재귀

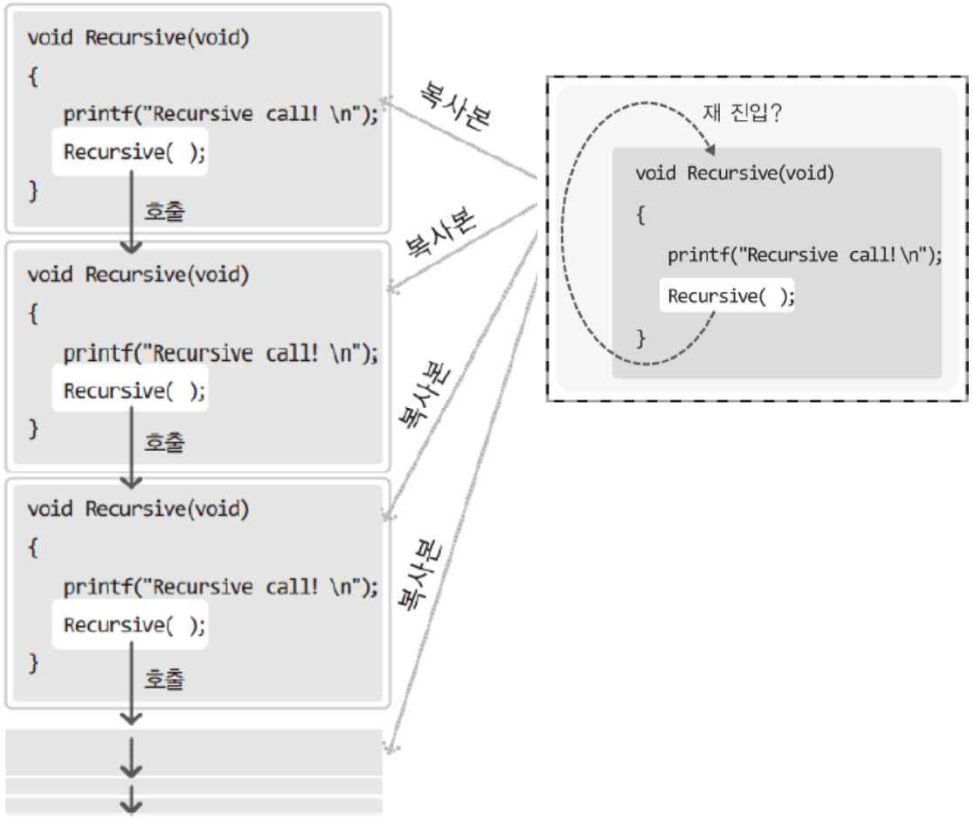
2주차-강의

남춘성

SUNG KYUN KWAN UNIVERSITY

재귀함수 정의

- 재귀함수
  - 함수 내에서 자기 자신을 다시 호출하는 함수를 의미 함.




```
void Recursive(int num)
{
    if(num <= 0)        // 재귀의 탈출조건
        return;         //재귀의 탈출!
    printf("Recursive call! %d \n", num);
    Recursive(num-1);
}

int main(void)
{
    Recursive(3);
    return 0;
}
```

```
Recursive call! 3
Recursive call! 2
Recursive call! 1
```

재귀함수 디자인 방법

- 재귀조건 정의
  - 자기 자신을 호출하기 위한 반복조건 정의
- 종료 조건 정의
  - 함수가 종료되기 위한 조건 정의
- Factorial의 예

$$n! = n \times \underbrace{(n-1) \times (n-2) \times (n-3) \times \ldots \times 2 \times 1}_{(n-1)!}$$

$$n! = n \times (n-1)!$$

재귀조건

$$f(n) = \begin{cases} n \times f(n-1) & \ldots. n \geq 1 \\ 1 & \ldots. n = 0 \end{cases}$$

종료조건

```
if(n == 0)
    return 1;
else
    return n * Factorial(n-1);
```

재귀함수 - 피보나치 수열 I

SUNG KYUN KWAN UNIVERSITY

- 피보나치 수열
  - 정의 : 앞의 수 2개를 더해서 현재의 수를 만들어가는 수열
  - 표현 : n번째 값 = 수열의 n-1번째 값 + 수열의 n-2번째 값

$$fib(n) = \begin{cases} 0 & \ldots\ldots n=1 \\ 1 & \ldots\ldots n=2 \\ fib(n-1)+fib(n-2) & \ldots\ldots \text{otherwise} \end{cases}$$

  - 예) 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …

재귀함수 - 피보나치 수열 II

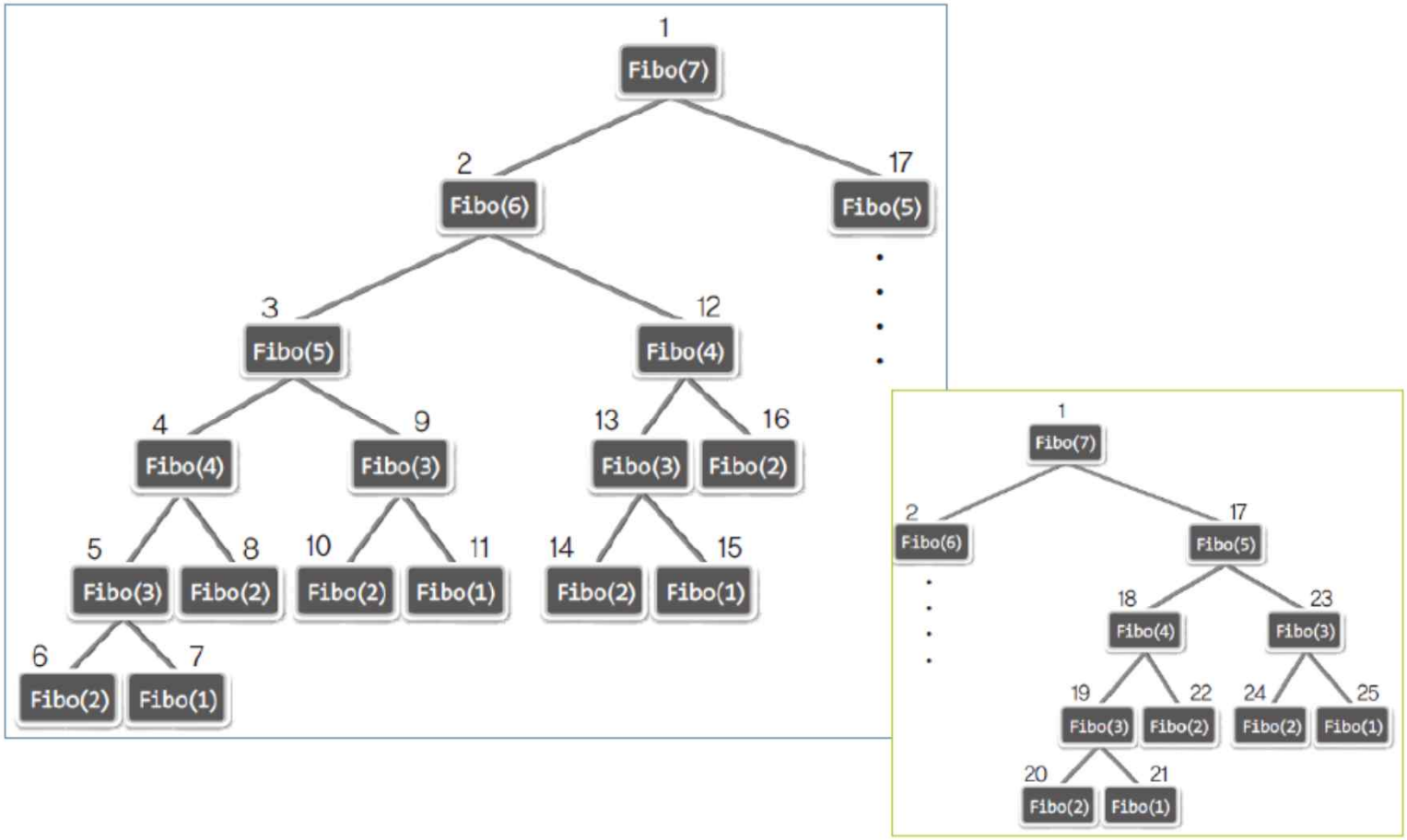
- 피보나치 수열 코드 구현

$$fib(n) = \begin{cases} 0 & \ldots\ n=1 \\ 1 & \ldots\ n=2 \\ fib(n-1)+fib(n-2) & \ldots\ \text{otherwise} \end{cases}$$

```
int Fibo(int n)
{
    if(n == 1)
        return 0;
    else if(n == 2)
        return 1;
    else
        return Fibo(n-1) + Fibo(n-2);
}
```

재귀함수 - 피보나치 수열 III

- 피보나치 수열 함수 흐름



재귀함수 - 이진 탐색 알고리즘 I

SUNG KYUN KWAN UNIVERSITY

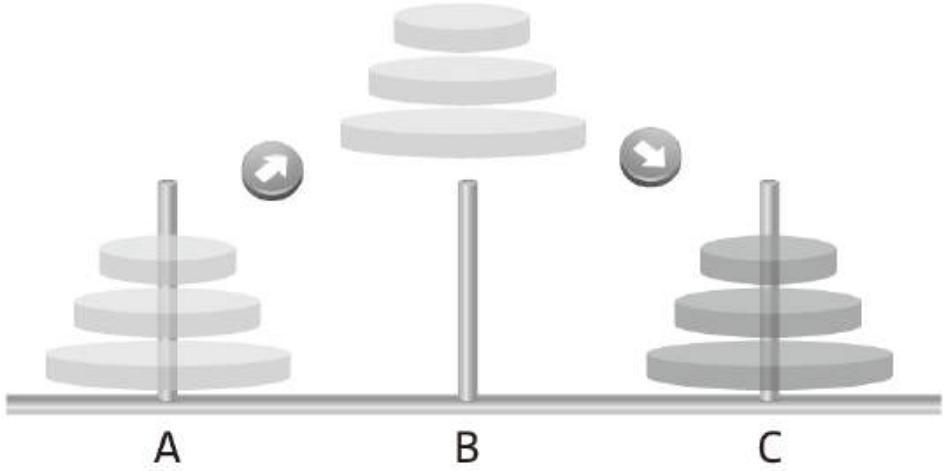
- 이전에 배웠던 이진 탐색에서의 알고리즘 핵심
  - 탐색 범위 중앙에 목표 값이 저장되었는지 확인
  - 저장되지 않았다면 **탐색 범위를 반으로 줄여서 다시 탐색**

```
int BSearchRecur(int arr[], int first, int last, int target) {
  int mid;
  if (first > last)
    return -1;

  mid = (first + last) / 2;

  if (arr[mid] == target)
    return mid;
  else if (target < arr[mid])
    return BSearchRecur(ar, first, mid - 1, target);
  else
    return BSearchRecur(ar, mid + 1, last, target);
}
```

재귀함수 – 하노이타워 I

SUNG KYUN KWAN UNIVERSITY

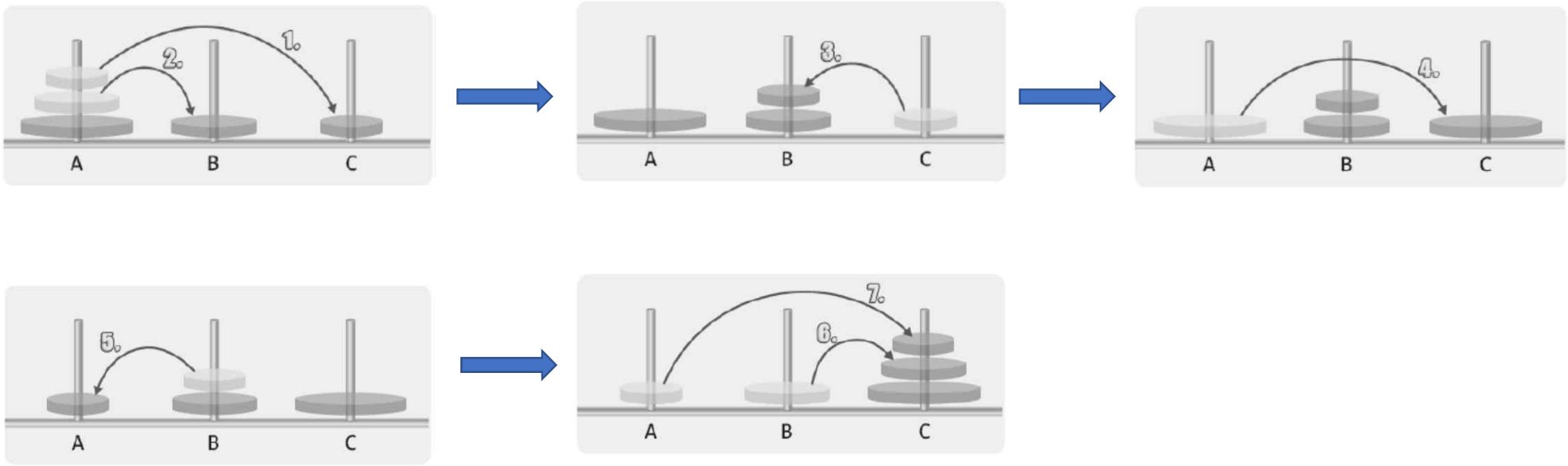
- 하노이 타워 문제
  - 정의
    - 하나의 막대에 쌓여 있는 원반을 다른 하나의 원반에 그대로 옮기는 방법
  - 조건
    - 원반은 한 번에 하나씩만 옮기는 것이 가능
    - 작은 원반의 위에 큰 원반이 올 수가 없음



재귀함수 – 하노이타워 II

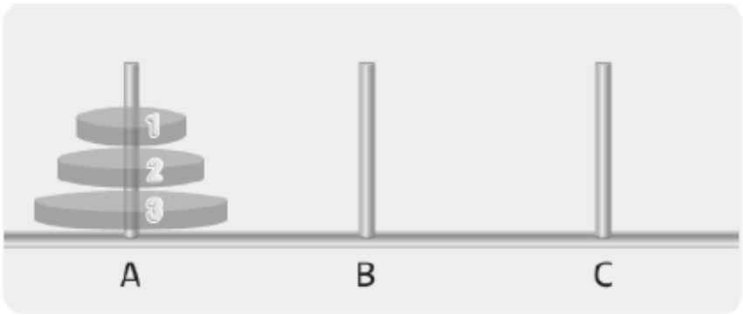
SUNG KYUN KWAN UNIVERSITY

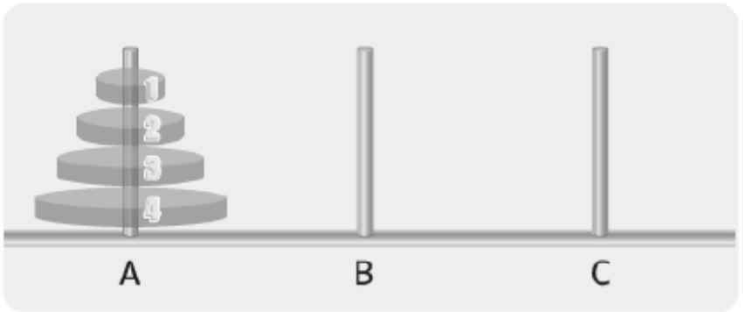
- 하노이 타워 수행



재귀함수 – 하노이타워 III

- 하노이 타워 반복 패턴 찾기
  - 원반의 개수는 상관이 없음





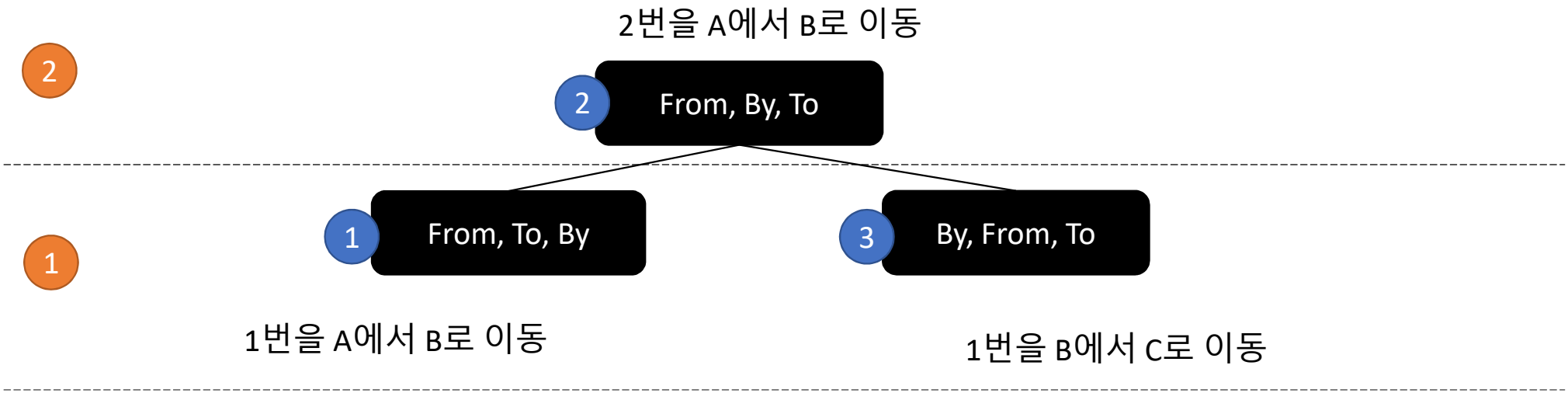
A 원반을 C로 옮기기 위해선 다음 조건이 만족:
원반 3이 C로 옮겨져야 함. 그러기 위해서는 원반 B에 원반 1과 2를 옮겨야 함.

조건이 같음(반복패턴이 같음)

A 원반을 C로 옮기기 위해선 다음 조건이 만족:
원반 4가 C로 옮겨져야 함. 그러기 위해서는 원반 B에 원반 1,2,3을 옮겨야 함.

재귀함수 – 하노이타워 IV

- 하노이 타워 반복 패턴 찾기
  - 과정 정리
    - 1. 큰 원반 n개를 A에서 C로 이동 : HanoiTowerMove(num, from, by, to);
    - 2. 작은 원반 n-1개를 A에서 B로 이동 : HanoiTowerMove(num-1, from, to, by);
    - 3. 큰 원반 1개를 A에서 C로 이동 : print(move it);
    - 4. 작은 원반 n-1개를 B에서 C로 이동 : HanoiTowerMove(num-1, by, from, to);



재귀함수 – 하노이타워 V

- 예)