

/\* elice \*/

# 실리콘밸리 문제로 배우는 알고리즘 기초

심화된 자료구조



박지나 선생님

# 목차

1. 지난 수업 복습
2. 연결리스트(Linked List)
3. 큐(Queue)
4. 스택(Stack)

# 지난 수업 복습

# 지난 수업 복습

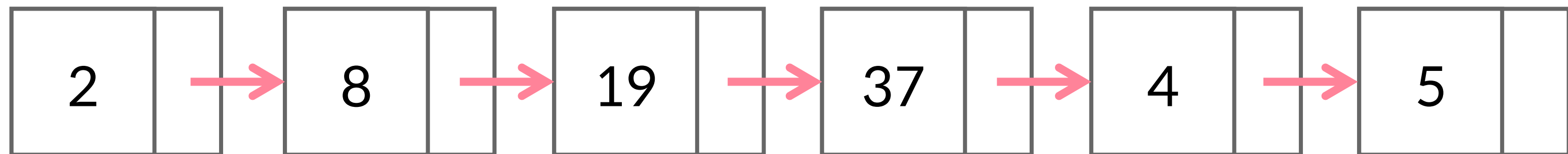
- 1) 시간 복잡도란?
- 2) 공간 복잡도란?
- 3) 해쉬란?
- 4) 배열이란?

# 연결 리스트

# 연결 리스트란?

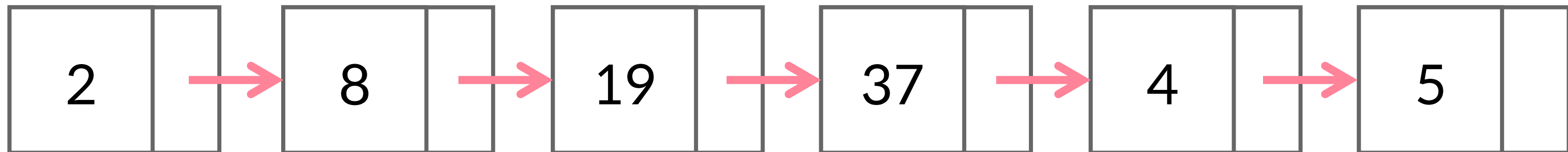
여러개의 **노드**들이 한 줄로 연결

노드 = **저장할 데이터** + **다음 노드로의 연결**



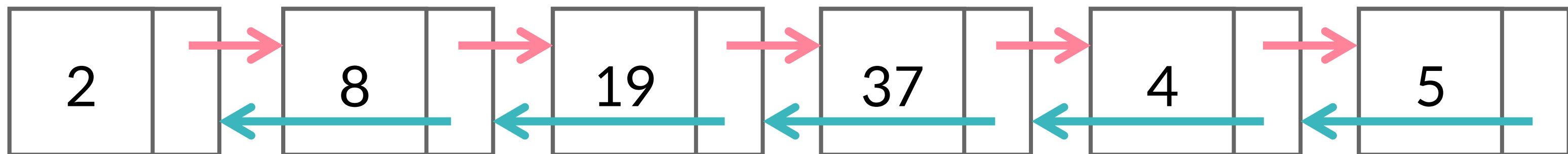
# 단순 연결 리스트

한 방향으로만 이어진 연결 리스트



# 이중 연결 리스트

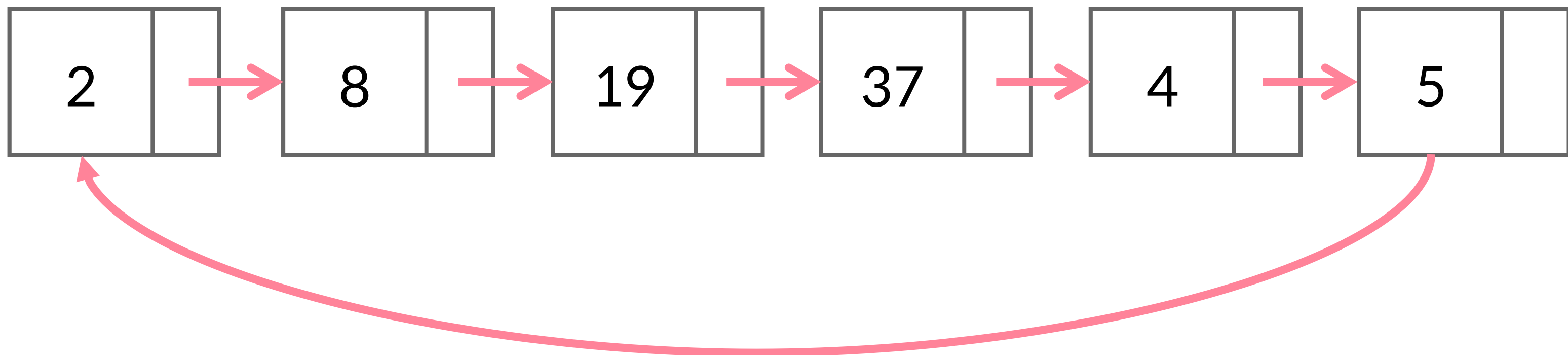
양쪽 방향으로 이어진 연결 리스트





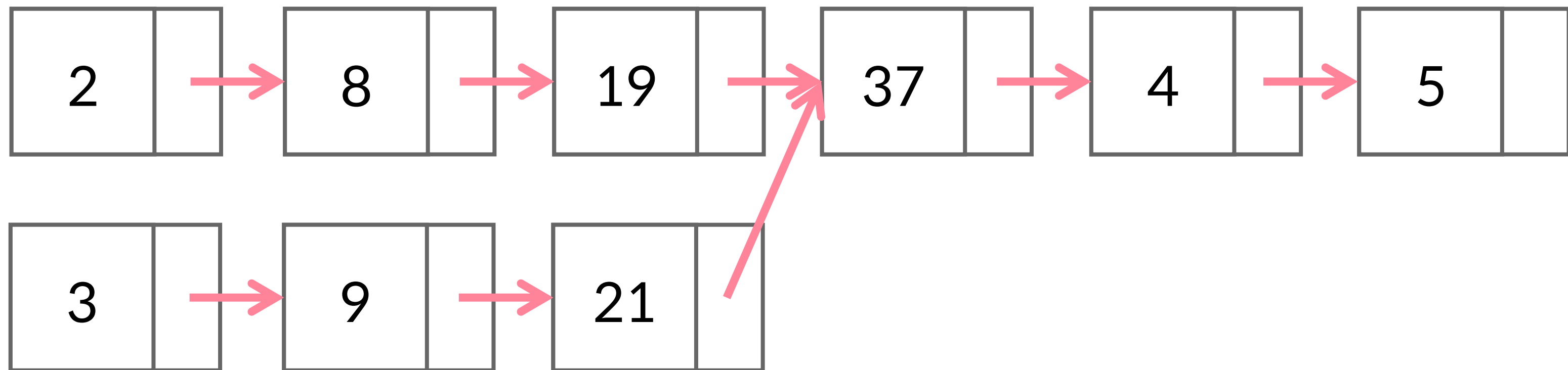
# 원형 연결 리스트

가장 뒤의 노드가 맨 앞의 노드에 연결된 연결 리스트



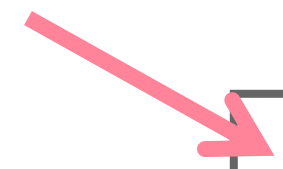
# 기타 연결 리스트

아무 형태의 연결 리스트 모두 가능!



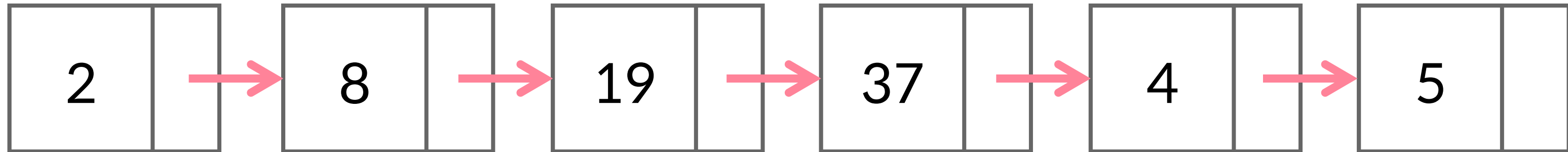
# 배열 VS 연결 리스트

**배열**: 인덱스 이용해서 데이터 접근



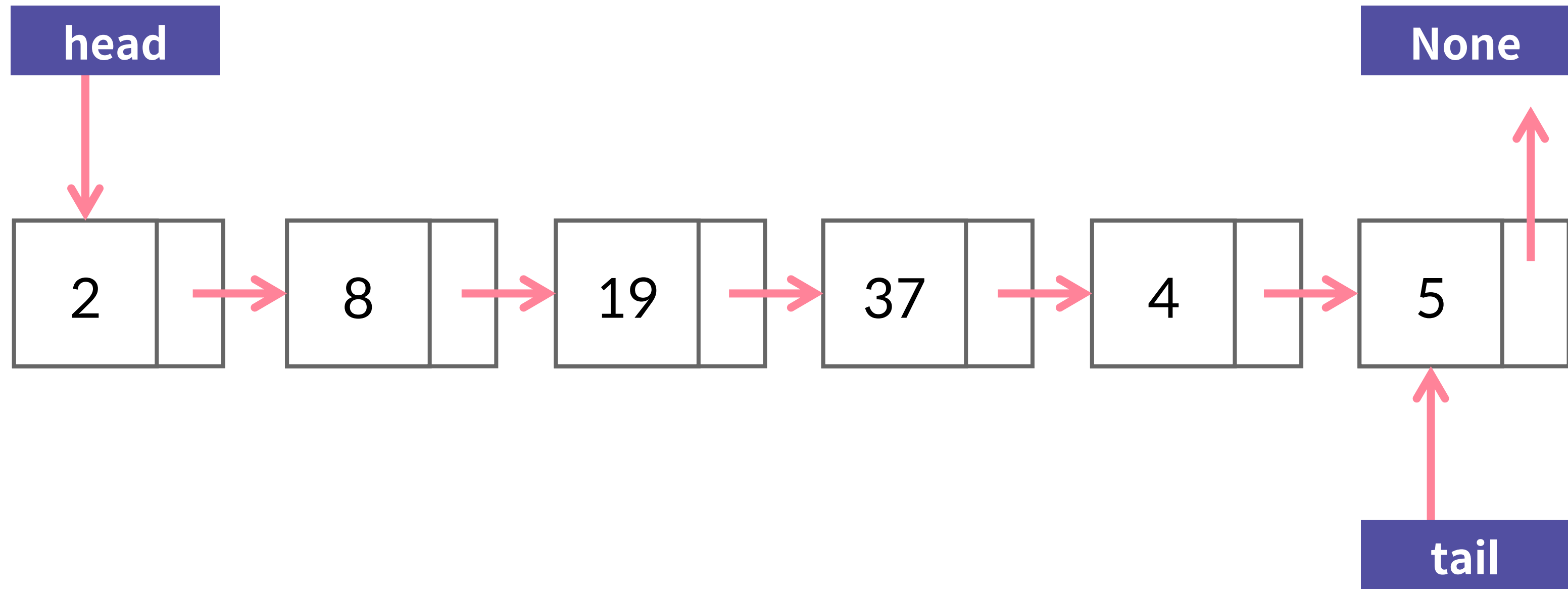
0	1	2	3	4	5
2	8	19	37	4	5

**연결 리스트**: 현재 노드에서 연결된 노드로만



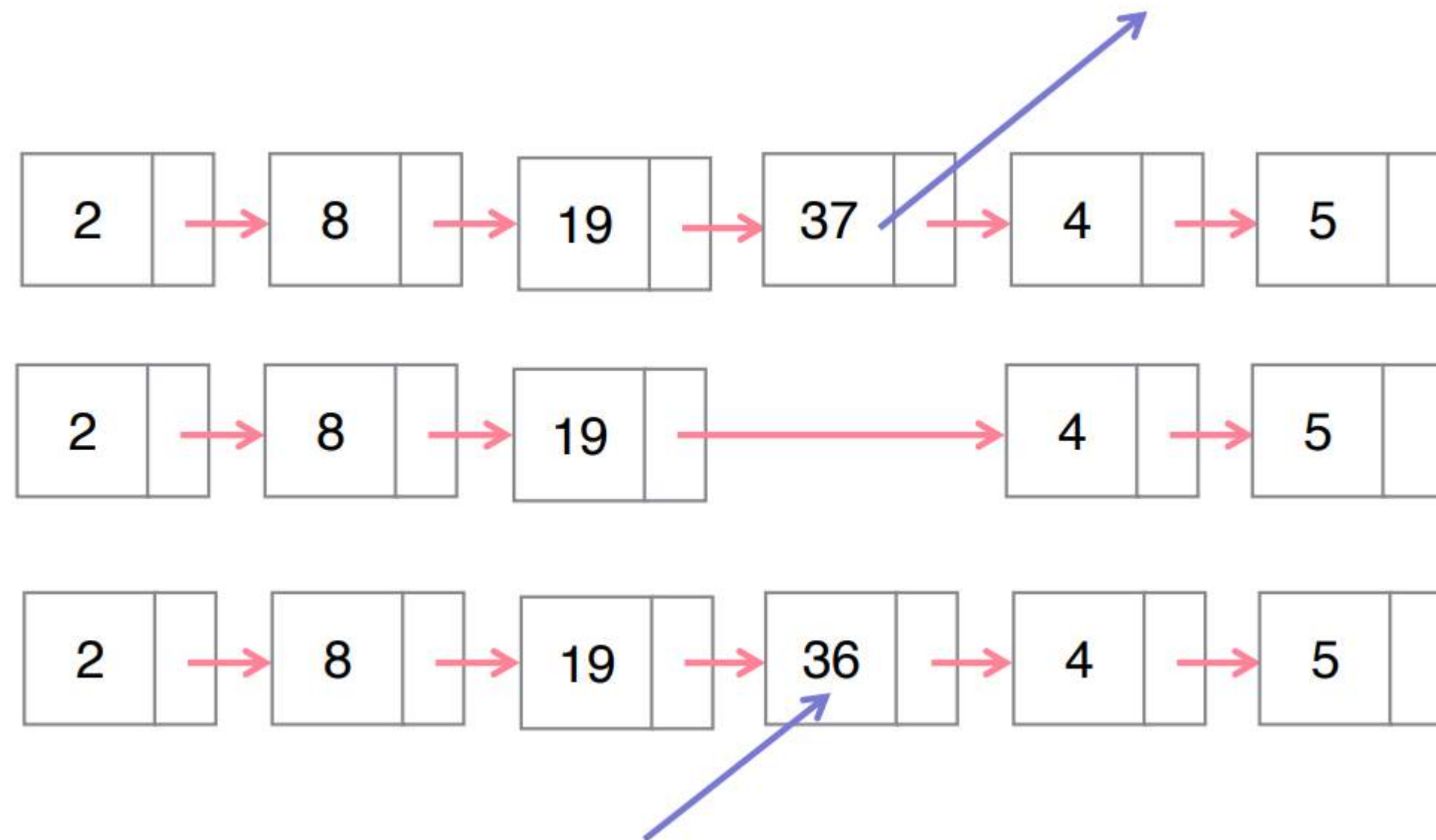
# 헤드 Head

어딘가에서는 시작을 해야하므로!



# 연결 리스트 시간 복잡도

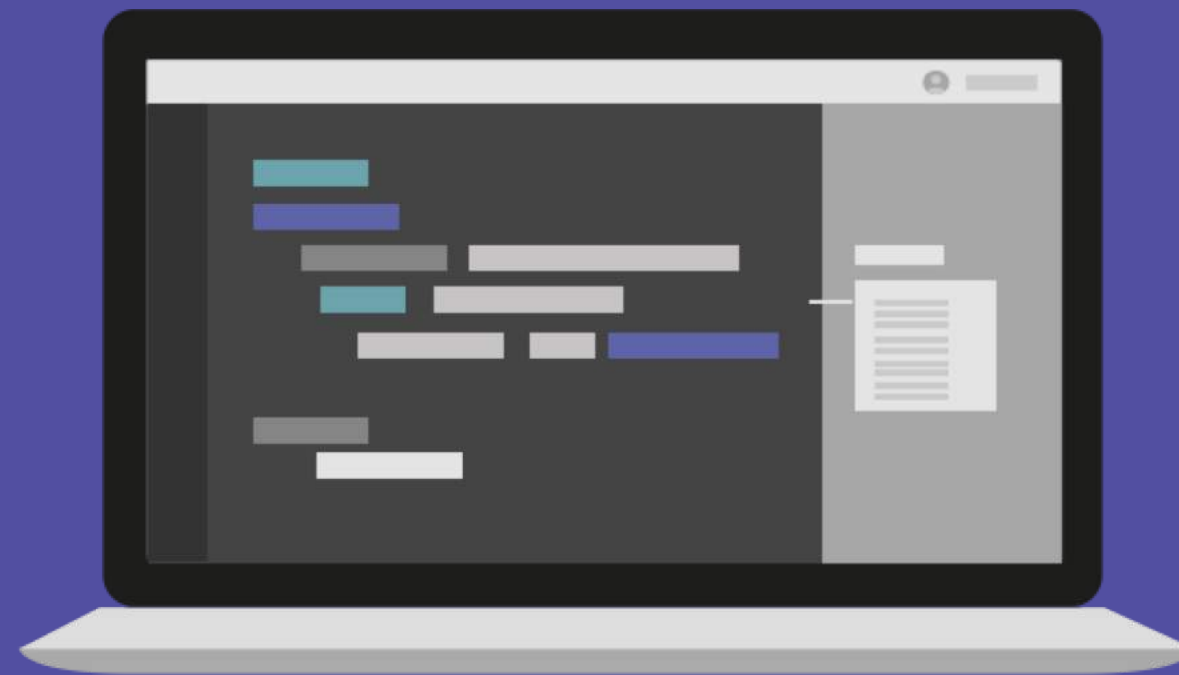
자료 중간에 추가/삭제 :  $O(1)$



배열 시간 복잡도와 비교해보면? : `nums.insert(3, 9)`

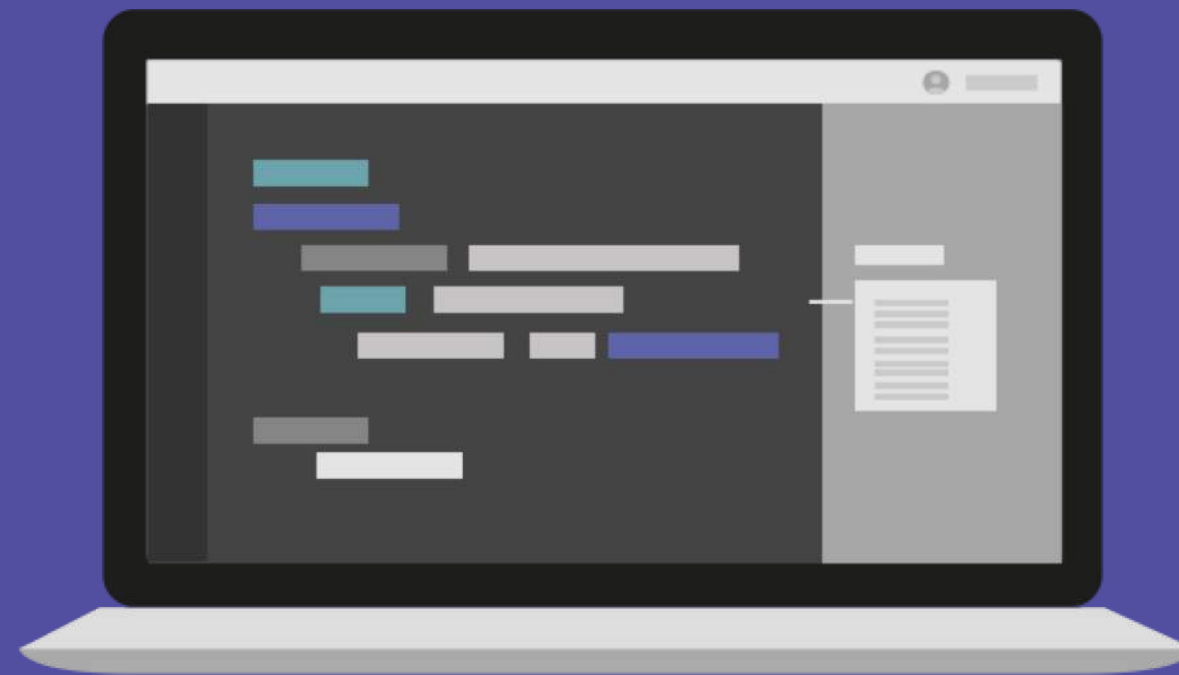
[실습 1]

연결 리스트 <> 배열 변환하기



## [실습 2]

# 연결 리스트에서 노드 삭제

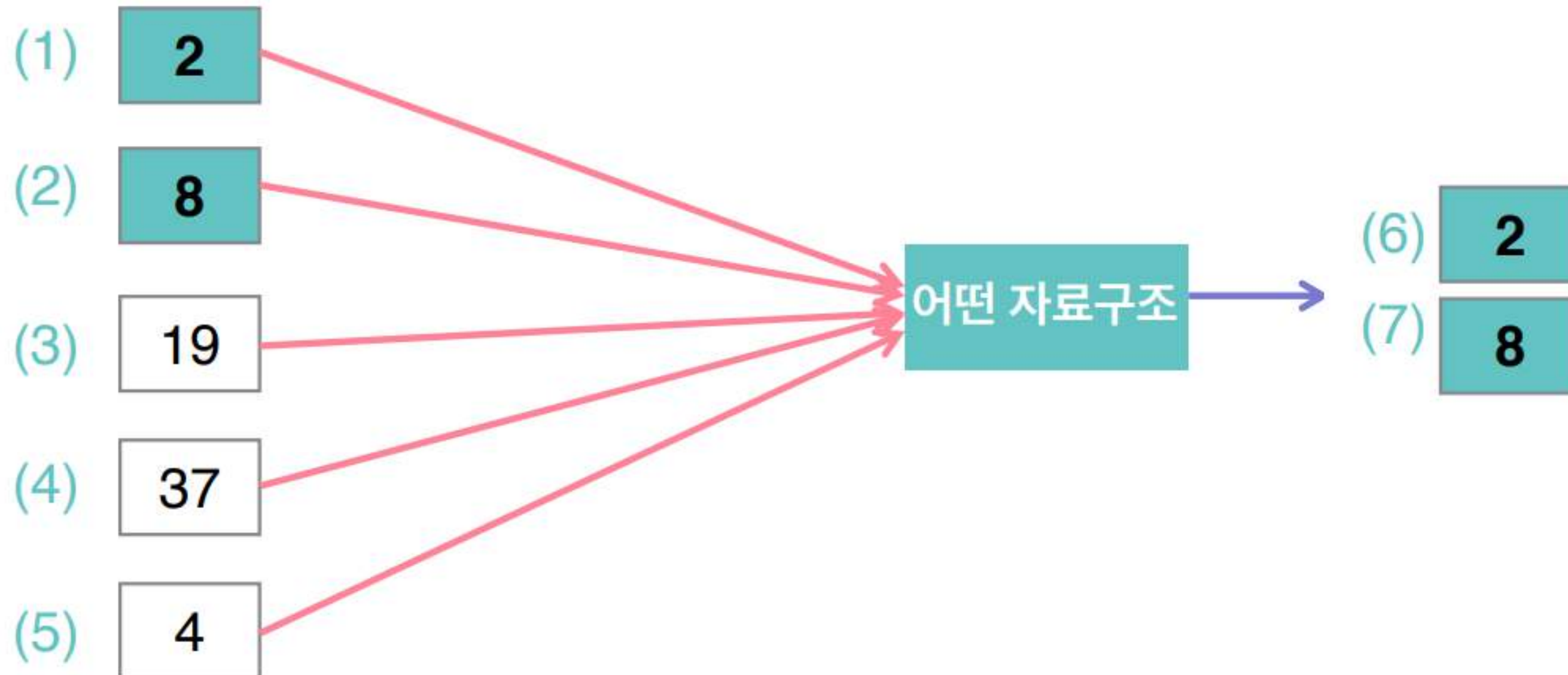


큐



# 큐

먼저 줄 선 사람이 먼저 나간다  
= **FIRST IN FIRST OUT (FIFO)**

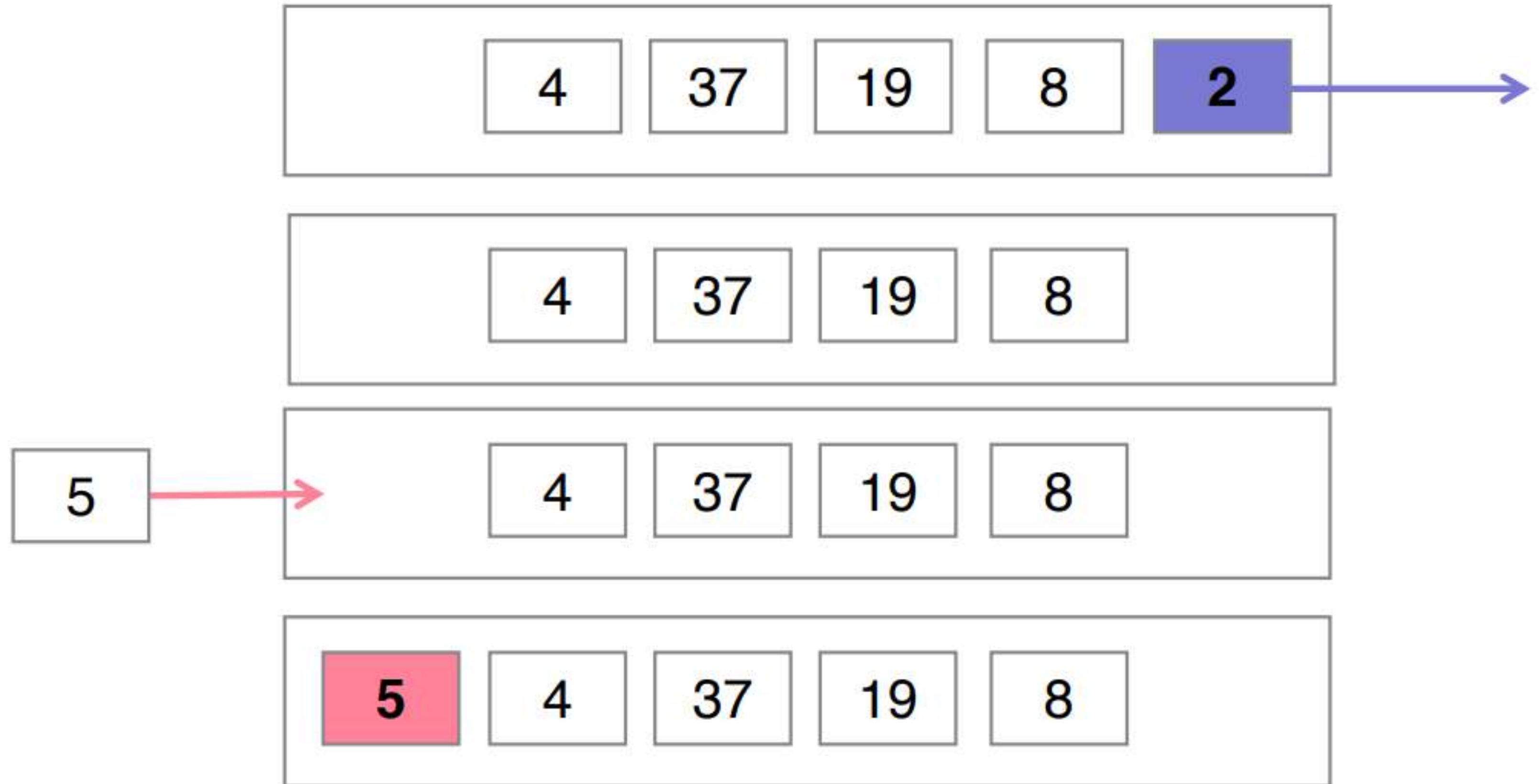


# 큐 시간 복잡도

입력하기 :  $O(1)$

출력하기 :  $O(1)$

# 큐 작동 방식



# 큐 in Python

queue library 활용

```
import queue  
  
q = queue.Queue()  
  
q.put(2)  
  
q.put(8)  
  
q.get()
```

# 큐 in Python

배열을 큐(Queue)로 활용

```
q = [8, 19, 37, 4, 5]
```

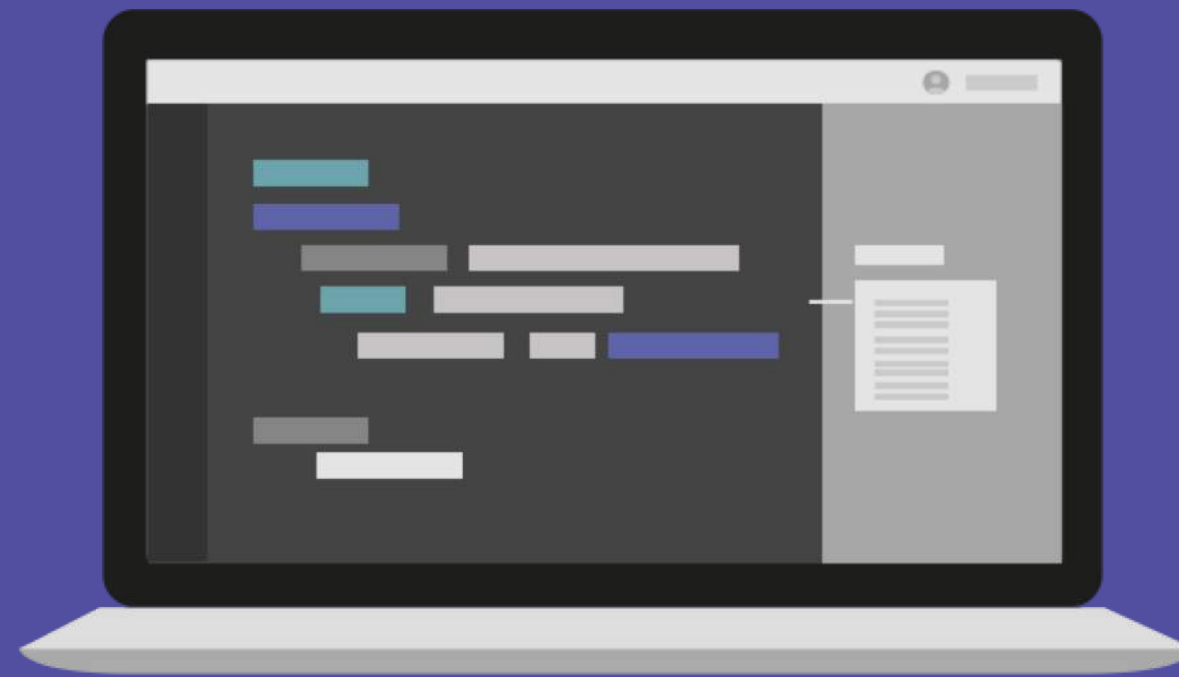
```
q.insert(0, 2) # 맨 앞에 입력한다
```

```
q.pop() # 맨 뒤에서 가져온다
```

배열을 활용했을 때 문제점은? 시간 복잡도!

[실습3]

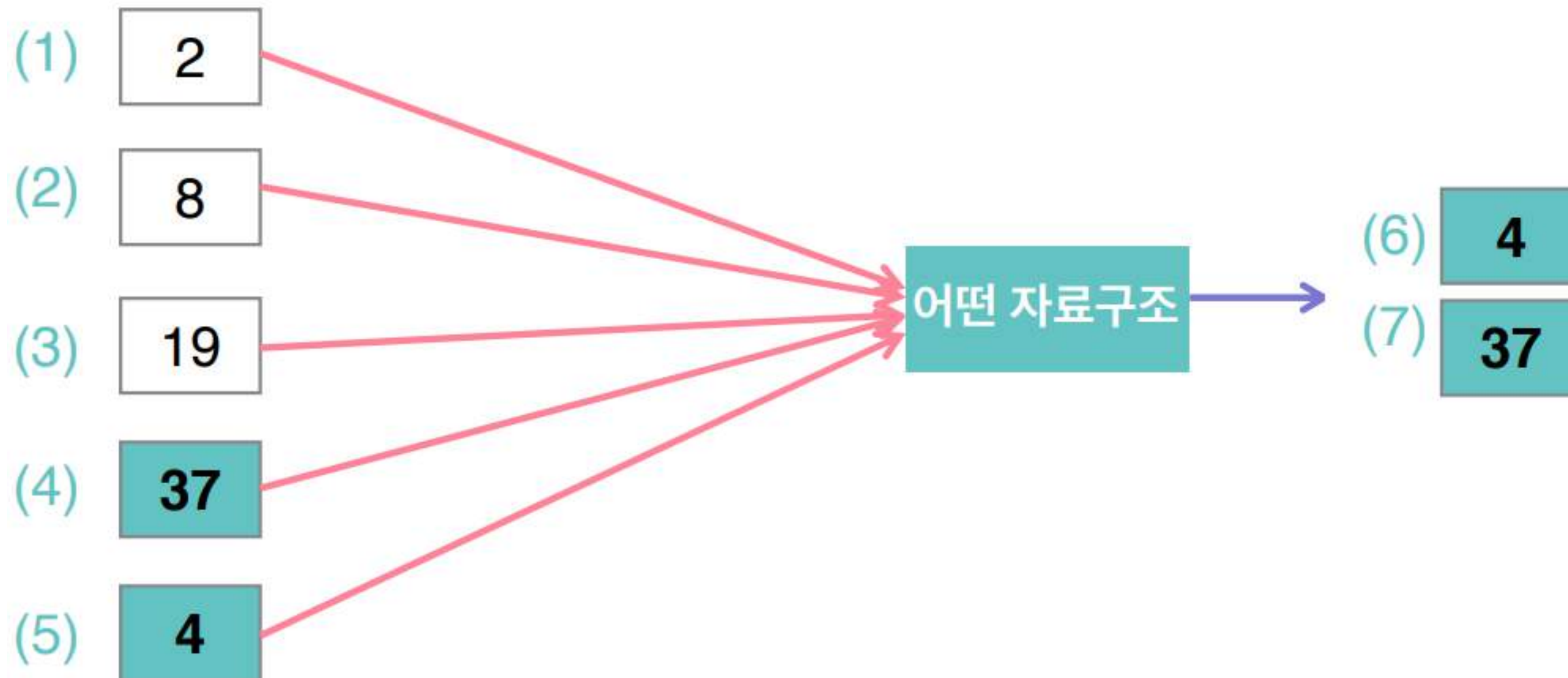
스트리밍 데이터의 이동 평균



스택

# 스택

나중에 줄 선 사람이 먼저 나간다  
= **LAST IN FIRST OUT (LIFO)**



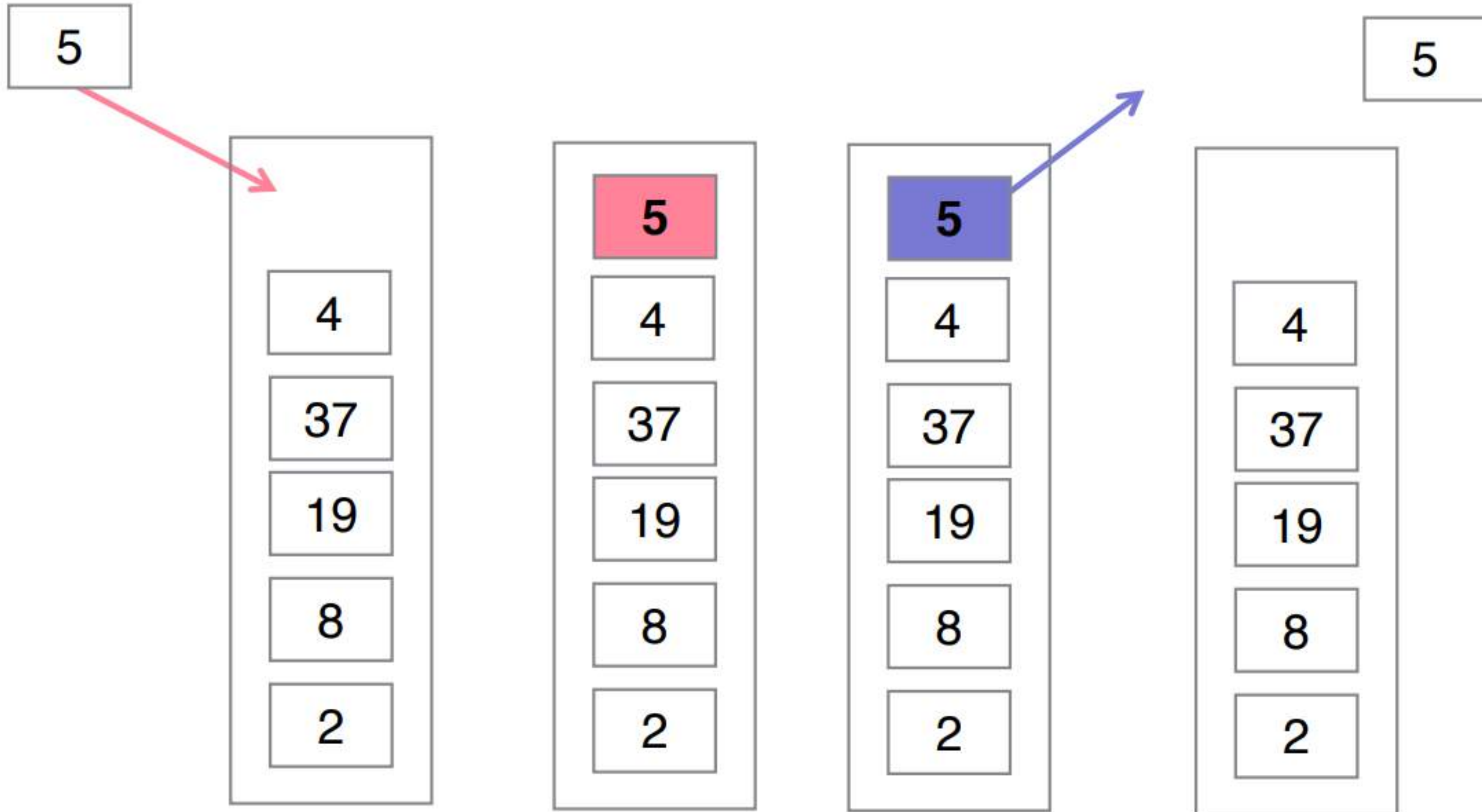


# 스택 시간 복잡도

입력하기 :  $O(1)$

출력하기 :  $O(1)$

# 스택 작동 방식



# 스택 in Python

배열을 스택(Stack)으로 활용

```
Stack = []
```

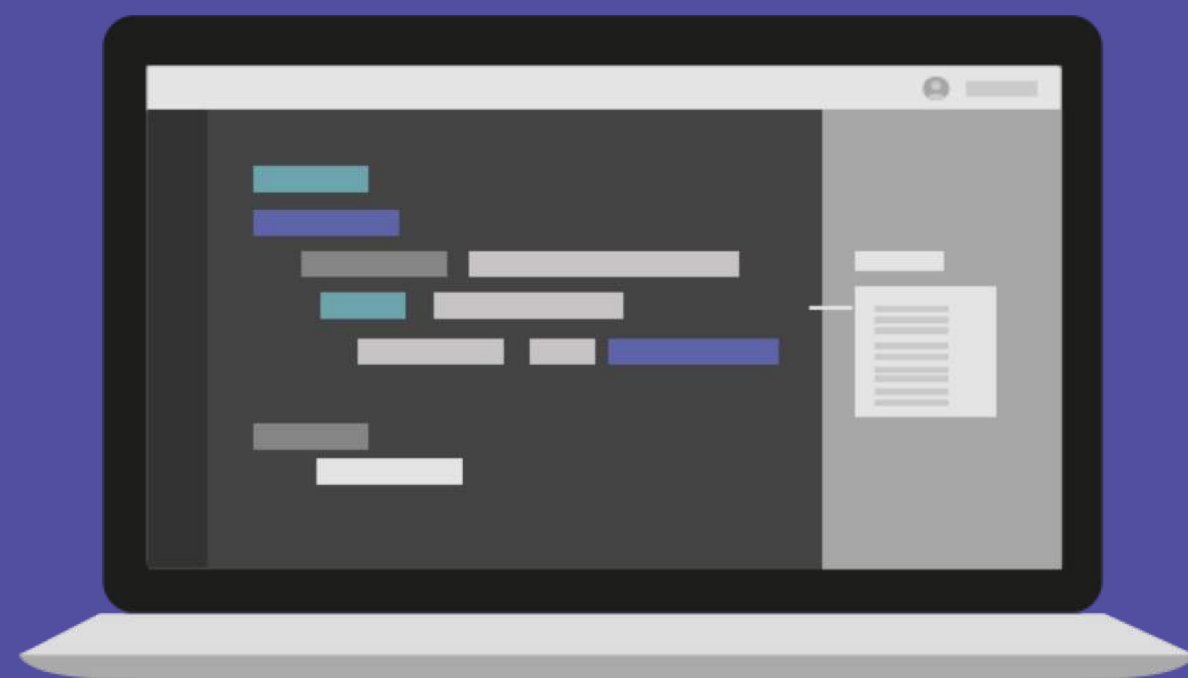
```
Stack.append(2)
```

```
Stack.append(5)
```

```
Stack.pop() #5를 반환
```

# [실습4]

## 괄호 매칭



# 오늘 배운 내용

- 1) 연결 리스트 (Linked List)
- 2) 큐 (Queue)
- 3) 스택 (Stack)

/\* elice \*/

문의 및 연락처

[academy.elice.io](https://academy.elice.io)

[contact@elice.io](mailto:contact@elice.io)

[facebook.com/elice.io](https://facebook.com/elice.io)

[medium.com/elice](https://medium.com/elice)