알고리즘

신제용

ZEROBASE SCHOOL

기초 알고리즘

신제용

재귀 호출과 반복
(Recursive Calls and Iterations)

신 제 용

01 점화식과 재귀 호출

점화식 표현과 이를 재귀 호출로 구현하는 과정을
학습합니다.

학습 키워드 - 점화식, 재귀, 재귀 함수

재귀 호출 (Recursive call)

- **함수가 자기 자신을 호출**하는 것을 재귀 호출이라 한다.
- **분할 정복**(Divide & Conquer), **점화식** 등을 구현하는 데에 많이 사용된다.
- 재귀 구현은 항상 **반복(Iteration) 구현으로 변환**될 수 있다.

점화식 (Recurrence relation)

- 재귀식(Recursion relation)이라고도 부르며, **수열의 항 사이의 관계**를 나타낸다.

- 점화식으로 표현된 수열을 n에 대한 식으로 표현하는 것을 '풀이한다'(solve)고 하며, 풀이한 결과를 **일반식**이라고 한다.

- 대표적인 점화식의 예

  피보나치 수열: $f(n) = f(n-1) + f(n-2)$

  팩토리얼: $f(n) = n \times f(n-1)$

  등차 수열: $f(n) - f(n-1) = d$

  등비 수열: $f(n)/f(n-1) = r$

재귀 함수의 구현

- 재귀 호출을 할 때에는 반드시 **탈출 조건**이 필요하다.
  → 탈출 조건이 없으면, 재귀 호출은 무한히 계속된다.

- **점화식에 의거하여 재귀 호출**을 수행한다.
  → 입력 파라미터를 달리하여 **결국 탈출 조건에 도달**할 수 있게 한다.

```python
def fibonacci(n):
    if n < 2: # 탈출 조건
        return n
    return fibonacci(n-1) + fibonacci(n-2) # 재귀 호출(점화식 구현)
```

분할 정복 (Divide & Conquer)

- 재귀 호출을 이용하여 큰 문제를 작은 문제로 나누어 해결하는 방법
  → 재귀 호출을 이용해 **Top-Down 형식**으로 구현

```python
def sum_all(x):
    if len(x) == 1: # 종료 조건1
        return x[0]

    if len(x) == 0: # 종료 조건2
        return 0

    mid = len(x) // 2

    return sum_all(x[:mid]) + sum_all(x[mid:]) # 분할 정복
```

02 점화식의 반복 구현

다음 챕터에서는 점화식을 재귀 호출이 아닌
반복을 이용해 구현하는 방법을 학습합니다.

02 점화식의 반복 구현

점화식을 재귀 호출이 아닌 반복문을 이용해 구현하는
방법을 학습합니다.

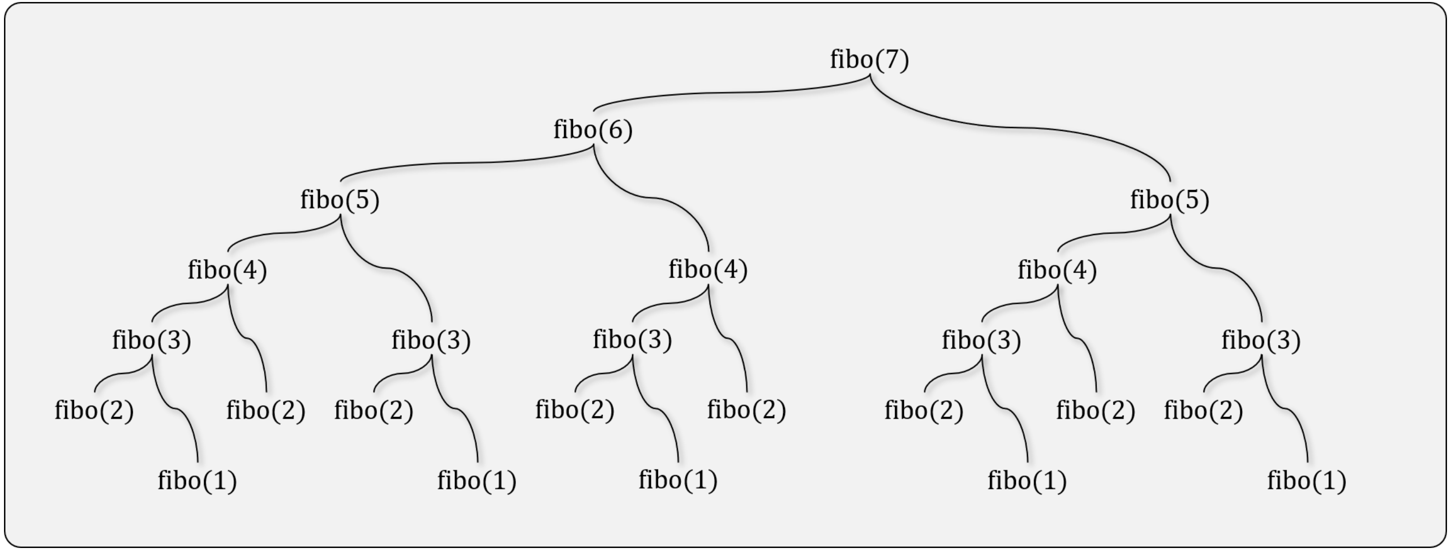
학습 키워드 - 점화식, 반복문, 스택

재귀 호출의 한계

- 여러 번 재귀 호출이 발생하는 경우, **기하급수적으로 호출 횟수가 증가**
→ 함수 호출 스택(Function call stack)의 크기에 제한이 있어, 일정 횟수 이상 호출이 불가하다.

- 실질적인 계산에 필요한 연산보다, **함수 호출에 의한 Overhead**가 발생한다.

재귀 호출의 한계 (피보나치 수열)



반복 구현 (Iteration)

- **종료조건을 초기값**(initial value)로 하며, **Bottom-Up으로 구현**

```
def fibonacci(n):
    fn_1, fn = 0, 1  # 초기값 설정

    if n == 0:
        return fn_1
    if n == 1:
        return fn

    for i in range(2, n + 1):  # 반복문 구현
        fn_1, fn = fn, fn_1 + fn

    return fn
```

Tail Recursion

- 재귀 함수에서, **재귀 호출이 마지막에 단 한번 수행**되는 것을 의미한다.

- 컴파일러에서 Tail Recursion 최적화를 지원하는 경우, **함수 호출 스택을 재활용**한다.

- 컴파일러에서 최적화를 지원하지 않으면 적용되지 않는다. (Python은 미지원)

```python
def fibonacci(n, a=0, b=1):
    if n == 0:
        return a
    if n == 1:
        return b

    return fibonacci(n-1, b, a+b)
```

03 재귀 예시 문제 풀이

다음 챕터에서는 재귀 구현을 기초 유형 풀이를 통해 학습합니다.

03 재귀 예시 문제 풀이

재귀 함수로 해결할 수 있는 문제를 직접 해결해 보면서
재귀 구현을 확실하게 이해합니다.

학습 키워드 - 재귀, 점화식, 구현

Problem1

문제 설명

카탈랑 수는 0번, 1번, 2번, ... 순으로 아래와 같이 구성되는 수열을 의미한다.

- 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, ...

이를 점화식으로 나타내면 아래와 같다.

$$C_0 = 1, C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i} \ (n \geq 0)$$

카탈랑 수의 `n` 번째 값을 구하는 프로그램을 작성하세요.

입력 예시

입력	출력
0	1
2	2
5	42
7	429

Problem2

문제 설명

회문 또는 팰린드롬(palindrome)은 앞 뒤 방향으로 같은 순서의 문자로 구성된 문자열을 말한다.

- 예시) `'abba'`, `'kayak'`, `'madam'`

유사회문은 문자열 그 자체는 회문이 아니지만 한 문자를 삭제하면 회문이 되는 문자열을 말한다.

- 예시) `'summuus'` 의 5번째 또는 6번째 문자 `'u'` 를 제거하면 `'summus'` 인 회문을 만들 수 있다.

주어진 문자열을 확인한 후 문자열 종류에 따라 다음과 같이 출력하는 프로그램을 작성하세요.

- 회문: 0
- 유사회문: 1
- 기타: 2

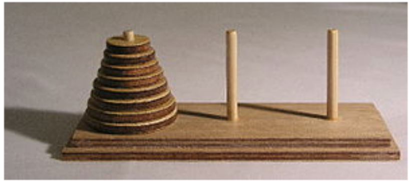
입력 예시

입력	출력
`'abba'`	`0`
`'summuus'`	`1`
`'xabba'`	`1`
`'xabbay'`	`2`
`'comcom'`	`2`
`'comwwmoc'`	`0`
`'comwwtmoc'`	`1`

Problem3

문제 설명

하노이의 탑은 퍼즐의 일종이다.



(Tower of Hanoi from: wikipedia)

하노이의 탑 퍼즐 게임 규칙은 다음과 같다.

- 한 번에 한 개의 원판 만 옮길 수 있다.
- 큰 원판이 작은 원판 위에 있어서는 안된다.

원판의 개수 `n` 이 주어졌을 때, 가장 왼쪽 기둥으로부터 끝 기둥으로 이동하는 과정을 출력하는 프로그램을 구현하세요.

단, 초기에 모든 원판은 가장 큰 원판부터 순서대로 왼쪽 기둥에 위치해 있다.

입력 예시

입력	출력
`2`	`[[1, 2], [1, 3], [2, 3]]`
`3`	`[[1, 3], [1, 2], [3, 2], [1, 3], [2, 1], [2, 3], [1, 3]]`