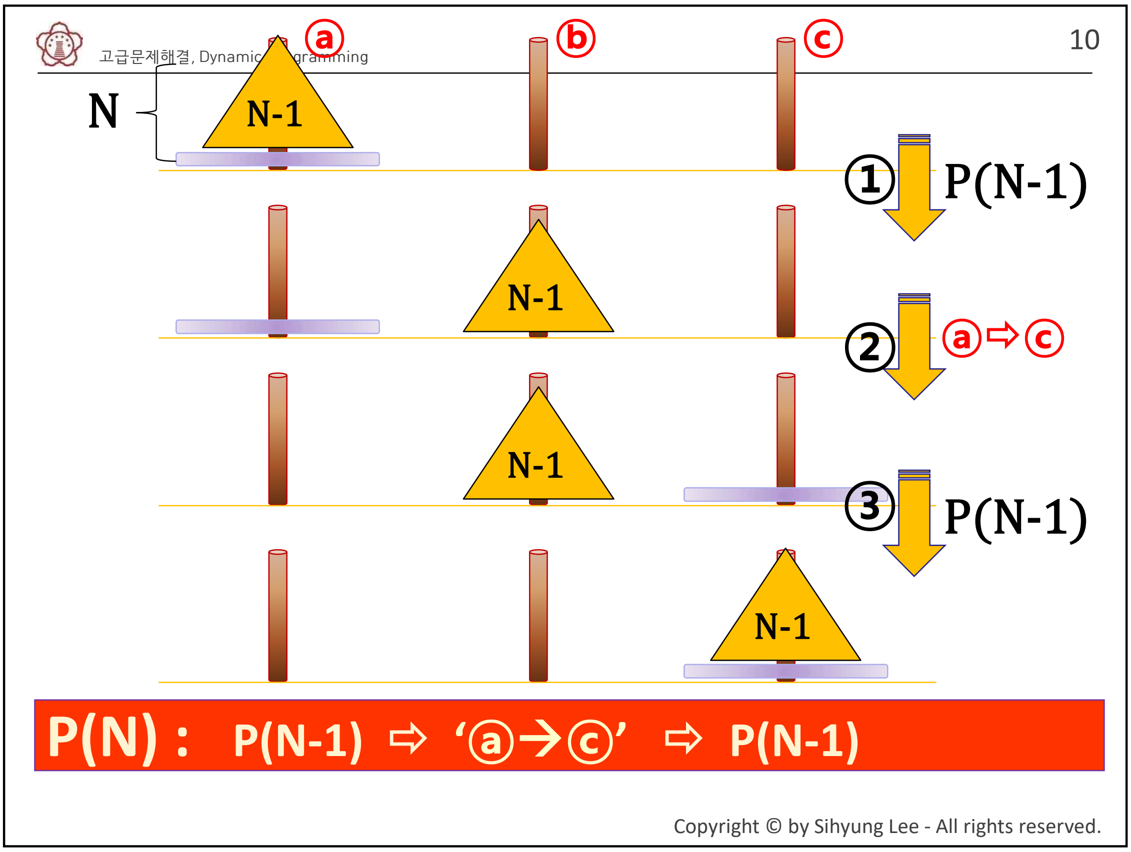
가장 큰 디스크가 최종적으로 막대 ⓒ 아래에 와야 하므로

① 먼저 2개의 작은 디스크를 가운데 막대 ⓑ로 옮기고
② 가장 큰 디스크를 막대 ⓒ로 옮긴 후
③ 2개의 작은 디스크를 막대 ⓑ에서 막대 ⓒ로 옮기며 가장 큰 디스크 위에 쌓으면 됩니다.

위의 단계 ①을 수행하기 위해서는 2개의 작은 디스크를 막대 ⓐ에서 ⓑ로 옮겨야 합니다. 이를 위해 N=2일 때의 해법 P(2)를 사용하되 막대 ⓒ를 경유지로 사용하면 됩니다.

위의 단계 ③을 수행하기 위해서는 2개의 작은 디스크를 막대 ⓑ에서 ⓒ로 옮겨야 합니다. 이를 위해 N=2일 때의 해법 P(2)를 사용하되 막대 ⓐ를 경유지로 사용하면 됩니다.

정리하면 N=3인 경우의 해법 P(3)에서는 ① 먼저 P(2)를 사용하고, ② 가장 큰 디스크를 ⓐ에서 ⓒ로 옮긴 후, ③ P(2)를 한 번 더 사용하면 됩니다.



지금까지 본 해법을 임의의 N 값에 대해 일반화 해 보겠습니다.

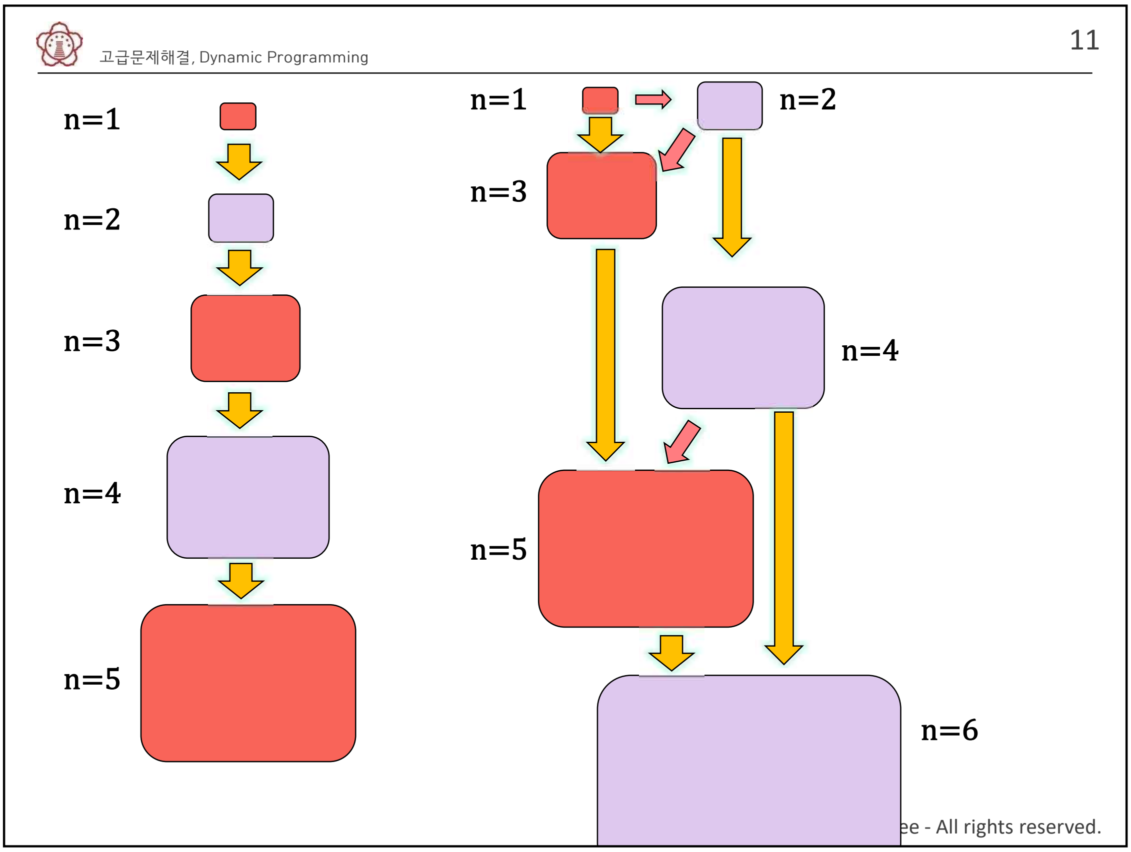
가장 큰 디스크가 최종적으로 막대 ⓒ 아래에 와야 하므로

① 먼저 N-1개의 작은 디스크를 가운데 막대 ⓑ로 옮기는데, 이 때 N-1개 디스크에 대한 해법 P(N-1)을 활용하고

② 가장 큰 디스크를 막대 ⓒ로 옮긴 후

③ N-1개의 작은 디스크를 막대 ⓑ에서 막대 ⓒ로 옮기기 위해 N-1개 디스크에 대한 해법 P(N-1)을 다시 활용합니다.

정리하면 작은 문제와 큰 문제 간의 연관성을 발견하고 이를 활용한다면 여러 다른 크기의 문제를 보다 간단하고 체계적으로 풀어낼 수 있습니다.



지금까지는 P(N)을 구하기 위해 직전 문제에 대한 해법 P(N-1)을 활용하는 예를 보았습니다.

하지만 DP를 적용할 때는 반드시 P(N-1)만을 활용해야 하는 것은 아니며, 필요에 따라 임의의 P(N-i), i>=1,도 활용 가능합니다. 즉 **직전 문제보다 더 이전 문제에 대한 해법을 활용해 P(N)을 구할 수도 있습니다**. 예를 들어 P(10)을 구하기 위해 P(8)이나 P(4) 등을 활용할 수도 있습니다.

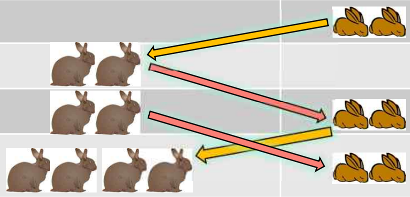
또한 이에 더해서 반드시 "하나의" 이전 문제에 대한 해법 만을 활용해야 하는 것도 아닙니다. **둘 이상의 서로 다른 이전 문제에 대한 해법을 활용할 수도 있습니다**. 예를 들어 P(N-i)와 P(N-j), i≠j,를 함께 활용해 P(N)을 구할 수도 있습니다.

이러한 경우의 예를 보겠습니다.

How many rabbit pairs do we have on month N (P(n))?

"A baby rabbit pair grows into adults in one month,
then they give birth to a pair of (female and male) rabbits every month"

Month M	Adult Rabbit Pairs (Can produce babies)	Baby Rabbit Pairs	Total # of Pairs P(M)
0			1
1			1
2			2
3			3
4			
5			
...			



생물학자들은 토끼 개체 수의 증가를 모델링 하기 위해 다음과 같은 가설을 세웠습니다:

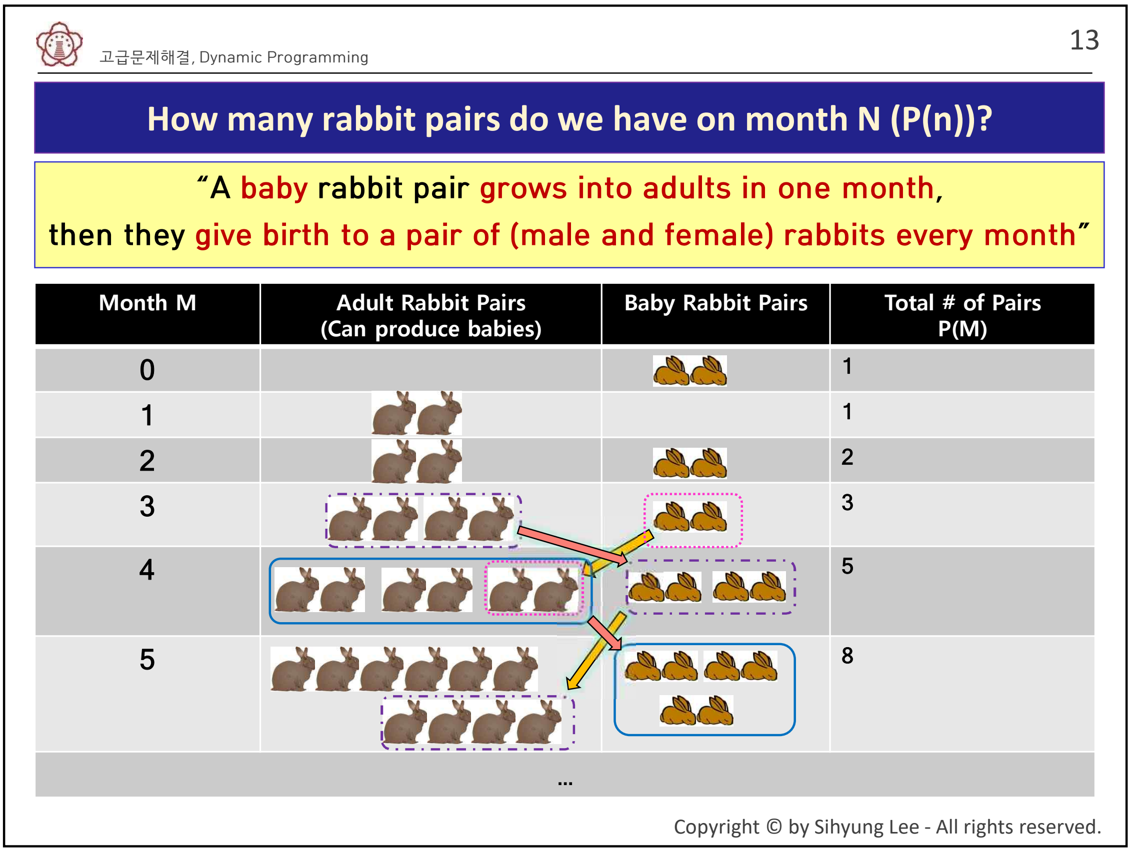
- 막 태어난 아기 토끼는 1달 후에 성체가 되며
- 성체가 된 토끼 한 쌍은 매달 한 쌍의 아기 토끼를 낳는다(암수 각 하나씩).

위와 같은 가설에 따라 토끼 개체 수가 어떻게 증가하는지 보겠습니다. 처음(month #0)에는 위 표와 같이 막 태어난 한 쌍의 아기 토끼가 암수 각 하나씩 있다고 가정하겠습니다.

Month #0→1: 아기 토끼 한 쌍이 성체 한 쌍으로 자라납니다. 성체로 자라날 때까지는 새로운 아기 토끼를 낳을 수 없으므로 month #1에는 아기 토끼가 없습니다.

Month #1→2: 성체가 된 토끼 한 쌍이 아기 토끼 한 쌍을 낳습니다. 따라서 month #2에는 성체 한 쌍 + 아기 한 쌍 = 총 두 쌍의 토끼가 존재합니다.

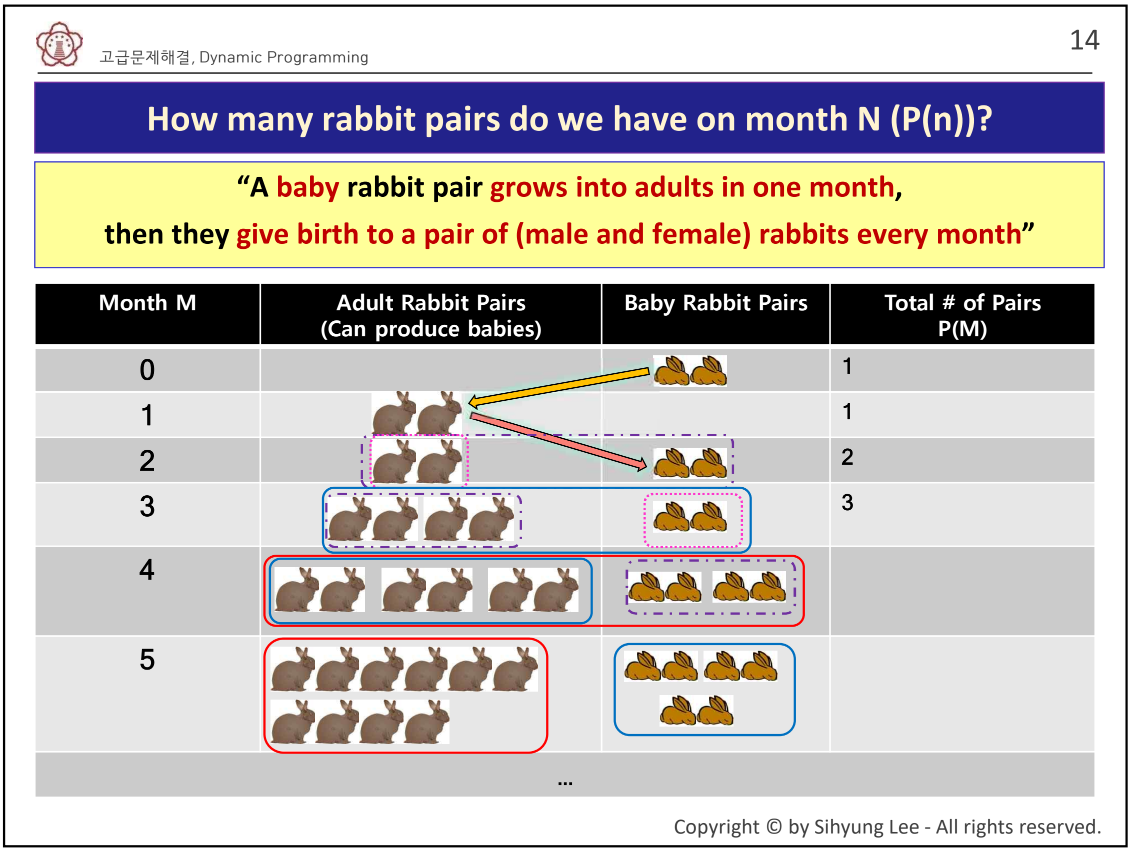
Month #2→3: 성체 토끼 한 쌍이 새로운 아기 토끼 한 쌍을 낳습니다. month #2에 새로 태어났던 아기 한 쌍은 month #3에 성체로 자라납니다. 따라서 month #3에는 성체 두 쌍 + 아기 한 쌍 = 총 세 쌍의 토끼가 존재합니다.



Month #3→4: 성체 토끼 두 쌍이 각각 아기 토끼 한 쌍 씩 낳아 총 두 쌍의 아기 토끼가 새로 태어납니다. month #3에 새로 태어났던 아기 한 쌍은 month #4에 성체로 자라납니다. 따라서 month #4에는 성체 세 쌍 + 아기 두 쌍 = 총 다섯 쌍의 토끼가 존재합니다.

Month #4→5: 성체 토끼 세 쌍이 각각 아기 토끼 한 쌍 씩 낳아 총 세 쌍의 아기 토끼가 새로 태어납니다. month #4에 새로 태어났던 아기 두 쌍은 month #5에 성체로 자라납니다. 따라서 month #5에는 성체 5쌍 + 아기 3쌍 = 총 8쌍의 토끼가 존재합니다.

지금까지 본 예제에 따르면 P(M) (month M에 존재하는 총 토끼 쌍 수)과 이전 문제에 대한 해 P(M-i)들 간에는 어떤 관계가 있을까요?



P(M)과 P(M-i) 간의 관계를 아래 ①~②와 같이 나누어 생각해 보겠습니다.

① **month M에서 <u>성체인</u> 토끼 쌍의 수**는 직전 달인 month M-1에서 성체인 토끼 쌍의 수 + 아기 토끼 쌍의 수입니다. 즉 month M에서 성체인 토끼 쌍의 수는 **직전 달의 총 토끼 쌍 수인 P(M-1)** 입니다. 직전 달 M-1에서 성체인 토끼는 이번 달 M에도 그대로 성체로 남아 있으며, 직전 달 M-1에서 아기인 토끼는 이번 달 M에 성체로 자라나기 때문입니다.

② **month M에서 <u>아기</u>인 토끼 쌍의 수**는 직전 달인 month M-1에서 성체였던 토끼 쌍의 수와 같습니다. 직전 달 M-1에서 성체였던 각 토끼 쌍이 이번 달 M이 되면서 아기 토끼 한 쌍 씩을 낳기 때문입니다. 직전 달 M-1에서 성체였던 토끼 쌍의 수는 ①에서 확인한 관계를 적용하면 **그 전 달 M-2에서 총 토끼 쌍 수인 P(M-2)** 입니다.

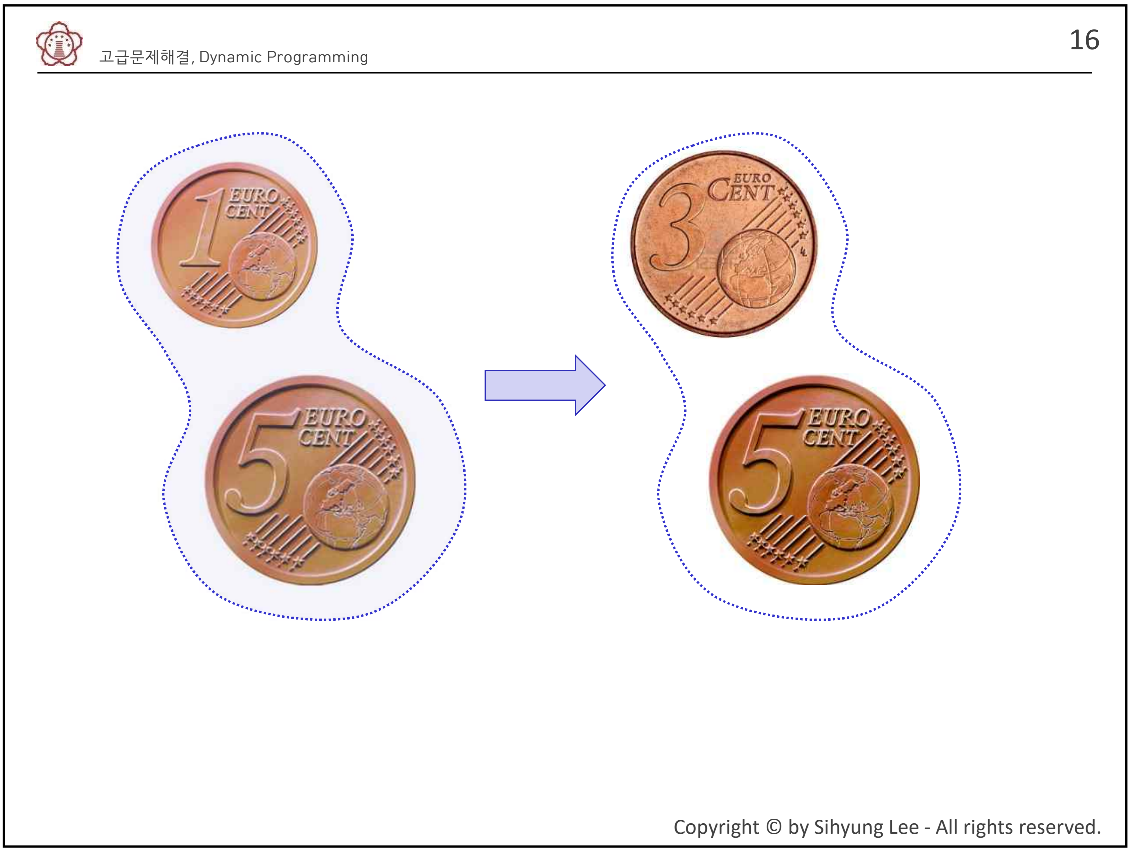
위 ①~②를 종합하면 month M에서 총 토끼 쌍 수 P(M) = ① + ② = P(M-1) + P(M-2) 가 됩니다. 이 관계는 수학 교재에 종종 나오는 관계로 피보나치 수열(Fibonacci sequence)이라고 합니다. 피보나치 수열은 토끼 객체 수 뿐 아니라 다른 생물학적 현상을 모델링하는데도 사용 되고 있습니다.

[Q] 토끼 개체 수를 구하는 문제를 생각해 보자. 아래와 같은 숫자를 관찰했다면 month i+2에 _아기_ 토끼는 몇 쌍인가?

- P(i) = 4181
- P(i+1) = 6765
- P(i+2) = 10946
- P(i+3) = 17711

이 자료에서 [Q]로 제시된 문제는 학습 후 풀이하는 온라인 퀴즈에 그대로 나오니 학습하면서 그때그때 문제를 미리 풀어 두세요.

온라인 퀴즈에서 보기의 순서는 바뀔 수 있으니 유의하세요. (예: 보기 1이 보기 2가 되고, 보기 2가 보기 1이 될 수도 있음)



P(N)이 직전 해인 P(N-1) 뿐만 아니라 그 전의 해인 P(N-i), i>1도 활용하는 또다른 예를 보겠습니다.

1센트와 5센트 두 종류의 동전이 사용되고 있다고 가정하겠습니다. 물가가 상승하여 1센트로 구매할 수 있는 물건이 거의 없어지자 1센트 동전을 3센트 동전으로 대체하는 방안이 제안되었습니다. 즉 3센트와 5센트 두 종류의 동전을 사용하는 계획입니다.

이 계획을 실행하기에 앞서 먼저 **어떤 가격의 물건이라도 3센트와 5센트 두 종류의 동전을 조합**하면 거스름돈 없이 **구매할 수 있음을 확인**하고 싶습니다. 예를 들어 9센트의 물건은 3센트 동전 3개로 구매할 수 있으며 (3×3=9), 10센트의 물건은 5센트 동전 2개로 구매할 수 있습니다 (5×2=10).

하지만 모든 가격에 대해서 이와 같이 지불이 가능할까요? 예를 들어 101센트, 277센트, 599 센트 등도 3센트와 5센트 동전 만으로 정확히 지불 가능할까요?

Can we pay all prices ≥8 with combinations of 3-cent and 5-cent coins?

N(Price to pay)	P(N) (Coin combinations)
8	3 5
9	3 3 3
10	
11	
12	
13	
14	
15	
…	

(1) Complete the table for N=10~12

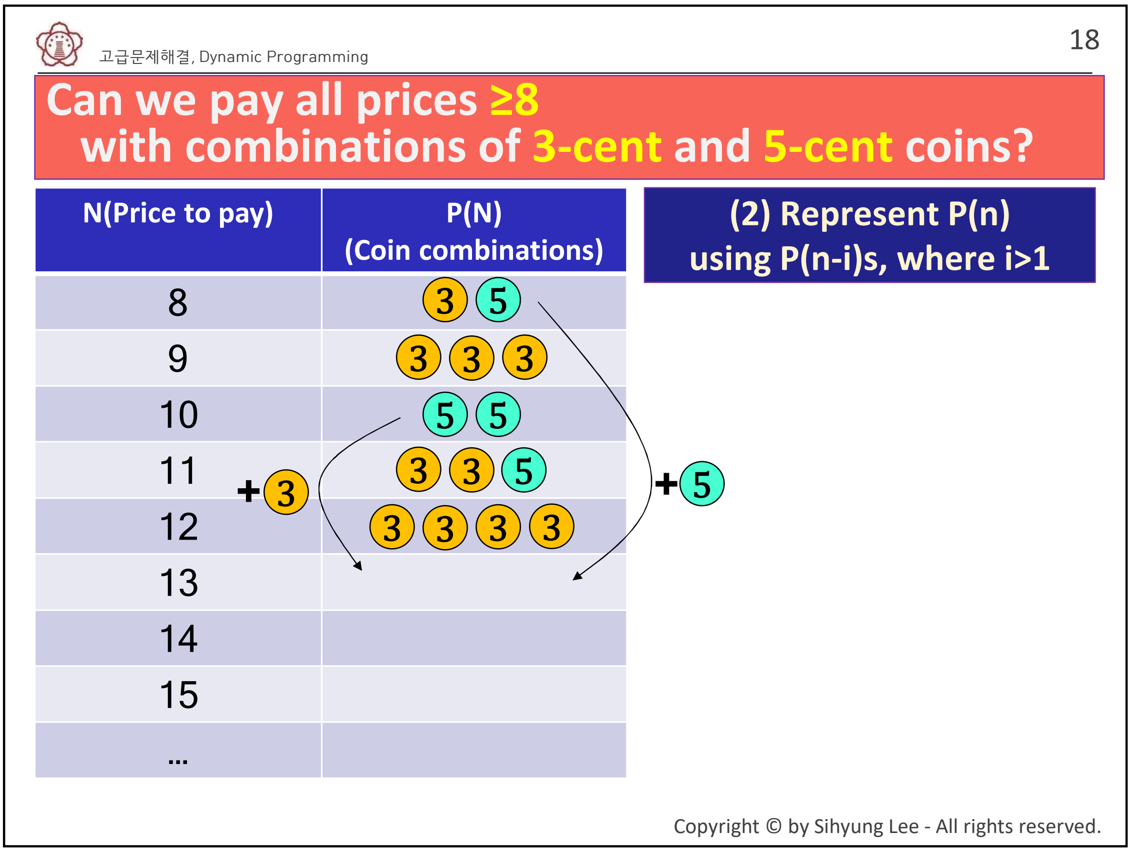
앞 문제에 대한 답을 알아보기에 앞서 한 가지 가정을 하겠습니다. **모든 상품은 8센트 혹은 그보다 높은 가격**에 판매되며, 8센트보다 낮은 가격의 제품은 없다고 하겠습니다.

우리가 풀어야 할 문제를 다시 정리하면 다음과 같습니다:

P(N)과 P(N-i)간의 관계를 찾고, 이를 활용해 8센트 혹은 그보다 높은 모든 가격이 3센트와 5센트만으로 지불 가능함을 확인

P(N)은 N센트를 지불하기 위한 3센트와 5센트 동전의 조합입니다. 예를 들어 P(8)={3,5} 이며, 8센트는 3센트 동전 하나와 5센트 동전 하나로 지불할 수 있다는 뜻입니다. P(9)={3,3,3} 이며, 9센트는 3센트 동전 3개로 지불할 수 있다는 뜻입니다.

먼저 위 표에서 N=10~12에 대해 P(N)을 구해 보세요. P(10)~P(12)를 구하면서 P(N)과 이전 문제의 해 P(N-i)들 간의 관계에 대해서도 생각해 보세요.



이제 P(N)과 P(N-i), i>1, 간의 관계를 찾아 보겠습니다.

우리가 사용할 수 있는 동전은 3센트와 5센트 두 종류이므로, **P(N)=P(N-3)+{3}** 혹은 **P(N)=P(N-5)+{5}** 두 가지 방법 중 하나로 P(N)을 구할 수 있습니다. 예를 들어 P(13)은 P(10)+{3}={5,5}+{3}={3,5,5}로 구할 수 있으며 또는 P(8)+{5}={3,5}+{5} = {3,5,5}로도 구할 수 있습니다. 두 가지 중 어느 방법을 사용하더라도 결국 P(13)={3,5,5}를 얻을 수 있습니다.

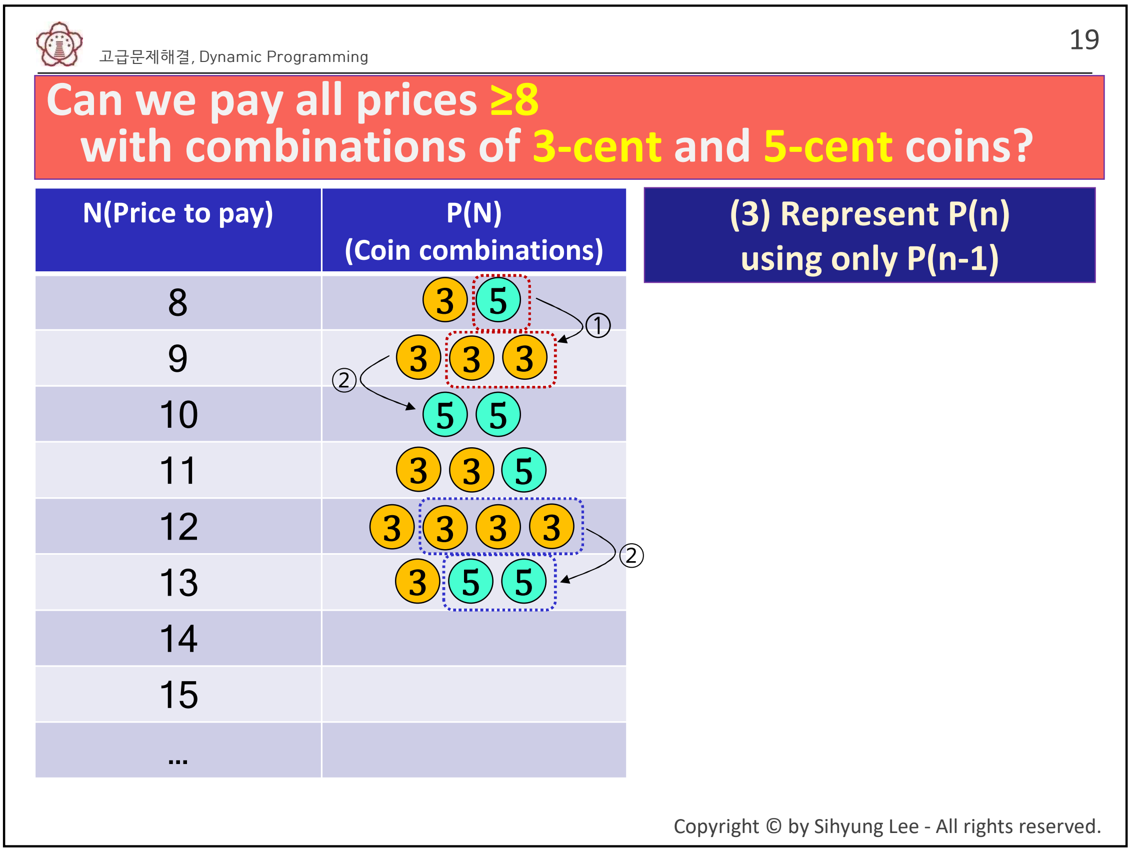
이와 유사하게

P(14) = P(11) + {3} = P(9) + {5} = {3,3,3,5} 이며

P(15) = P(12) + {3} = {3,3,3,3,3} 또는 P(15) = P(10) + {5} = {5,5,5} 입니다.

P(15)에 대해서는 두 가지의 다른 해를 찾을 수 있는데, 이는 15센트를 두 가지의 서로 다른 동전의 조합 중 하나로 지불할 수 있다는 뜻입니다.

정리하면 P(N)=P(N-3)+{3} 과 P(N)=P(N-5)+{5}이라는 관계를 찾음으로써 임의의 N에 대해 P(N)을 보다 체계적으로 구할 수 있게 되었습니다. 또한 8센트 혹은 그보다 높은 모든 가격에 대해 3센트와 5센트의 조합으로 지불 가능함도 확인할 수 있게 되었습니다.



이번에는 P(N)과 P(N-1) 간의 관계를 찾아 보겠습니다. 즉 바로 직전 문제의 해만을 사용하여 P(N)을 구하는 방법을 생각해 보겠습니다. 이 방법은 P(N-3)이나 P(N-5)를 사용하는 방법보다는 조금 더 복잡합니다. 아래와 같이 ①~② 두 가지 경우로 나누어 생각해 보겠습니다.

① **P(N-1)이 5센트 동전을 하나 이상 포함하는 경우**입니다. 이 경우에는 **5센트 하나를 3센트 2개로 대체**함으로써 P(N)을 구할 수 있습니다. 예를 들어 P(8)={3,5}은 5센트 동전 하나를 포함하며, 이를 3센트 2개로 대체함으로써 P(9)={3,3,3}를 구할 수 있습니다. 위 그림의 화살표 ①을 참조하세요. 마찬가지로 P(10)={5,5}는 5센트 동전 하나 이상을 포함하며, 이 중 하나를 3센트 2개로 대체함으로써 P(11)={3,3,5}을 구할 수 있습니다.

하지만 P(N-1)이 항상 5센트 동전을 포함하지는 않으며 이러한 경우는 ①의 방법을 적용할 수 없습니다. 예를 들어 P(9)={3,3,3}에는 5센트 동전이 없으며, P(12)={3,3,3,3}에도 5센트 동전은 없습니다. 이러한 경우 아래의 ②와 같이 P(N)을 구합니다.

② **P(N-1)이 5센트 동전을 포함하지 않는 경우**입니다. 즉 P(N-1)이 3센트 동전만으로 구성되어 있는 경우입니다. 이 경우는 반드시 **3센트 동전이 3개 이상 있다**고 볼 수 있습니다. 이에 해당하는 경우 중 N이 가장 작은 경우가 P(9)={3,3,3}이며, 다음으로 작은 경우가 P(12)={3,3,3,3} 입니다. 따라서 이보다 큰 N에 대해서 5센트 동전을 볼 수 없다면 반드시 3센트 동전이 3개 이상 있다고 볼 수 있습니다.

②의 경우에는 **3센트 3개를 5센트 2개로 대체**함으로써 P(N)을 구할 수 있습니다. 예를 들어 P(9)={3,3,3}는 3센트 3개를 포함하며, 이들을 5센트 2개로 대체함으로써 P(10)={5,5}을 구할 수 있습니다. 마찬가지로 P(12)={3,3,3,3}는 3센트 4개를 포함하며 이 중 3개를 5센트 2개로 대체함으로써 P(13) = {3,5,5}을 구할 수 있습니다.

정리하면 P(N)은 직전 문제 P(N-1)과 관계가 있을 수도 있고 혹은 그 전의 문제와 관계가 있을 수도 있습니다. 어떤 관계를 사용하는 것이 더 간단한지는 문제에 따라 다릅니다. 따라서 늘 여러 가능한 경우를 고려해 보는 것이 좋습니다.

[Q] 3센트/5센트 동전 문제를 생각해 보자. P(8)은 {3,5} 한 가지 조합으로만 표현될 수 있지만 P(15)는 {3,3,3,3,3}과 {5,5,5}의 서로 다른 두 가지 조합으로 표현될 수 있다. 아래 중 서로 다른 두 가지 조합으로 표현될 수 있는 것은?

- P(14)
- P(16)
- P(17)
- P(18)

이 자료에서 [Q]로 제시된 문제는 학습 후 풀이하는 온라인 퀴즈에 그대로 나오니 학습하면서 그때그때 문제를 미리 풀어 두세요.

온라인 퀴즈에서 보기의 순서는 바뀔 수 있으니 유의하세요. (예: 보기 1이 보기 2가 되고, 보기 2가 보기 1이 될 수도 있음)

[Q] Dynamic programming에 대한 설명으로 잘못된 것은?

- 이미 구한 작은 문제의 해를 활용해 더 큰 문제의 해를 효율적으로 구함
- DP를 적용하기 위해서는 서로 다른 크기의 문제들 간 관계를 잘 파악해야 함
- DP를 사용하면 DP를 사용하지 않는 경우에 비해 연산 횟수를 줄일 수 있음
- P(N)은 직전 문제의 해인 P(N-1)만을 활용할 수 있으며, 그 전 문제의 해인 P(N-i), i>1,은 활용할 수 없음

이 자료에서 [Q]로 제시된 문제는 학습 후 풀이하는 온라인 퀴즈에 그대로 나오니 학습하면서 그때그때 문제를 미리 풀어 두세요.

온라인 퀴즈에서 보기의 순서는 바뀔 수 있으니 유의하세요. (예: 보기 1이 보기 2가 되고, 보기 2가 보기 1이 될 수도 있음)