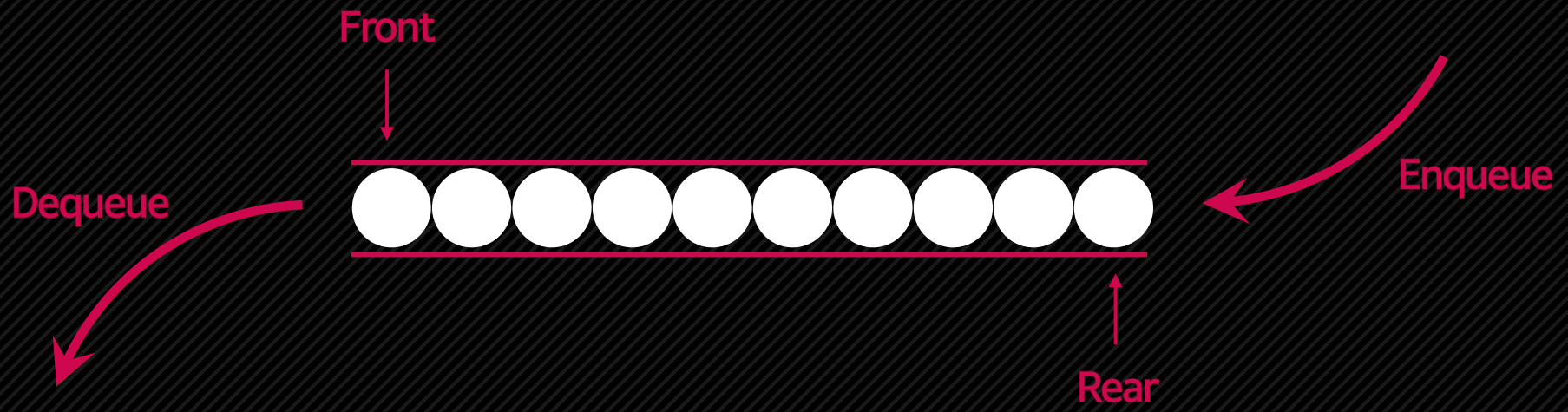
알고리즘 특강
큐/스택

기본적인 자료구조인 큐와 스택에 대해 알아봅시다.
이후 다양한 알고리즘을 공부할 때 기본이 되니 잘 숙지하셔야 합니다.

큐



Queue

- FIFO (First In, First Out) 구조를 띄고 있는 자료구조로, 삽입과 삭제 연산이 서로 다른 한군데에서 발생함.
- 삽입/삭제 연산에 있어 시간복잡도가 O(1)임.
- 맨 앞을 front, 맨 뒤를 rear 라고 하고, 삽입 연산을 enqueue, 삭제 연산을 dequeue라고 함.
- 주로 순차적으로 진행되어야 하는 일을 스케줄링 할 때 사용 됨.

큐의 구현체

```
import queue

listQueue = []
For I in range(5):
    listQueue.append(i)

listQueue.pop(0) # 추천하지 않음!!!


normalQueue = queue.Queue()
# normalQueue = queue.Queue(maxsize = 0)
for i in range(5):
    normalQueue.put(i)


while normalQueue.qsize():
    print(normalQueue.get())

if normalQueue.empty():
    print("Queue is empty!!")
```

- Queue를 사용하기 위해서 import 함.
- list.pop(0)을 통해 유사하게 구현 가능하나, 시간복잡도 문제 상 추천하지 않음!!
- Queue 선언.
  매개변수는 maxsize (Queue의 크기. 0 < 일시 무한.)
- Queue의 Enqueue 연산은 put으로,
  Dequeue 연산은 get으로 이루어 짐.
- Queue의 크기를 리턴
- Queue의 크기가 0 일 때 True 리턴

연습문제

Silver 4 - 카드 2

요약

- N장의 카드가 있다. 각각의 카드는 차례로 1부터 N까지의 번호가 붙어 있으며, 1번 카드가 제일 위에, N번 카드가 제일 아래인 상태로 순서대로 카드가 놓여 있다.
- 카드가 한 장 남을 때 까지, 맨 위에 있는 카드를 바닥에 버린다. 그 다음, 제일 위에 있는 카드를 제일 아래에 있는 카드 밑으로 옮기는 행위를 반복합니다.
- N이 주어졌을 때, 제일 마지막에 남는 카드를 구하는 프로그램을 작성하시오.

제약조건

- N의 범위는 1 <= N <= 500,000 이다.

연습문제

Silver 4 - 카드 2

요약

- N장의 카드가 있다. 각각의 카드는 차례로 1부터 N까지의 번호가 붙어 있으며, 1번 카드가 제일 위에, N번 카드가 제일 아래인 상태로 **순서대로 카드가 놓여 있다.**
- 카드가 한 장 남을 때 까지, **맨 위에 있는 카드**를 바닥에 버린다. 그 다음, **제일 위에 있는 카드**를 **제일 아래에 있는 카드** 밑으로 옮기는 행위를 반복합니다.
- N이 주어졌을 때, 제일 마지막에 남는 카드를 구하는 프로그램을 작성하시오.

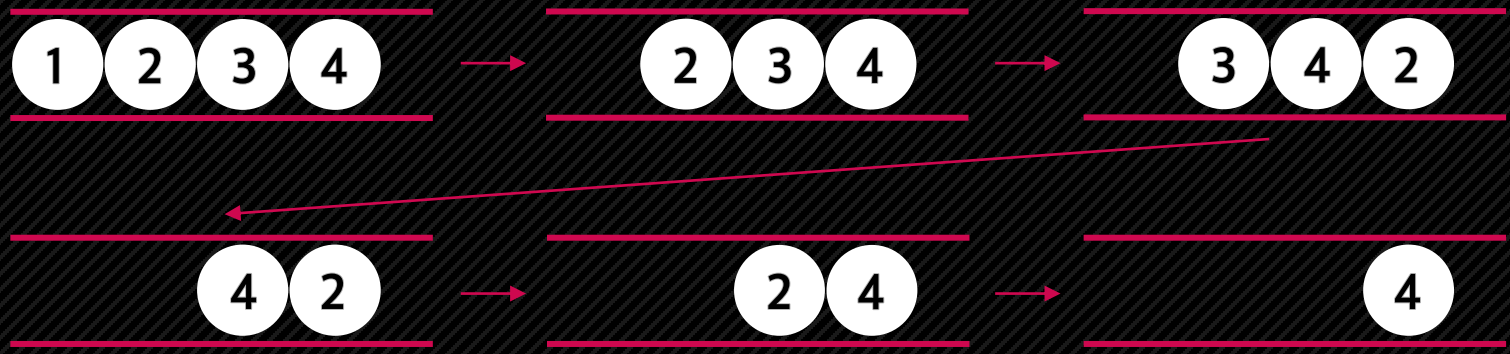
Queue

- FIFO (First In, First Out) 구조를 띄고 있는 자료구조로, 삽입과 삭제 연산이 **서로 다른 한군데에서 발생함.**
- 주로 순차적으로 진행되어야 하는 일을 **스케줄링 할 때** 사용 됨.

연습문제

Silver 4 - 카드 2

N = 4



연습문제

Silver 4 - 카드 2

```
function solution(number) do
    set queue = []

    foreach(i <- 1 ... number) do
        enqueue i to queue
    end

    while loop (size of queue is 1) {
        dequeue of queue
        set anotherNumber = front of queue
        dequeue of queue
        enqueue anotherNumber to queue
    }

    return front of queue

end
```

Dequeue를 두 번 할 때 마다 큐의 길이가 1씩 감소 →
O(2 * N) = O(N)

실습

Silver 4 - 요세푸스 문제 0

요약

- 1번부터 N번까지 N명의 사람이 원을 이루면서 앉아있고, 양의 정수 K(≤ N)가 주어진다.
- 순서대로 K번째 사람을 제거한다. 한 사람이 제거되면 남은 사람들로 이루어진 원을 따라 이 과정을 계속해 나간다. 이 과정은 N명의 사람이 모두 제거될 때까지 계속된다.
- 사람들이 제거되는 순서를 (N, K)-요세푸스 순열이라고 한다. 예로 (7, 3)-요세푸스 순열은 <3, 6, 2, 7, 5, 1, 4>이다.
- 이 때, N, K가 주어지면 (N, K)-요세푸스 순열을 구해라.

제약조건

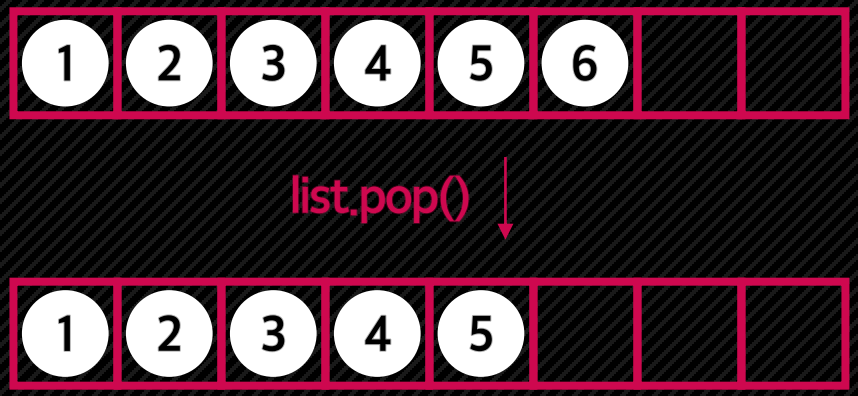
- N과 K의 범위는 1 <= K <= N <= 1,000 이다.

스택

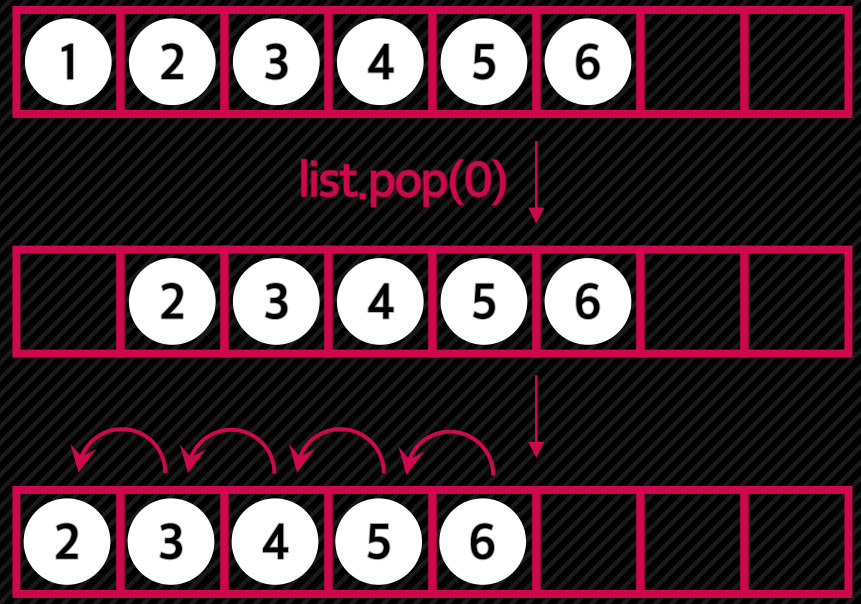
Stack

- FILO (First In, Last Out) 구조를 띄고 있는 자료구조로, 삽입과 삭제 연산이 동일한 한군데에서 발생함.
- 삽입/삭제 연산에 있어 시간복잡도가 O(1)임.
- 이전에 활용한 데이터를 역으로 추적하거나, 처음 들어온 데이터보다 나중에 들어온 데이터가 빨리 나가야 할 때 사용.
- Python에서는 LifoQueue라는 구현체가 있으나, List의 pop을 활용하면 Stack 처럼 사용할 수 있음.

List의 삭제



끝 원소만 삭제 → O(1)



삭제 후 모든 원소 이동 → O(N)

List를 Stack 처럼 사용하는 것은 괜찮지만, Queue처럼 사용하지 말 것!!!

연습문제

Silver 4 - 괄호

요약

- 데이터를 입력 받다가, 중간에 잘못된 데이터가 들어오면 0이 들어온다.
- 0이 들어오면 잘못된 데이터가 들어온 것 이므로, 값을 삭제한다.
- 이후 남아있는 값들의 합을 구하라.

제약조건

- 전체 수의 수는 1 <= K <= 100,000 이다.
- 최종적으로 적어낸 수의 합은 $2^{31}-1$ 보다 같거나 작다.

연습문제

Silver 4 - 괄호

요약

- 데이터를 입력 받다가, 중간에 잘못된 데이터가 들어오면 0이 들어온다.
- 0이 들어오면 잘못된 데이터가 들어온 것 이므로, 값을 삭제한다.
- 이후 남아있는 값들의 합을 구하라.

Stack

- FILO (First In, Last Out) 구조를 띄고 있는 자료구조로, 삽입과 삭제 연산이 동일한 한군데에서 발생함.
- 삽입/삭제 연산에 있어 시간복잡도가 O(1)임.
- 이전에 활용한 데이터를 역으로 추적하거나, 처음 들어온 데이터보다 나중에 들어온 데이터가 빨리 나가야 할 때 사용.

연습문제

Silver 4 - 괄호

```
function solution(number) do
    set stack = []

    foreach (numberElement in number) do
        if numberElement != 0 do
            push numberElement to stack
        end else do
            pop numberElement to stack
        end
    end

    set answer = 0

    while loop(stack is not empty) {
        answer += top of stack
        pop element of stack
    }

    return answer
end
```

숫자를 순차적으로 push하고, 순차적으로 pop함. →
O(2 * N) = O(N)

실습

Silver 4 - 괄호

요약

• 괄호 문자열(Parenthesis String, PS)은 두 개의 괄호 기호인 '(' 와 ')' 만으로 구성되어 있는 문자열이다. 그 중에서 괄호의 모양이 바르게 구성된 문자열을 올바른 괄호 문자열(Valid PS, VPS)이라고 부른다.

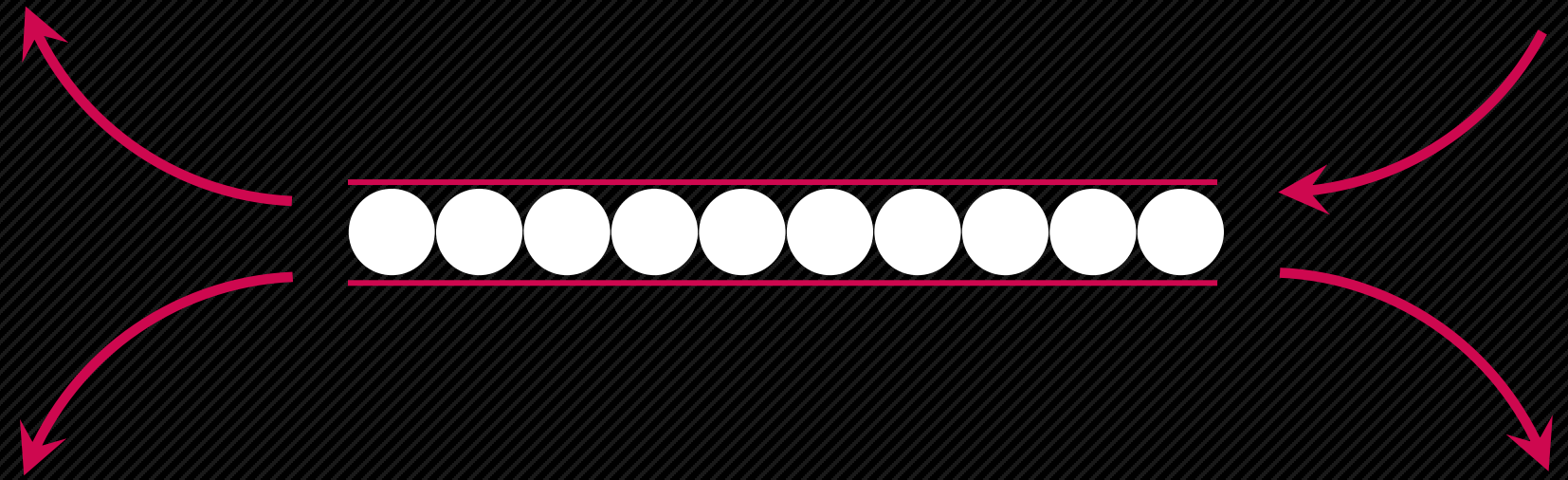
• 예를 들어 "(())()"와 "((()))" 는 VPS 이지만 "(()(", "(())())) , 그리고 "(()" 는 모두 VPS 가 아닌 문자열이다.

• 주어진 괄호 문자열이 VPS인지 아닌지 판단하는 프로그램을 작성하시오.

제약조건

• 하나의 괄호 문자열의 길이는 2 이상 50 이하이다.

덱



Deque

- Queue/Stack의 구조를 합친 구조로, 양쪽에서 삽입/삭제를 모두 할 수 있음.
- 삽입/삭제 연산에 있어 시간복잡도가 O(1)임.

큐 VS 덱

```python
import queue

normalQueue = queue.Queue()
for i in range(5):
    normalQueue.put(i)

while normalQueue.qsize():
    print(normalQueue.get())

if normalQueue.empty():
    print("Queue is empty!!")
```

```python
from collections import deque

dq = deque("1234")

dq.append("5")
# dq.appendleft("0")

dq.rotate(1)

while len(dq):
    print(dq[0])
    dq.popleft()
```

- Deque는 iterable한 데이터를 기반으로 선언 가능하며, 왼쪽에서 진행하는 연산은 left를 붙임.
- Queue는 멀티 스레드 환경에 최적화 되었으며, Deque의 경우 속도 측면에서 더 빠름.
- 코딩테스트 환경에서는 Deque를 쓰는 것을 권장.

추가 추천 문제

Silver 3 - 과제는 끝나지 않아!

- 문제를 보자 마자 어떤 자료구조를 사용해야 하는지 감이 와야 합니다.

Silver 1 - 세훈이의 선물가게

- 구현에 집중된 문제지만, 결국 구현 과정에서 배운 내용을 사용해야 합니다.

Level 2 - 기능개발

- 굳이 자료구조를 사용하지 않아도 풀 수 있지만, 우선은 배운 내용을 연습해봅시다.

Level 2 - 주식가격

- 배운 자료구조들의 크기는 어떤 식으로 확인할까요?

“Any Question?”