Programming Practice for Data Science

Lecture 3: Recursion (9/27/24)

Taesup Kim
Graduate School of Data Science
Seoul National University

초보개발자 vs. 고수개발자

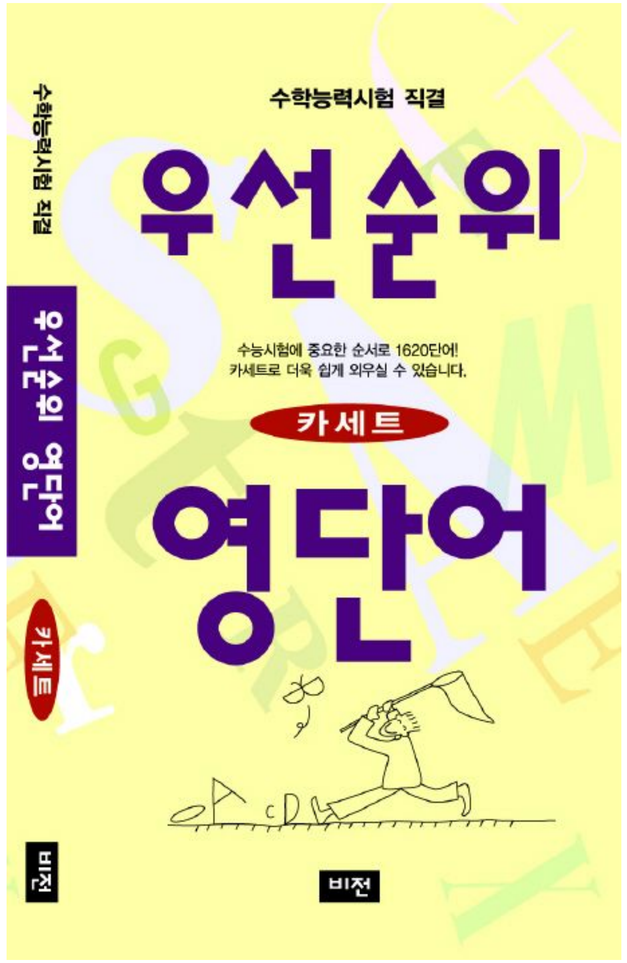
https://youtube.com/shorts/BFz4lW6vRS8?feature=shared


VERI
TAS
LUX
MEA
LAAL


N 사전
오늘의회화

N 사전

오늘의회화


수학능력시험 직결
우선순위
수능시험에 중요한 순서로 1620단어!
카세트로 더욱 쉽게 외우실 수 있습니다.
카세트
영단어
비전

브루드 커피

오늘의 커피 *Best*

Brewed Coffee

4,200원

아이스 커피

Iced Coffee

4,500원

브루드 커피

오늘의 커피 *Best*

Brewed Coffee

4,200원

아이스 커피

Iced Coffee

4,500원

지금 이 가격! 놓치지 마세요

오늘의 특가


오늘의집

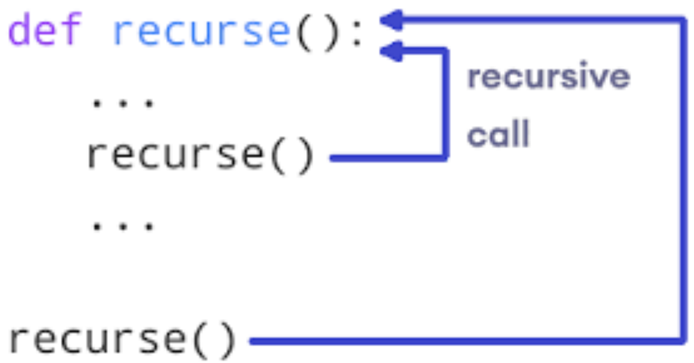
오늘의 "코딩문제"

- 하루에 한 문제씩
- 각자 페이스 / 방식으로 (셀프/동료/구글링/ChatGPT 등등)
- 제출→평가점수 (정답 중요x)

Recursive Function

Definition

- **A function calls itself directly or indirectly**
  - performing the same operations multiple times with different inputs
  - in every step, we try smaller inputs to make the problem smaller
  - base condition is needed to stop the recursion otherwise infinite loop will occur



Recursive Function

Algorithmic Steps

- **Step 1: Define a base case (termination condition)**
  - Identify the **simplest case** for which **the solution is known or trivial**
  - This is the **termination condition** for the recursion, as it prevents the function from infinitely calling itself

Recursive Function

Algorithmic Steps

- **Step 2: Define a recursive case**
  - Define the problem in terms of **smaller subproblems**
  - Break the problem down into smaller version of itself

Recursive Function

Algorithmic Steps

- **Step 3: Ensure the recursion terminations**
  - Make sure that the recursive function eventually reaches the base case
  - Does not enter an infinite loop.

Recursive Function

Example

- Fibonacci Sequence

0 1 1 2 3 5 8 13 21 34 55 …

Recursive Function

Example

- Fibonacci Sequence

0 1 1 2 3 5 8 13 21 34 55 …

Recursive Function

Example

- Fibonacci Sequence

0 1 1 2 3 5 8 13 21 34 55 …

Recursive Function

Example

- **Fibonacci Sequence**

0 1 1 2 3 5 8 13 21 34 55 …

$$F_i = F_{i-1} + F_{i-2} (i > 2)$$

Recursive Function

Example

- **Fibonacci Sequence**

0 1 1 2 3 5 8 13 21 34 55 …

$$F_i = F_{i-1} + F_{i-2} (i > 2)$$

$$F_1 = 0$$

$$F_2 = 1$$

Recursive Function

Example

- **Summation 1~n**

$$1 + 2 + 3 + \ldots + n = \sum_{i=1}^{n} i$$

$$\sum_{i=1}^{n} i = n + \sum_{i=1}^{n-1} i$$

$$F(n) = n + F(n-1)$$

Recursive Function

Example

- **Summation 1~n**

$$1 + 2 + 3 + \ldots + n = \sum_{i=1}^{n} i$$

$$\sum_{i=1}^{n} i = n + \sum_{i=1}^{n-1} i$$

$$F(n) = n + F(n-1)$$

$$F(1) = 1$$

Recursive Function

Example

- **최대공약수/ GCD (Greatest Common Divisor)**

> *유클리드 호제법(- 互除法, Euclidean Algorithm)은 2개의 자연수 또는 정식(整式)의* ***최대공약수(Greatest Common Divisor)****를 구하는 알고리즘의 하나이다.*
>
> *호제법이란 말은 두 수가 서로(互) 상대방 수를 나누어(除)서 결국 원하는 수를 얻는 알고리즘을 나타낸다.*
>
> *2개의 자연수(또는 정식) a, b에 대해서 a를 b로 나눈 나머지를 r이라 하면(단, a>b), a와 b의 최대공약수는 b와 r의 최대공약수와 같다.*
> *이 성질에 따라, b를 r로 나눈 나머지 r'를 구하고, 다시 r을 r'로 나눈 나머지를 구하는 과정을 반복하여 나머지가 0이 되었을 때 나누는 수가 a와 b의 최대공약수이다.*
>
> *이는 명시적으로 기술된 가장 오래된 알고리즘으로서도 알려져 있으며, 기원전 300년경에 쓰인 유클리드의《원론》제7권, 명제 1부터 3까지에 해당한다.*
>
> **GCD(a, b) = GCD(b, a%b)**
>
> *- 위키백과, 우리 모두의 백과사전.*

Recursive Function

Recursion vs. Iteration

Recursion	Iteration
Terminates when the base case becomes true.	Terminates when the condition becomes false.
Used with functions.	Used with loops.
Every recursive call needs extra space in the stack memory.	Every iteration does not require any extra space.
Smaller code size.	Larger code size.

Recursive Function

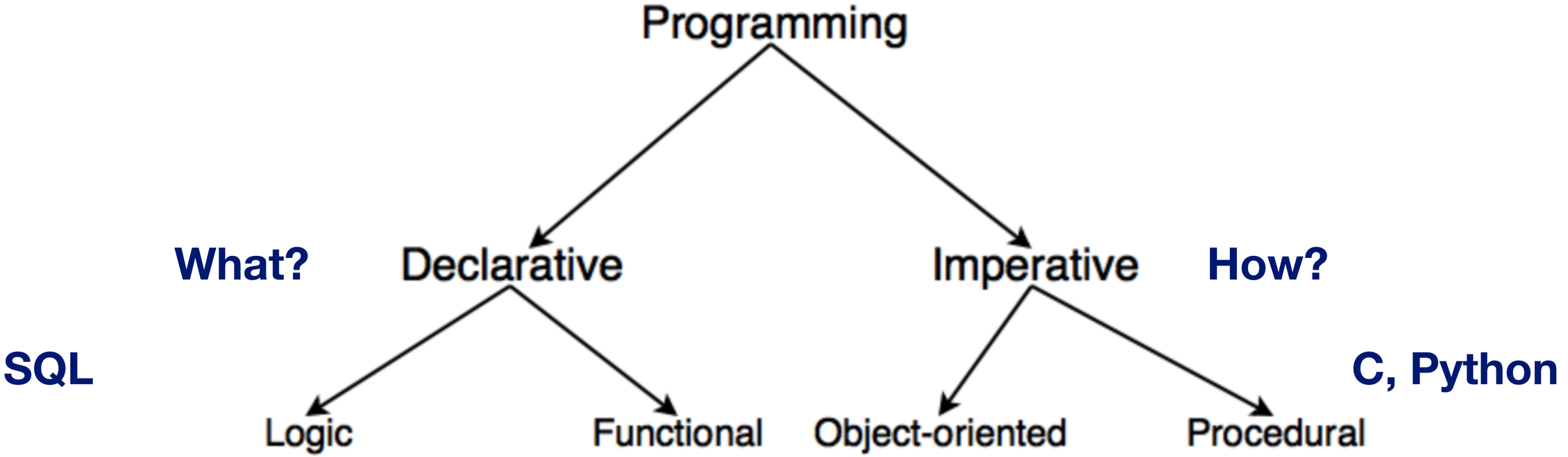
Need of Recursion

- Reduce the length of our code and make it easier to read and write

Recursive Function

Declarative Programming

- Imperative Programming vs. Declarative Programming

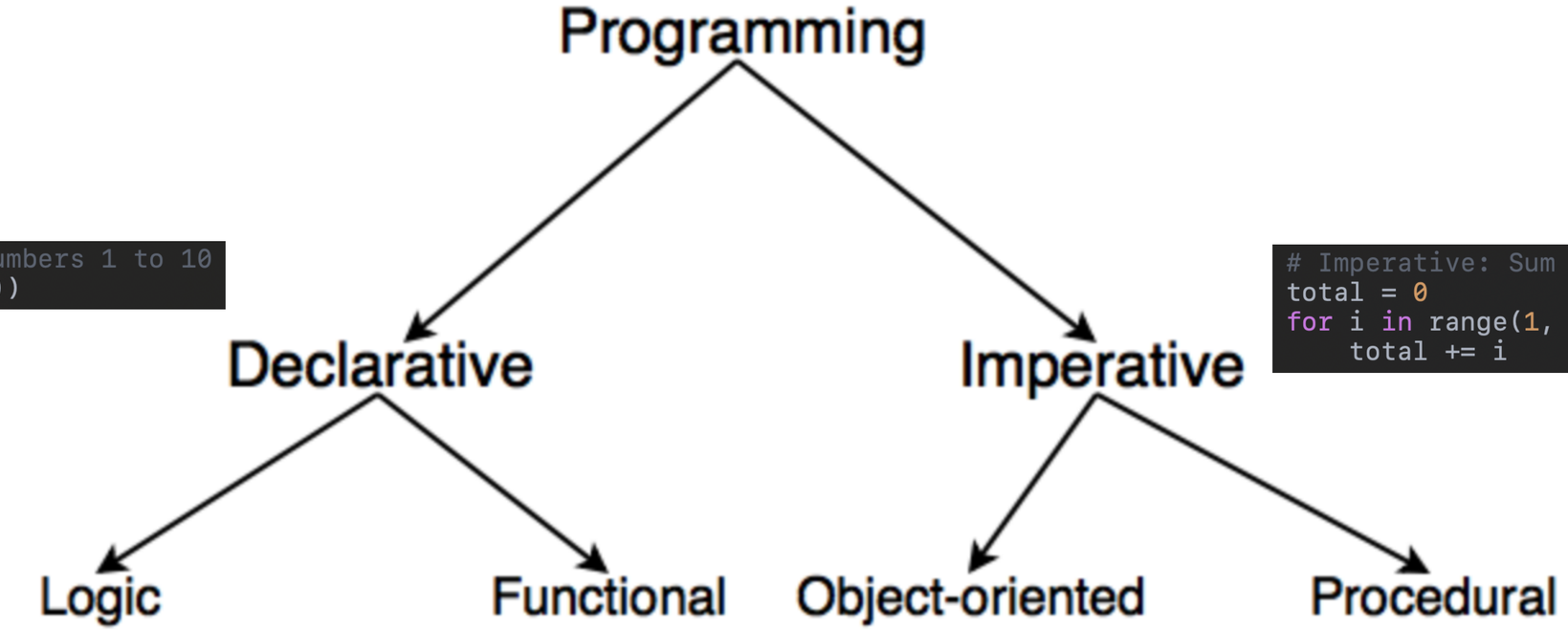


Recursive Function

Declarative Programming

- **Imperative Programming vs. Declarative Programming**

```python
# Declarative: Sum of numbers 1 to 10
total = sum(range(1, 11))
```




```python
# Imperative: Sum of numbers 1 to 10
total = 0
for i in range(1, 11):
    total += i
```

Recursive Function

Declarative Programming

- Recursive Function = Declarative Programming

Base case = When to terminate

Problem = Smaller subproblems

Recursive Function

Declarative Programming

- **Recursive Function = Declarative Programming**

Base case = When to terminate

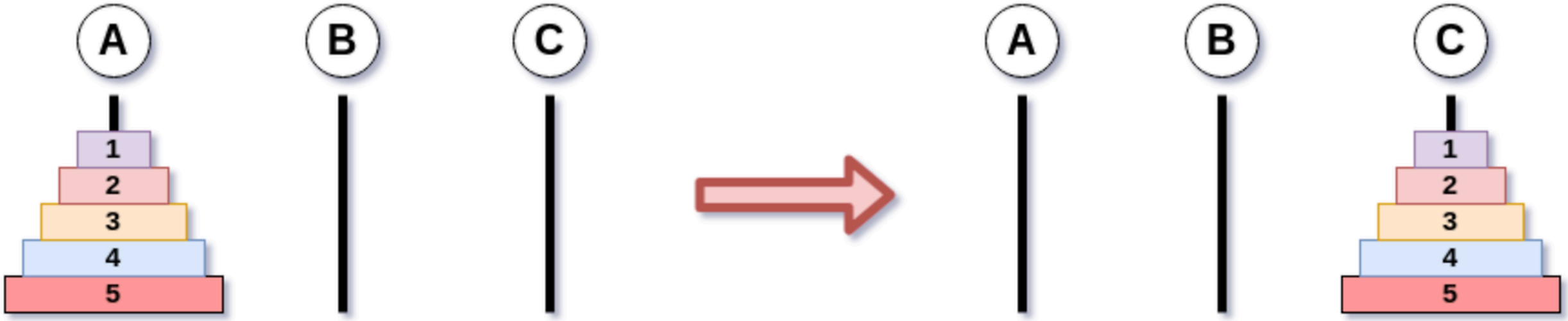
Problem = Smaller subproblems

“Reduce the length of our code and make it easier to read and write”

Recursive Function

Example

- Tower of Hanoi

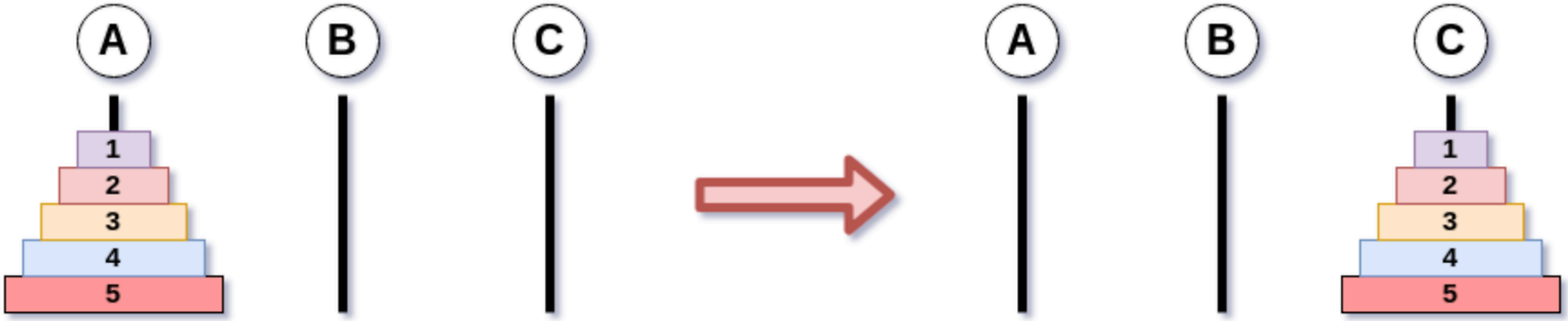


"각 원반을 어디에서 어디로 옮기는지 출력"

Recursive Function

Example

- **Tower of Hanoi**

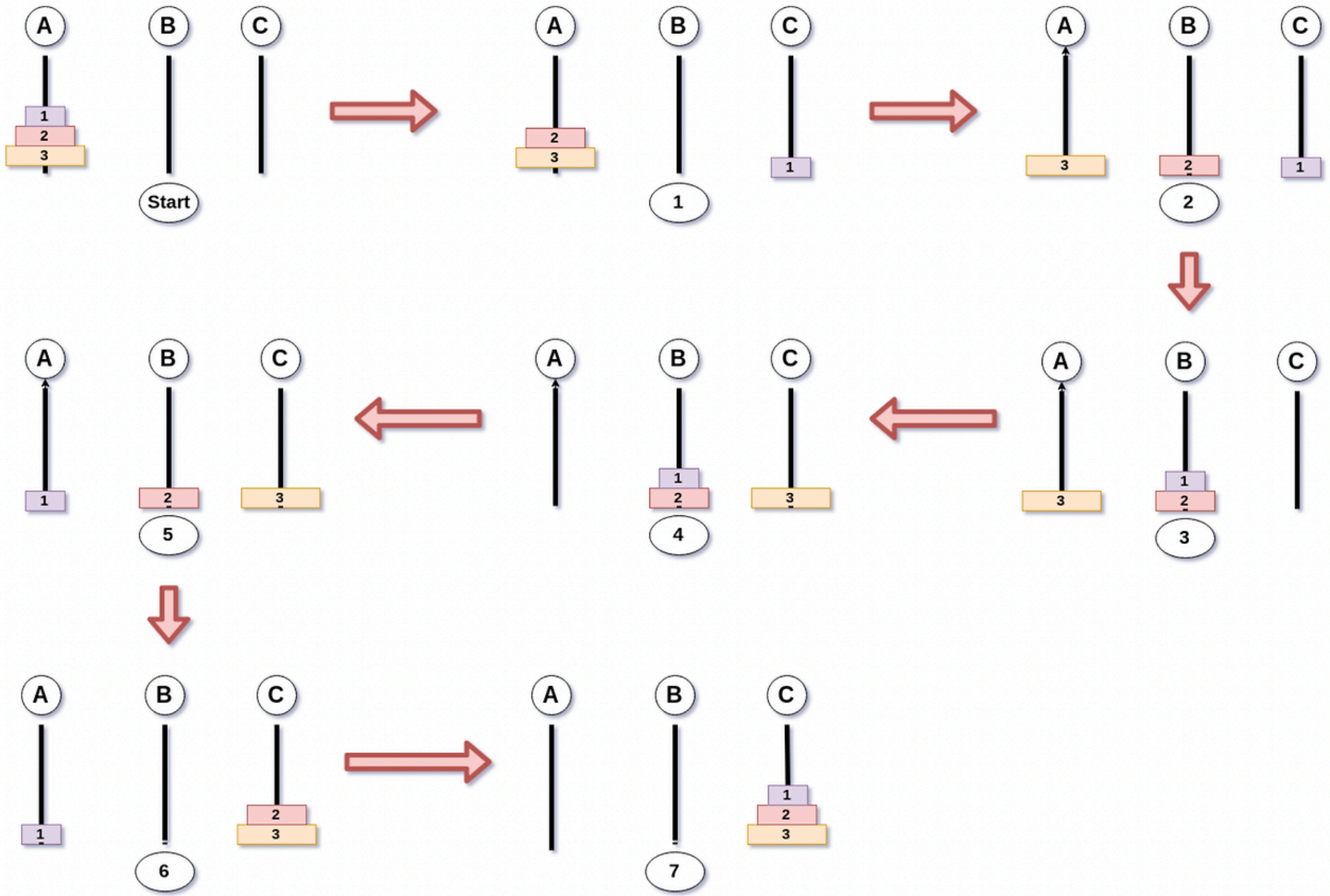


- 한번에 움직일수 있는 원반은 기둥위에 놓인 원반 하나뿐
- 어떤 원반 위에 그 보다 더 큰 원반을 쌓을수 없음

Recursive Function

Example

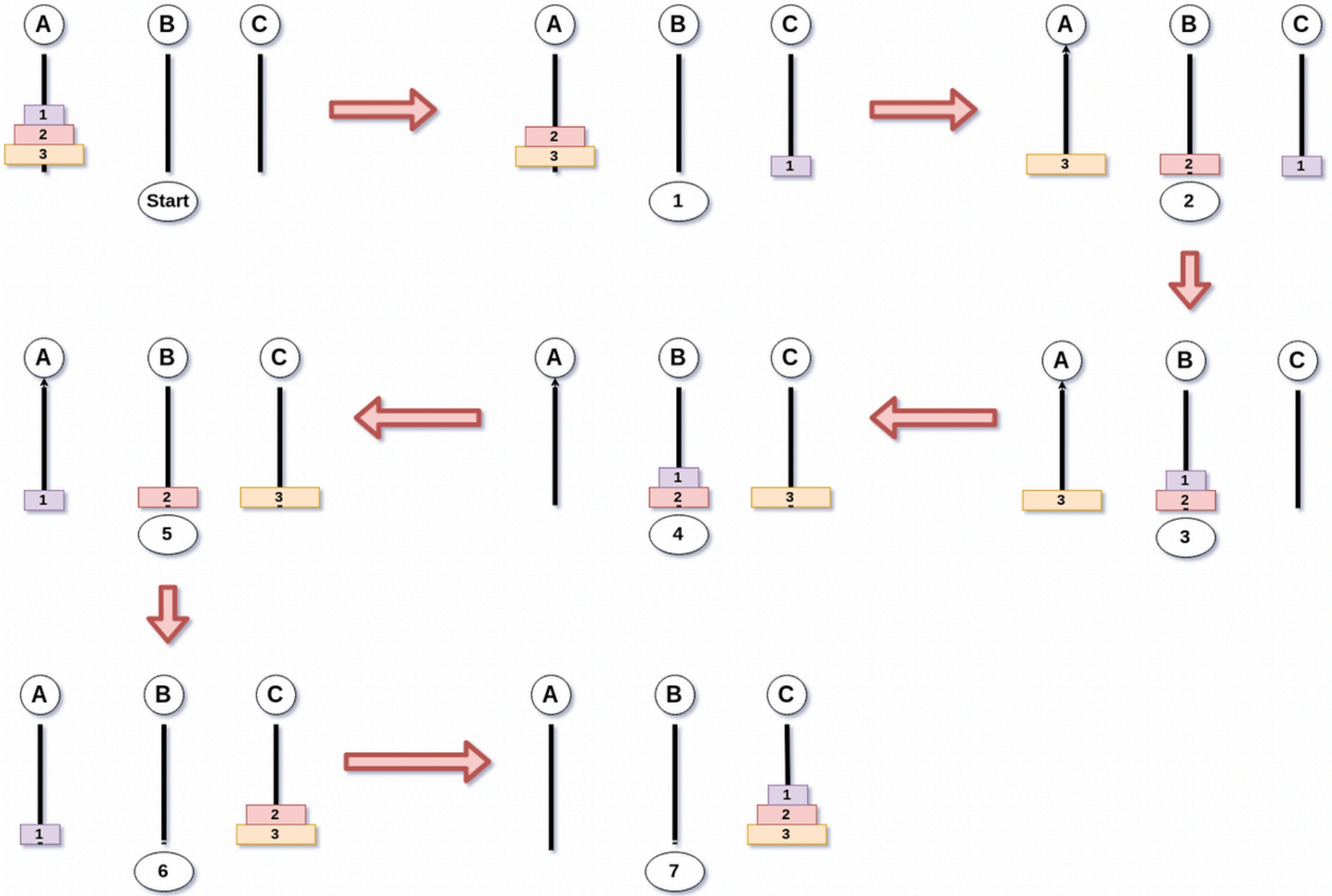
- Tower of Hanoi



Recursive Function

Example

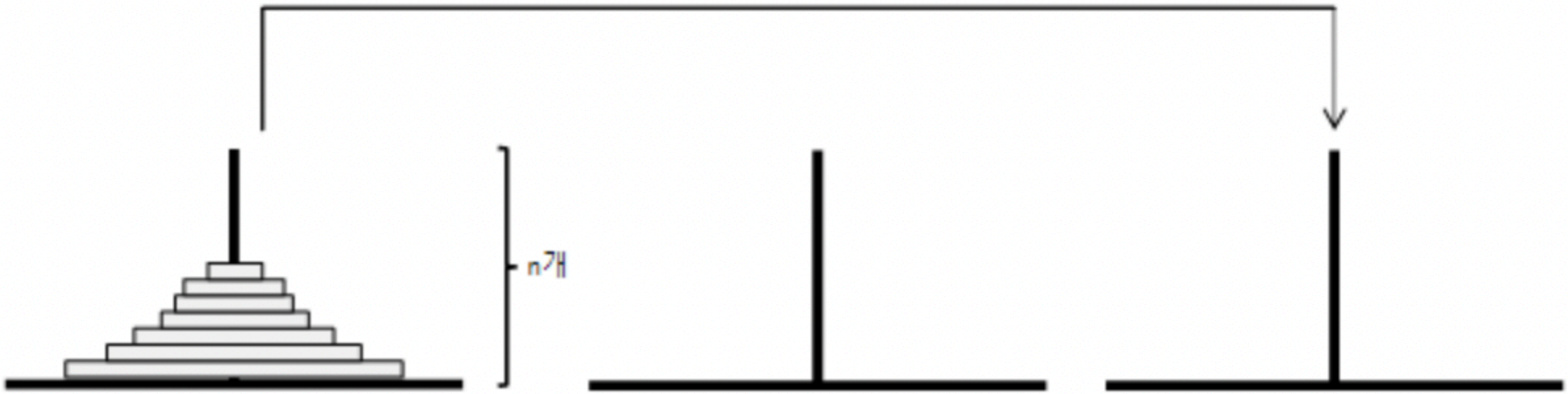
- Tower of Hanoi



Recursive Function

Example

- Tower of Hanoi with “Declarative Programming”

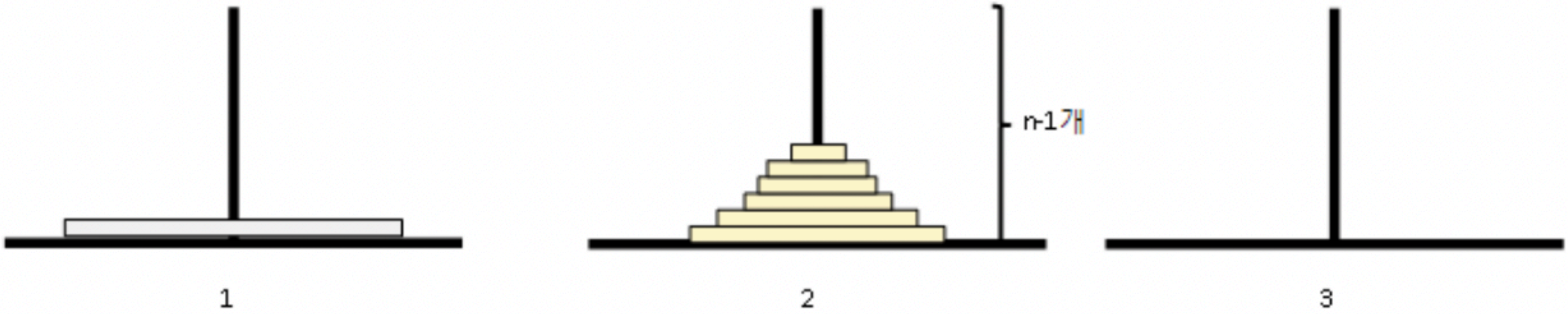


Hanoi(n, A, C)

Recursive Function

Example

- **Tower of Hanoi with “Declarative Programming”**

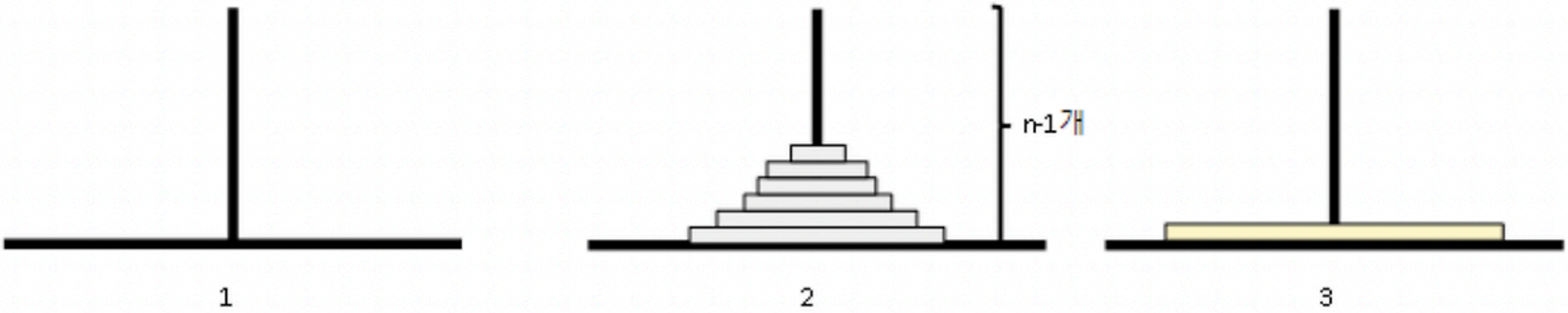


Hanoi(n-1, A, B)

Recursive Function

Example

- Tower of Hanoi with “Declarative Programming”

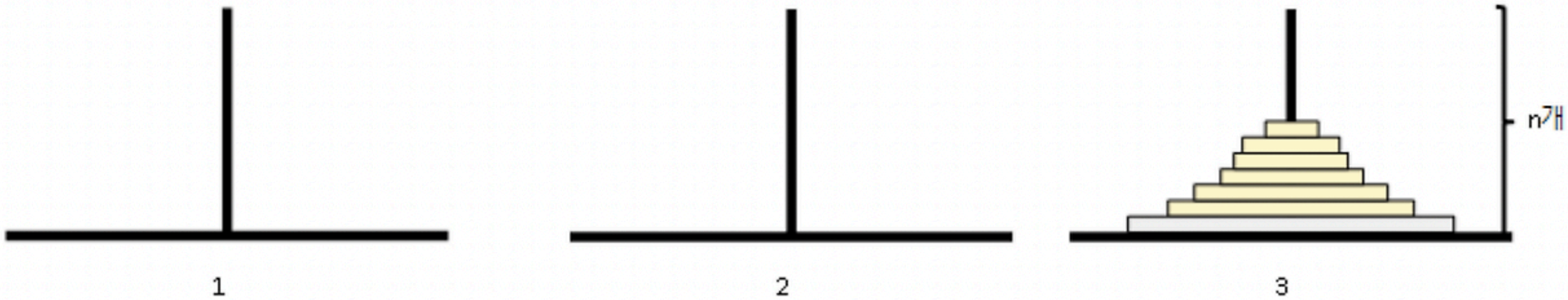


move “n” disk from A to C

Recursive Function

Example

- Tower of Hanoi with “Declarative Programming”



Hanoi(n-1, B, C)

Recursive Function

Example

- Tower of Hanoi with “Declarative Programming”

Hanoi(n, A, C)

Hanoi(n-1, A, B)

move “n” disk from A to C

Hanoi(n-1, B, C)

Recursive Function

Example

- Tower of Hanoi with “Declarative Programming”

Hanoi(n, A, C)

Hanoi(n-1, A, B)

move “n” disk from A to C

Hanoi(n-1, B, C)

Hanoi(n, A, C, ??)

Recursive Function

Example

- **Permutation (https://leetcode.com/problems/permutations/description/)**

Example 1:

```
Input: nums = [1,2,3]
Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

Example 2:

```
Input: nums = [0,1]
Output: [[0,1],[1,0]]
```

Example 3:

```
Input: nums = [1]
Output: [[1]]
```

Recursive Function

Example

- **Palindrome**
    - **F(a) = (a[0] ==a[-1]) and F(a[1:-1])**

Input: *N = 12321*
Output: *Yes*
Explanation: *12321 is a Palindrome number because after reversing its digits, the number becomes 12321 which is same as the original number.*

Input: *N = 1234*
Output: *No*
Explanation: *1234 is not a Palindrome number because after reversing its digits, the number becomes 4321 which is different from the original number.*