자료구조 4주차 실습

Recursion

감성인공지능연구실
방윤석 임희수

인하대학교
INHA UNIVERSITY

퀴즈 안내

인하대학교 INHA UNIVERSITY

- 퀴즈 일정 변경 : 4월4일 -> 4월 11일
- 퀴즈 범위 : 이번주 진도까지
- 퀴즈 시간 : 금요일 오전 이론 수업시간(다음주 재 공지)
- OT때 공지한 것처럼 퀴즈는 VSCode로만 진행
- 개인컴퓨터 사용 금지(자신의 자리가 VSCode 잘 실행 되는지 확인할 것)
- 문제 파일과 VSCode를 제외한 파일이 열려있을시 퇴장조치
- 자동완성기능 적발시 퇴장 조치(OT 파일에 끄는 법 자세히 안내)

ADT

ADT (Abstract Data Type, 추상 자료형)

- 자료들과 그 자료들에 대한 연산을 명확히 정의하는 것 (from wikipedia)
- **구현 방법은 명시되어 있지 않다**. 사용자에게 사용할 수 있는 **인터페이스(method)만 제시**하는 것
- **무엇을 할 지**에 집중하는 것
- (예 : stack은 데이터를 넣는 PUSH과 FILO으로 데이터를 빼내는 POP이 있음)

자료 구조

- 효율적인 접근 및 수정을 가능케 하는 자료의 조직, 관리, 저장을 의미 (from wikipedia)
- **구체적인 구현방식도 제시**된다.
- **어떻게 할 지**에 집중하는 것
- (예 : stack이라는 ADT를 배열로 구현)

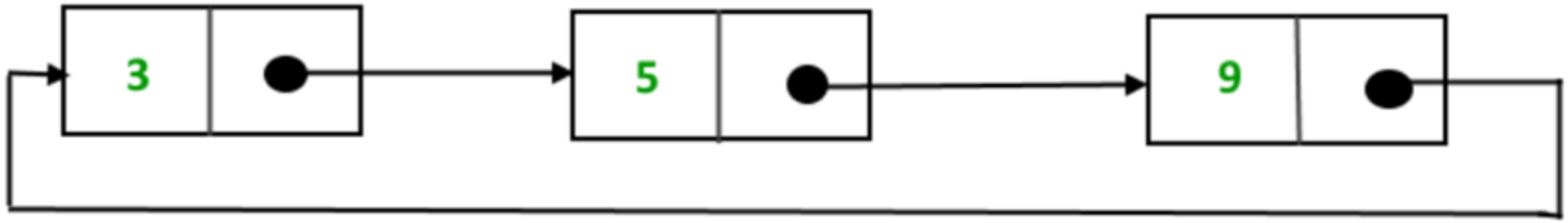
수업에 배우는 것은 자료구조 중 일부

- 자료구조 구현에 **하나의 답만 존재하지 않음**
- 문제 상황에 따라 **적합한 자료구조를 만드는 능력**이 필요

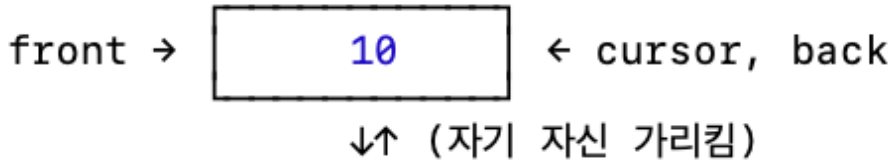
- 예) 문제상황 : 데이터를 순차적으로 저장하고, 이를 차례대로 출력해야 함
  - 배열을 이용
  - linked list를 이용
    - singly linked list를 이용
    - doubly linked list를 이용
    - circular linked list를 이용
  - queue를 이용
    - deque을 이용

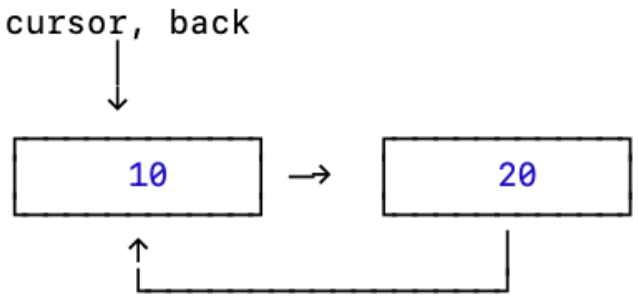
등등.. 문제 상황을 판단해서 적합한 자료구조를 구현하시면 됩니다.

Revisit Circular Linked List

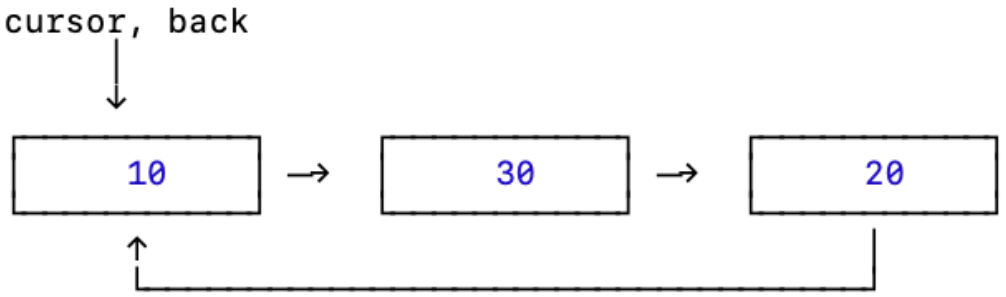


Revisit Circular Linked List





front = cursor.next = [20]



front = cursor.next = [30]

Recursion

Iteration vs Recursion

	Iteration(반복문)	Recursion(재귀함수)
정의	반복문을 여러번 실행해서 반복작업 진행	함수(자기 자신)를 다시 호출하며 반복 작업 진행
stack 메모리	사용하지 않음	호출 시 함수 관련된 변수들이 stack 메모리에 저장됨
무한 반복시	CPU 사이클을 반복적으로 사용 (문제는 없음)	stack overflow를 발생시킴
속도	빠름	느림
종료 조건	반복문의 종료 조건 만족시	함수가 더이상 호출되지 않을 시

함수의 호출

함수 호출 시 매개변수, return 값, 함수 종료 후 돌아갈 위치가 프로그램의 stack에 저장됨

- 프로그램 실행 시 메모리에 프로그램 코드 외에 stack 등의 구조도 같이 들어감
- python의 함수 재귀 호출 한도는 1,000

재귀 말고 반복문만 쓰는게 효율적인 것 아닌가?

- 재귀로 쓰면 프로그램이 이해하기 쉽게 바뀐다.
- 프로그램의 성능과 코드 작성자의 가독성과 유지보수성을 trade-off 하는 것

Iteration vs Recursion

Iterative :

```
def factorial_iterative(self, n):  1개의 사용 위치
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
```

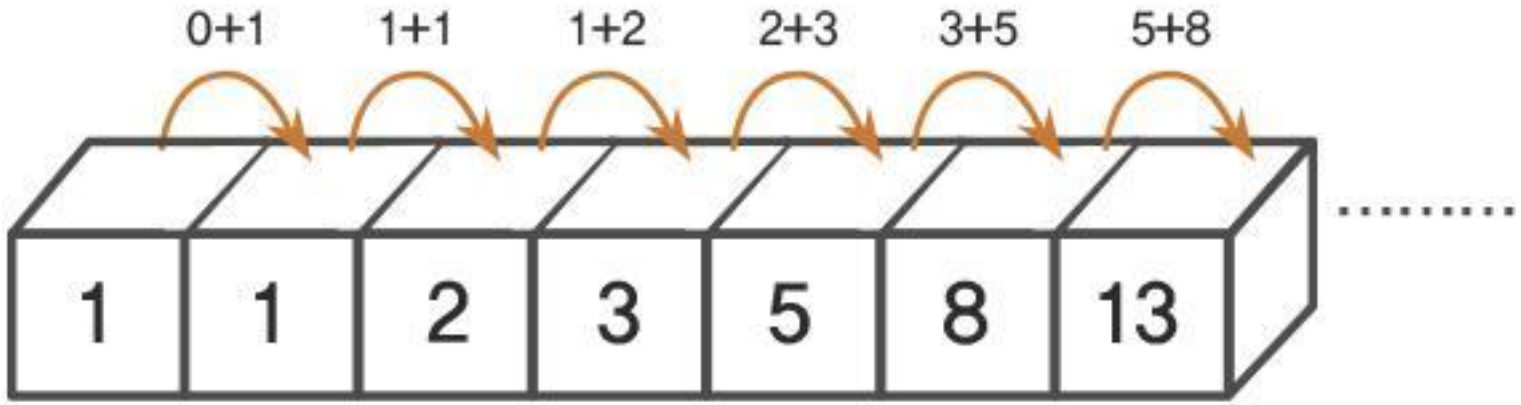
Recursive :

```
def factorial_recursive(self, n):  2개의 사용 위치
    if n == 0:
        return 1
    return n * self.factorial_recursive(n - 1)
```

Exercise #1

인하대학교 INHA UNIVERSITY

피보나치 수열



피보나치 수열은 $a_0 = 1, a_1 = 1, a_n = a_{n-1} + a_{n-2}$ 을 만족시키는 수열이다.

1.재귀 함수를 이용해 구현하세요
2. 반복문을 통해 구현하세요

Exercise #2

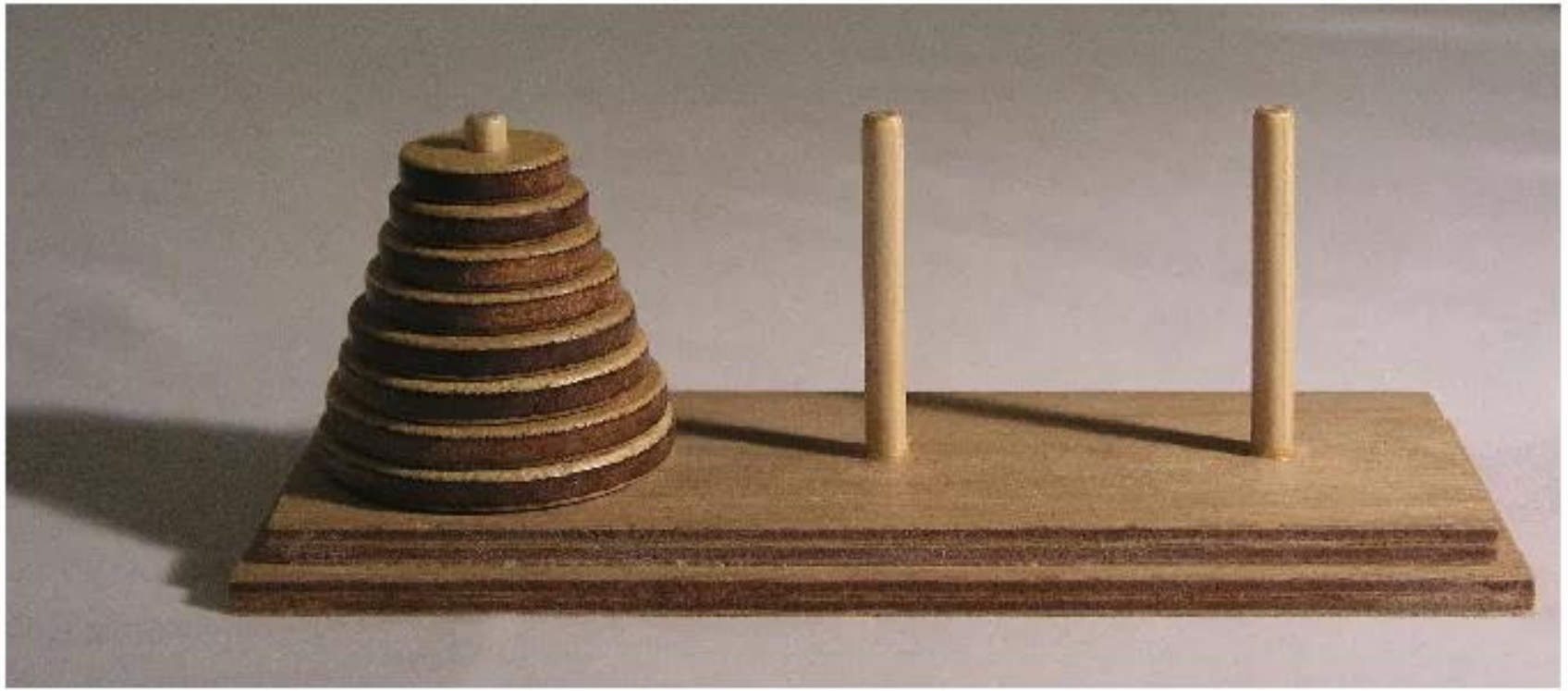
Recursive Reversed LinkedList

Reversed LinkedList를 Recursive 하게 구현해보세요

Problem #1

인하대학교 INHA UNIVERSITY

하노이의 탑(Hanoi Tower)은 **세 개의 기둥**과 여러 개의 크기가 서로 다른 원반을 가지고 있으며, 이 원반들을 한 기둥에서 다른 기둥으로 옮기는 퍼즐이다. 단, 한 번에 하나의 원반만 옮길 수 있고, **작은 원반 위에 큰 원반을 놓을 수 없다.**



3개의 기둥에 원반들을 쌓아 놓고 다른 쪽으로 옮기는 게임.

Problem #1

주어진 원반 개수 `n`에 대해, 원반을 모두 **시작 기둥 A**에서 **목표 기둥 C**로 옮기는 **전체 이동 경로**를 출력하도록 메소드를 작성하라.
각 이동은 **어떤 디스크**가, **어떤 기둥**에서, **어떤 기둥으로 이동**했는지를 포함해야 한다.

- 기둥 이름은 'A', 'B', 'C'로 주어진다.
- 디스크 번호는 1(가장 작은 원반)부터 n(가장 큰 원반)까지이다.
- 출력은 “Move disk n from X to Y 의 꼴로 출력하도록 한다.

Problem #1

메소드 설명

Hanoi(n = 5,source = "A", destination = "C", auxiliary = "B")

n개의 원반을 기둥 A에서 C로 옮기는 전체 과정을 출력한다.

예시

```
def main():
    Hanoi(n = 5,source = "A", destination = "C", auxiliary = "B")
```

```
Move disk 1 from source A to destination C
Move disk 2 from source A to destination B
Move disk 1 from source C to destination B
Move disk 3 from source A to destination C
Move disk 1 from source B to destination A
Move disk 2 from source B to destination C
Move disk 1 from source A to destination C
Move disk 4 from source A to destination B
Move disk 1 from source C to destination B
```

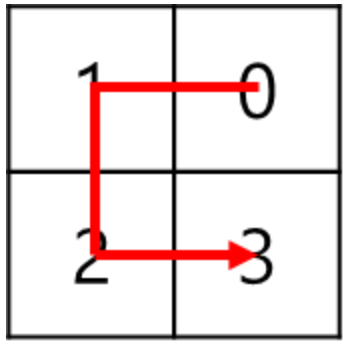
Problem #2

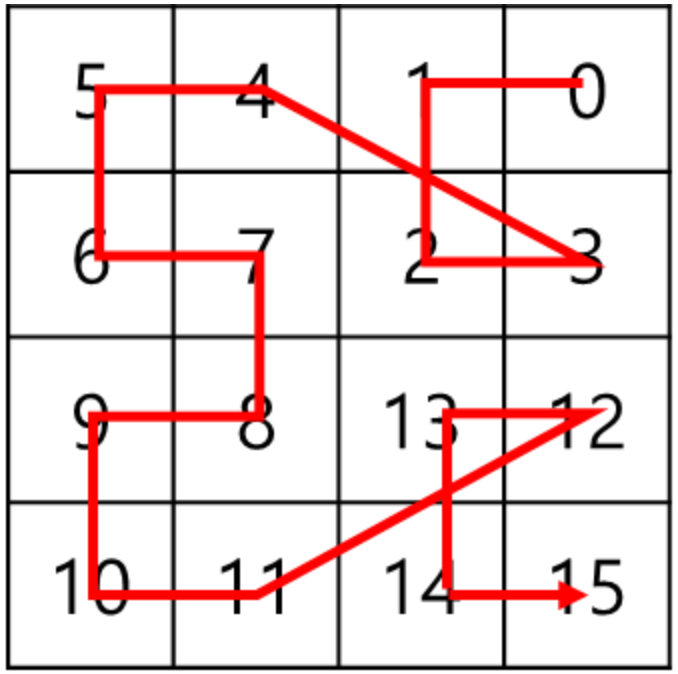
인하대학교 INHA UNIVERSITY

크기가 $2^N \times 2^N$ 인 배열을 ㄷ 모양으로 탐색하려고 한다. 2X2 배열에서 오른쪽 위 칸, 왼쪽 위 칸, 왼쪽 아래칸, 오른쪽 아래칸으로 움직이면 ㄷ 모양이다.

N>1인 경우, $2^{N-1} \times 2^{N-1}$ 으로 4등분 한 뒤, 재귀적으로 순서대로 방문한다.

아래는 탐색 순서 예시이다.





Problem #2

인하대학교 INHA UNIVERSITY

메소드 설명

- getVisitOrder(int n, int r, int c) : $2^n \times 2^n$ 격자에서 (r, c)가 방문되는 순서를 반환한다.

예시

```
def main():
    print(findVisitOrder(1, 1, 1))
    print(findVisitOrder(1, 0, 1))
    print(findVisitOrder(1, 0, 0))
    print(findVisitOrder(1, 1, 0))
    print(findVisitOrder(2, 1, 1))
```

```
1
2
3
4
9
```