씨앗 정기 활동

23.09.09

1회차

목차

1. 알고리즘이란?

2. 시간 복잡도 이해하기

3. 알고리즘과 자료구조의 관계

4. 다양한 자료구조 이해하기

5. 자료구조 적용 실습

알고리즘?

"어떤 문제를 해결하기 위한 단계적인 절차"

파스타를 맛있게 만드는 방법, 병천에서 홍대까지 가는 방법,…
어떤 숫자 N을 배열에서 찾는 방법,…

병천에서 홍대까지 가는 방법

1. 택시를 타고 홍대까지 가는 방법
2. 셔틀버스를 타고 교대역에서 내린 후 택시를 타는 방법
3. 청주공항까지 시외버스를 타고, 김포공항까지 비행기를 타고, 공항철도를 이용해서 홍대입구역으로 가는 방법

몇 번이 가장 좋은 방법일까?

장단점을 이해해야 한다.

철수는 돈이 매우 많고 차 타기를 좋아한다.
영희는 만원 밖에 없다.
민호는 차 멀미를 심하게 앓는다.

프로그램이나 서비스도 마찬가지!
속도, 용량, 보안, 편리성 등등,, **무엇에 중점을 두느냐**에 따라
구현방식, 즉 알고리즘이 달라진다.

컴퓨터의 자원?

컴퓨터의 자원 = 시간(속도) + 공간(용량)

메모리의 성능이 발전함에 따라
공간보다는 시간의 중요성이 강조되고 있음.

하지만, 메모리 용량을 제한하는 문제가 출제되거나
하드웨어의 한계로 공간 성능을 고려해야 하는 상황이 존재함

시간 복잡도를 알아보자.

ex) A라는 알고리즘은 동작하는데 3초가 걸립니다 ➔ X

why? 단순히 s, ms 등의 시간 단위로 표현하는 것은 의미 없음
(모든 컴퓨터의 성능, 환경이 다르기 때문)

따라서 알고리즘의 성능을 평가하기 위한 객관적인 지표가 필요함.

시간 복잡도

정의 : 입력 N 의 크기에 따라 실행되는 연산의 수

lim 무한대 를 떠올려보면 쉬워요.

평가방법 = 점근적 표기법 , n이 무한대로 커질 때의 복잡도를 간단히 표현

최상의 경우 : 오메가 표기 (Big-Ω Notation)

평균의 경우 : 세타 표기법 (Big-θ Notation)

=〉 평가하기가 너무 까다롭다.

최악의 경우 : 빅오 표기법 (Big-O Notation)

=〉 최악의 경우에 대해 알고리즘을 평가

최악의 경우란 무엇일까?

어떤 알고리즘이 수행되는데 있어서 최대 연산이 발생하는 입력

A = [9, 8, 7, 6, 5, 4, 3, 2, 1]

case 1. A 에서 10 을 찾기
case 2. A에서 3을 찾기
case 3. 최대값 / 최소값 / 평균 을 구하기

빅오 표기법 (Big-O Notation)

$f(n)$: 알고리즘의 연산횟수를 나타낸 함수

$x > m$ 에서 $kg(n) \geq f(n)$을 만족하는 양수 m, k 가 존재하면,
$f(n) \in O(g(n))$

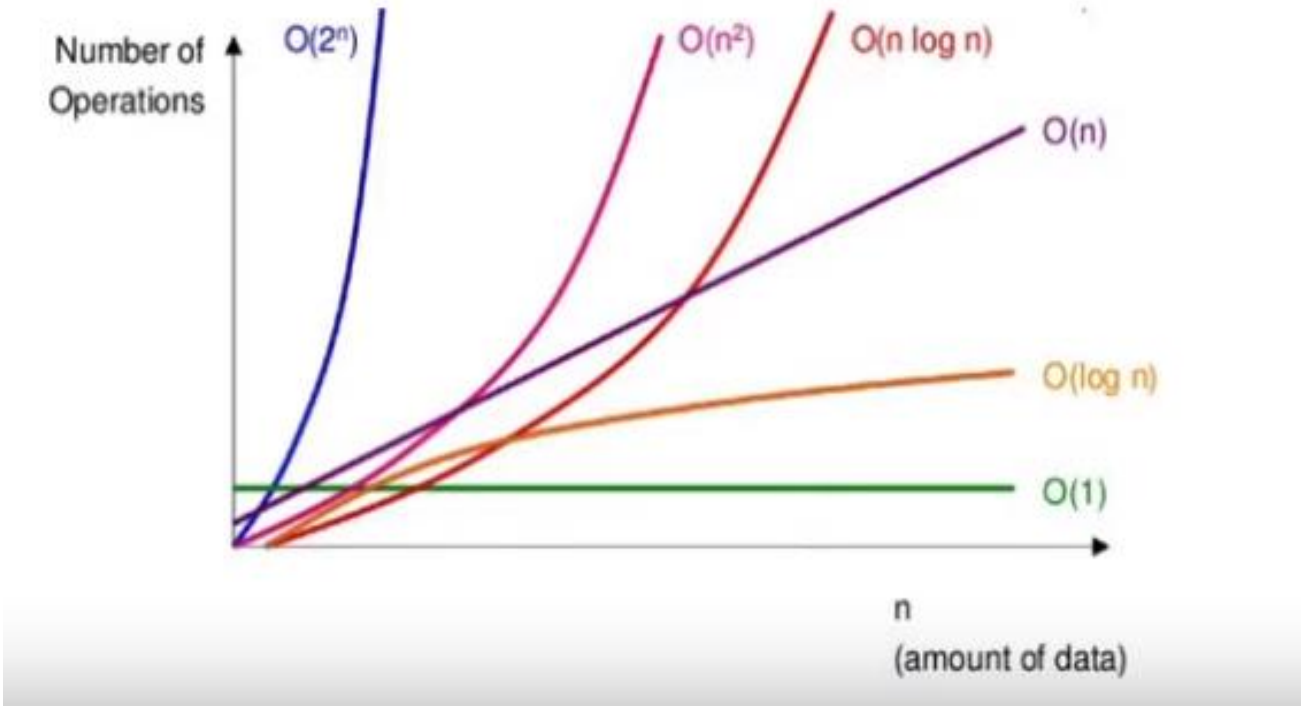
ex)

$f(n) = 3n^2 + 2n + 1 \in O(n^2)$

$g(n) = 300n^4 + 2500n \in O(n^4)$

$h(n) = 2^n + 3^n \in O(3^n)$

$\mathrm{k}(n) = 31251 \in O(1)$



자료구조가 뭘까?

자료를 처리(탐색, 수정, 삭제)하기 위해
컴퓨터가 잘 다룰 수 있는 형태로
자료들을 정리하고 조직화하는 구조

즉, 자료를 저장하는 공간
ex) 배열, 그래프, 트리 등등

짱구의 장난감 상자

짱구 :
가장 무거운 장난감부터 먼저 꺼내고 싶어

철수 :
난 미미인형을 꺼내고 싶어

훈이 :
난 아무거나 좋아!

각자의 필요를 충족할 수 있는 자료구조가 필요하다!

다시 컴퓨터로 돌아와서

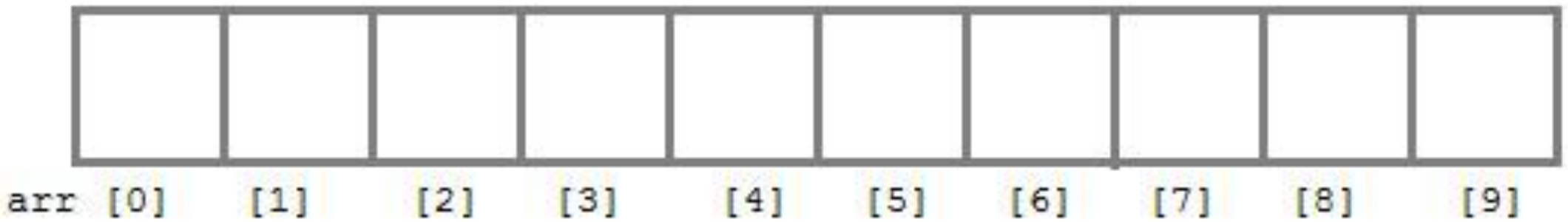
우리는 그동안 **배열**(리스트)라는 자료구조를 사용했습니다.

배열은 **여러 개의 칸이 이어져 있는** 1자 형태의 자료구조 입니다.
각 칸에 Index라는 번호를 붙여 놓고 값을 넣고, 지우고, 바꾸면서
편리하게 사용해왔습니다.

이제 이 편리한 자료구조를 한번 뜯어서 살펴볼까요?

배열(리스트)의 성능 알아보기.

1. 삽입　　2. 삭제　　3. 탐색



`len(a)`	$O(1)$	전체 요소의 개수를 리턴한다.
`a[i]`	$O(1)$	인덱스 i의 요소를 가져온다.
`a[i:j]`	$O(k)$	i부터 j까지 슬라이스의 길이만큼 k개의 요소를 가져온다. 이 경우 객체 k개에 대한 조회가 필요하므로 $O(k)$이다.
`elem in a`	$O(n)$	`elem` 요소가 존재하는지 확인한다. 처음부터 순차 탐색하므로 n만큼 시간이 소요된다.
`a.count(elem)`	$O(n)$	`elem` 요소의 개수를 리턴한다.

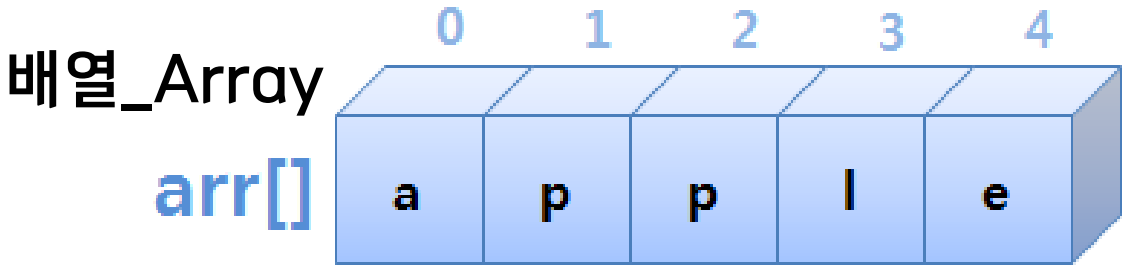
`a.append(elem)`	$O(1)$	리스트 마지막에 `elem` 요소를 추가한다.
`a.pop()`	$O(1)$	리스트 마지막 요소를 추출한다. 스택의 연산이다.
`a.pop(0)`	$O(n)$	리스트 첫번째 요소를 추출한다. 큐의 연산이다. 이 경우 전체 복사가 필요하므로 $O(n)$이다.
`del a[i]`	$O(n)$	`i`에 따라 다르다. 최악의 경우 $O(n)$이다.

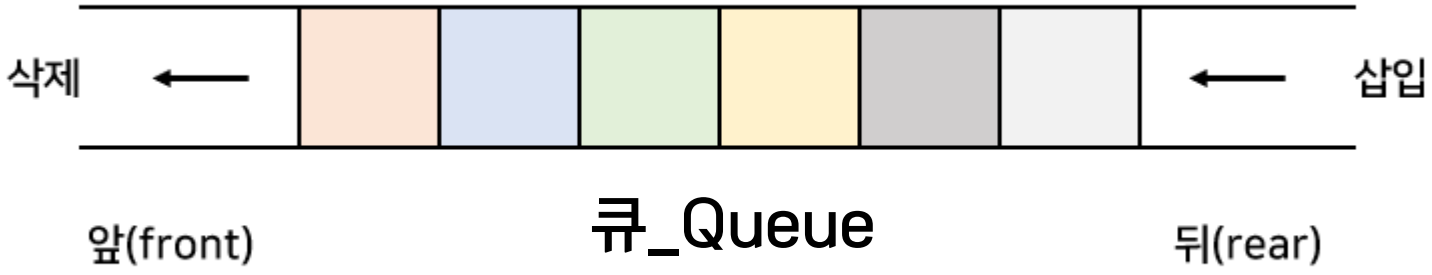
자료구조의 학습 영역

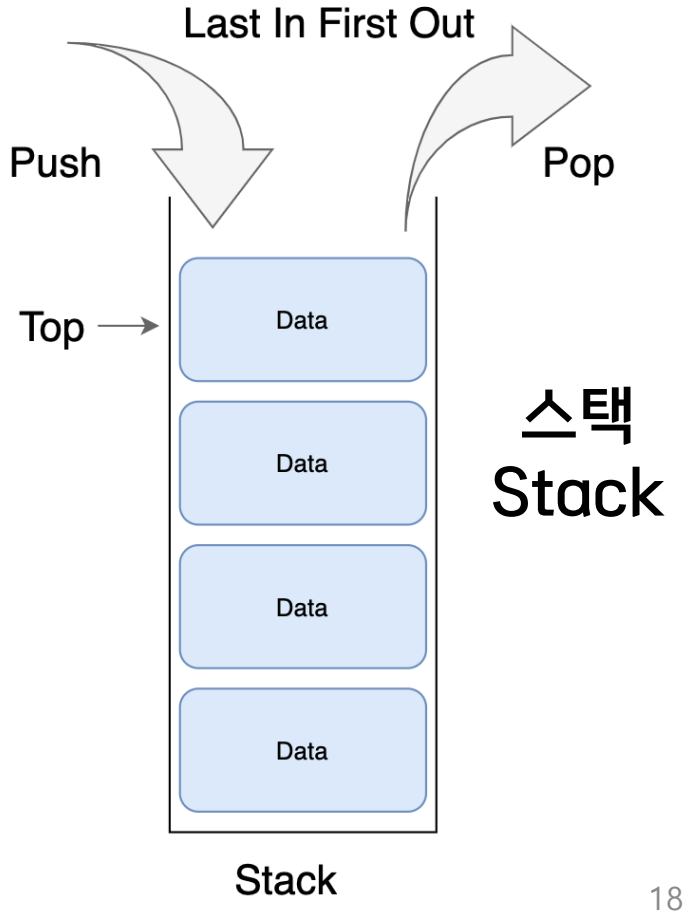
이미지 출처 : https://velog.io/@shinhojung814/WEEK-13-%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0

선형 자료구조

데이터를 순서대로 나열해서 저장하는 자료구조

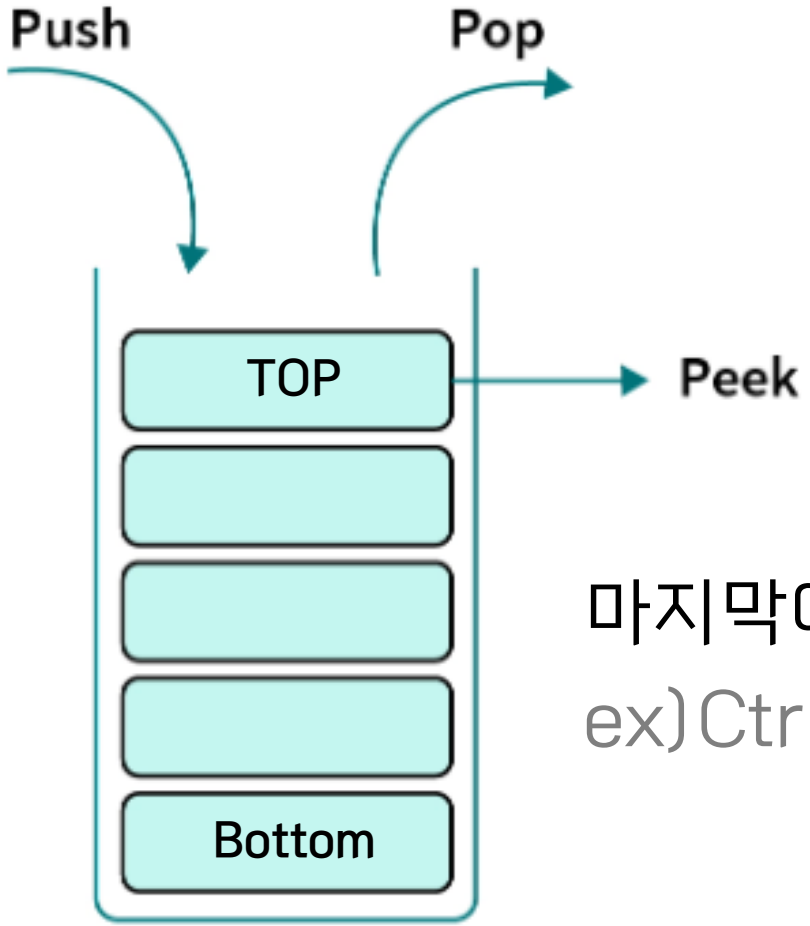






스택_Stack

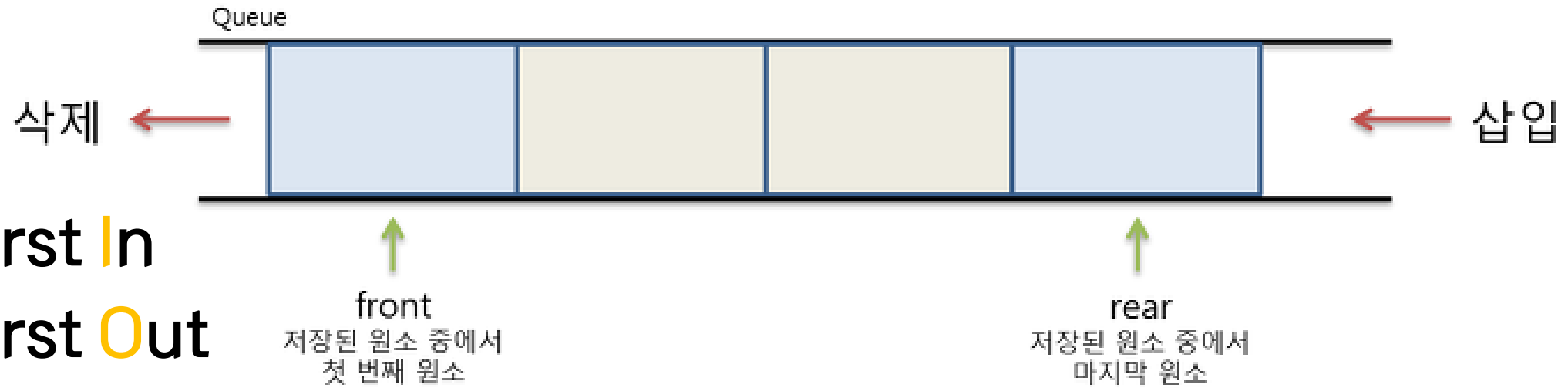
Last In
First Out
후입 선출



마지막에 들어온 것 부터 처리하는 서비스
ex) Ctrl+Z, 인터넷 뒤로가기

이미지 출처 :
https://www.scaler.com
/topics/stack-in-c/

큐_Queue



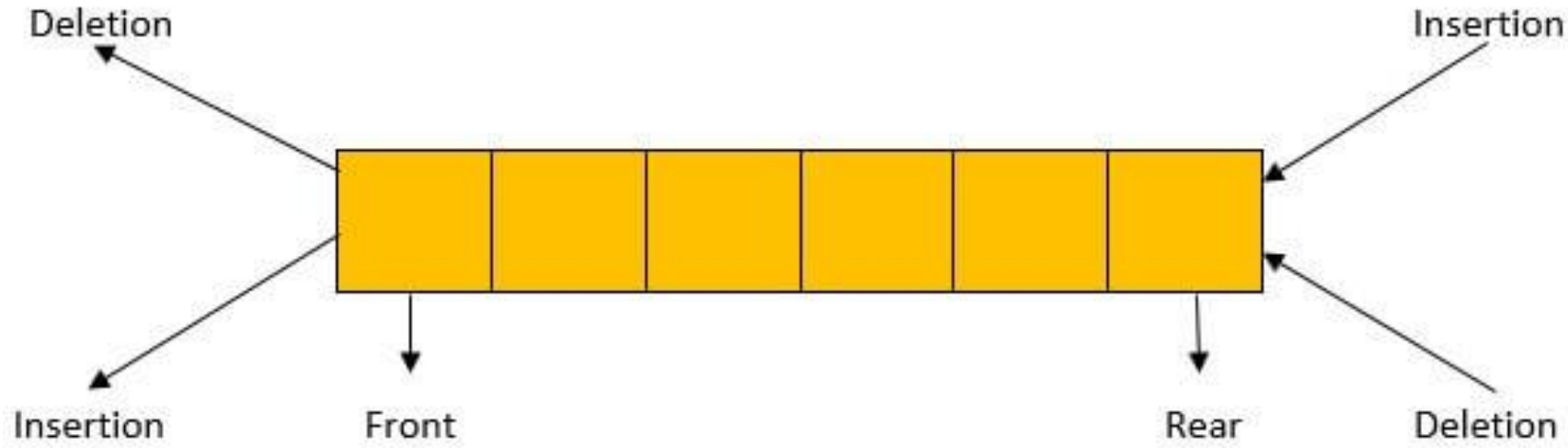
First In
First Out
선입 선출

먼저 들어온 것부터 처리하는 서비스

ex) 프린터기, 은행창구

이미지 출처 :
https://songeunjun
g92.tistory.com/23

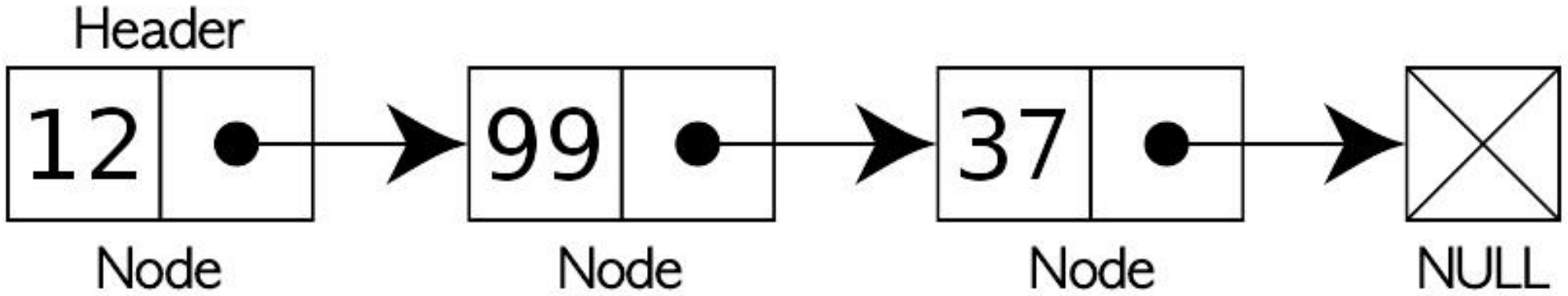
덱_Deque



양쪽에서 넣었다, 뺐다 할 수 있는 자료구조

이미지 출처 :
https://jjudrgn.tistory.com/15

연결 리스트의 노드_Node



노드 = 데이터 + 링크(다음 노드의 주소)

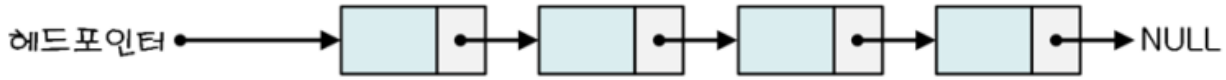
마지막 노드는 Null을 가리킴

이미지 출처 :
https://lamarr.dev/datastructure/2020/04/02/01-linked-list.html

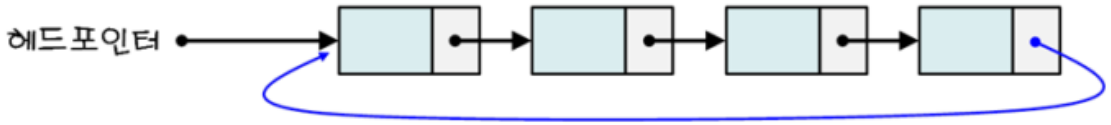
연결 리스트

리스트의 종류

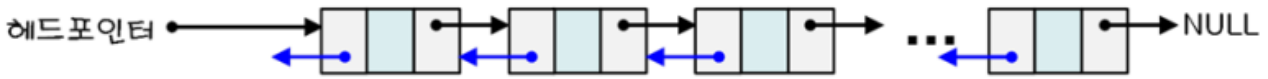
- 단순연결 리스트
  (Singly Linked List)



- 원형 연결 리스트
  (Circular Linked List)



- 이중 연결 리스트
  (doubly Linked List)

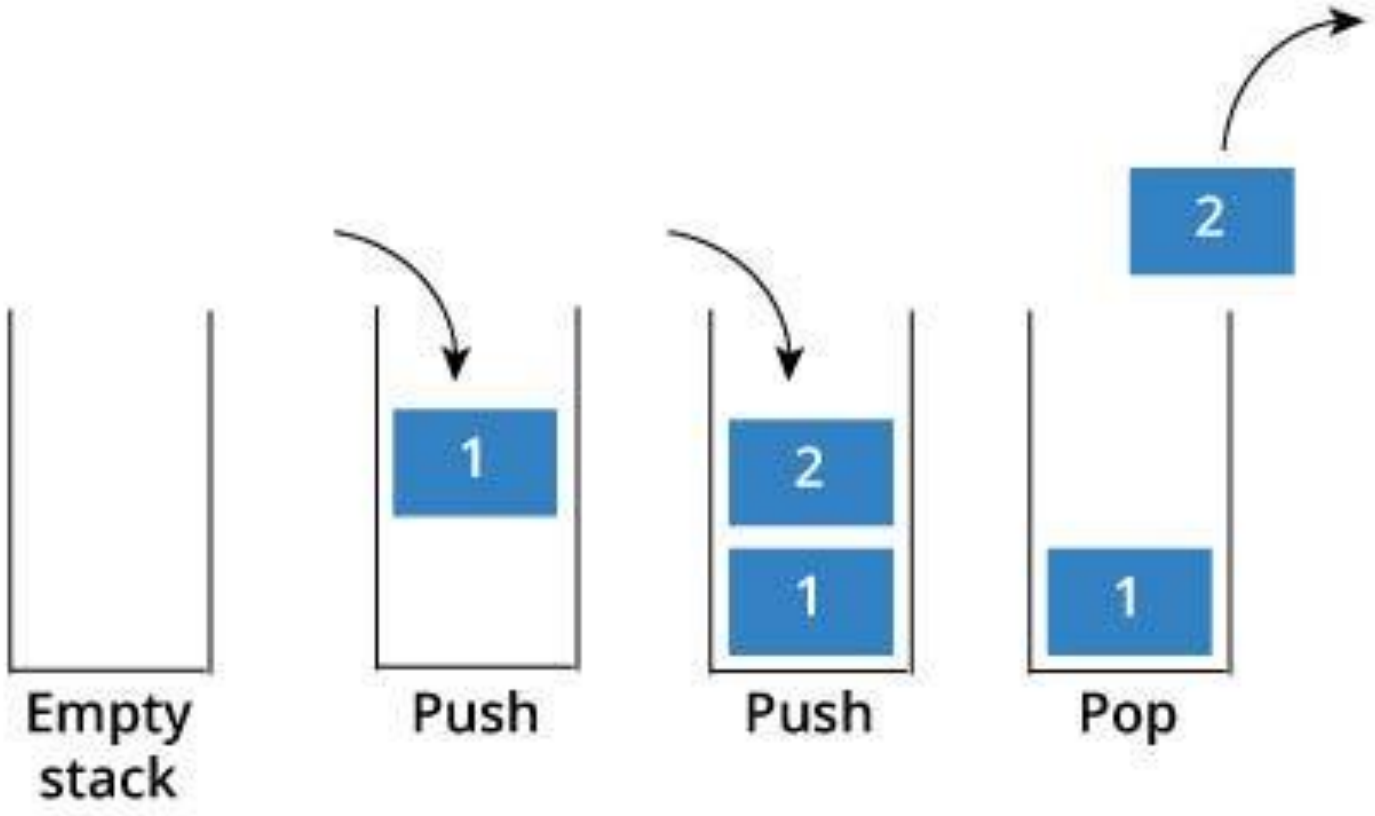


자료구조의 구현

사실 대부분의 자료구조는 모든 프로그래밍 언어에서
라이브러리, 모듈 등을 통해 사용 가능하므로
실제 프로그래밍 중에는 구현할 필요가 없음

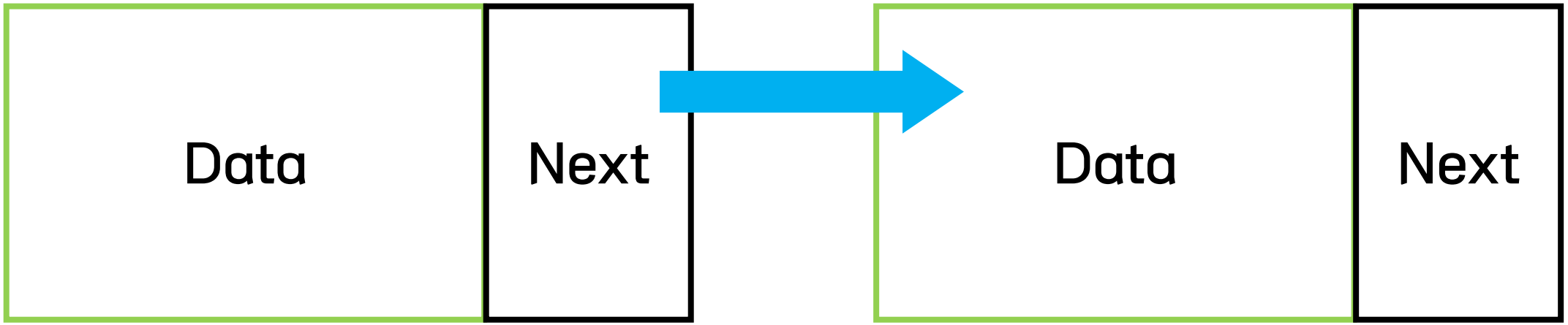
하지만, 알고리즘과 구현방식을 이해할 필요는 있음.
(성적, 프로그래밍 실력 향상, CS면접 대비 등등)

리스트를 이용한 스택 구현



이미지 출처 :
https://medium.com/@songjaeyoung92/%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0-javascript-stack-%EC%9D%B4%EB%9E%80-31f9bbb84897

노드 클래스 구현

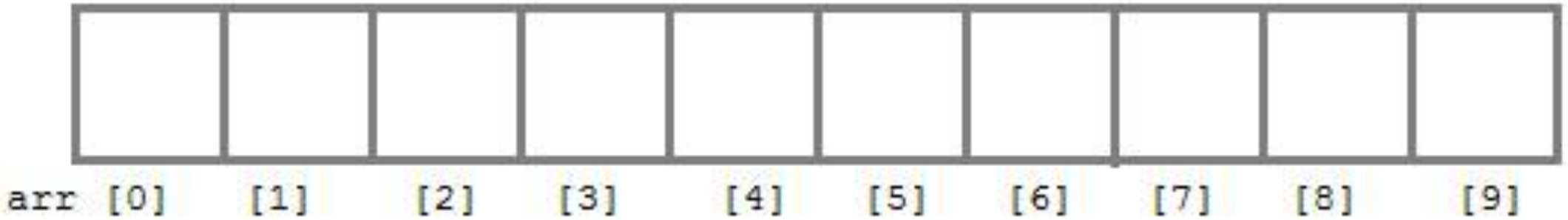


단순 연결 리스트 구현

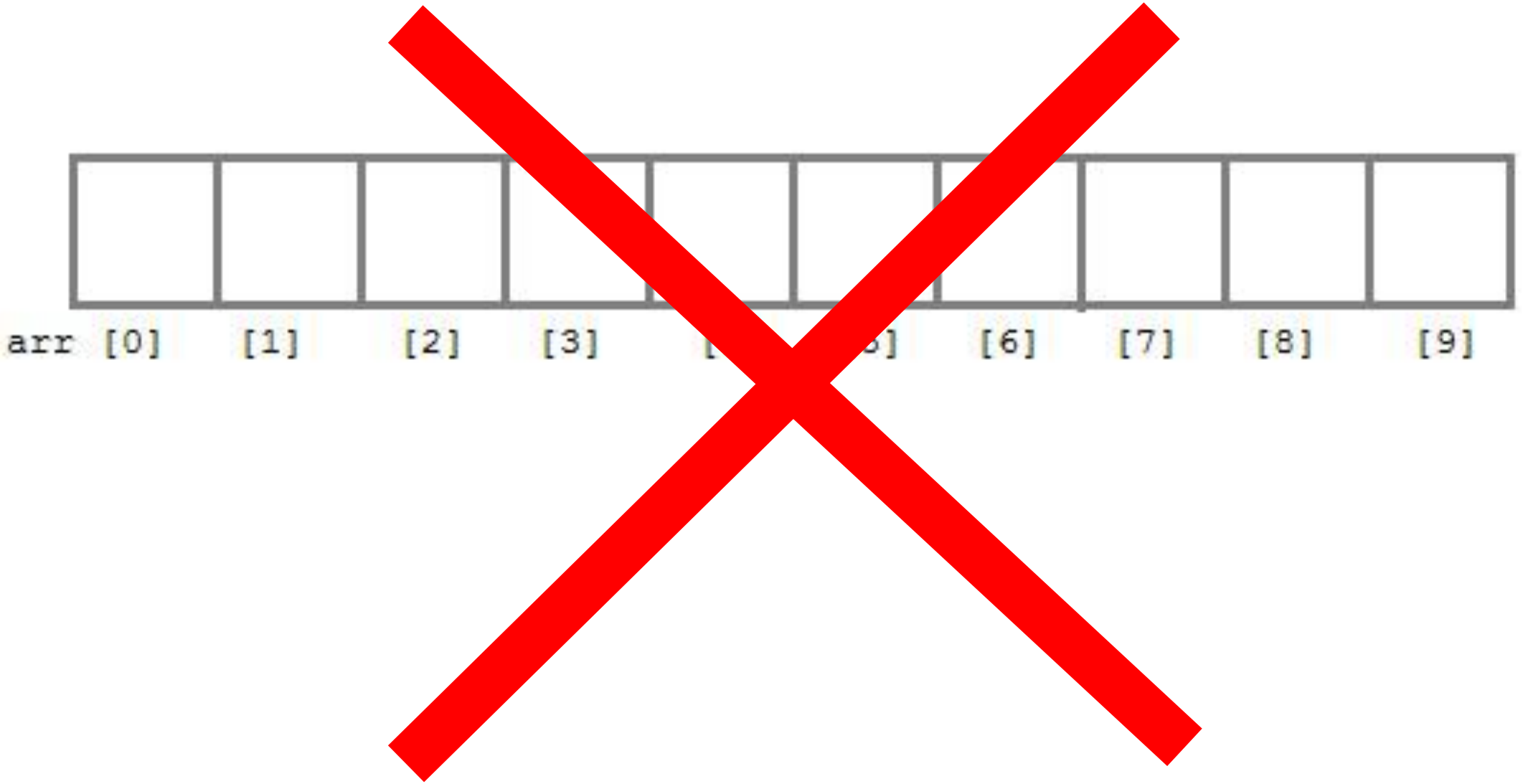
큐, 덱 구현?

1. 리스트로 구현하는 큐
2. 연결 리스트로 구현하는 큐
3. 원형 연결 리스트로 구현하는 큐
4. 덱 ?

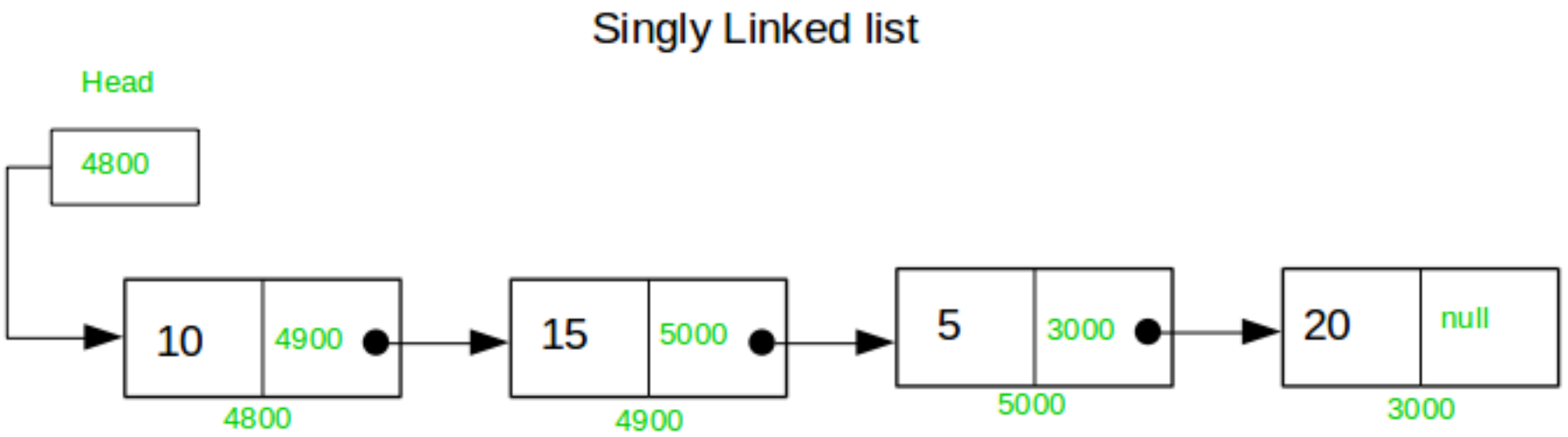
1. 리스트로 구현하는 큐



1. 리스트로 구현하는 큐

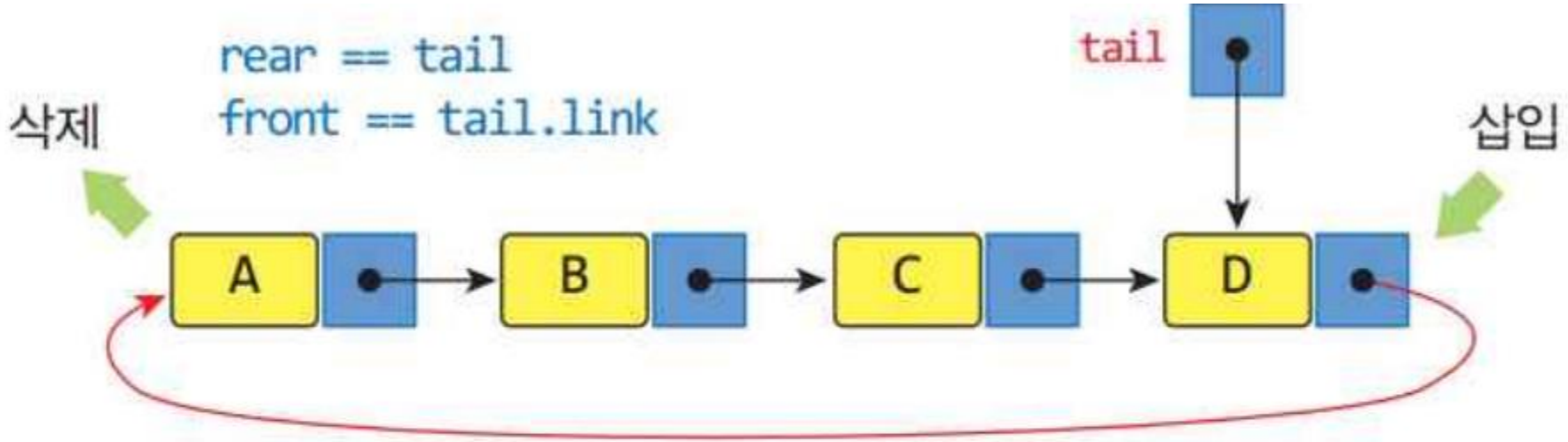


2. 단순 연결 리스트로 구현하는 큐



뭐가 문제일까?

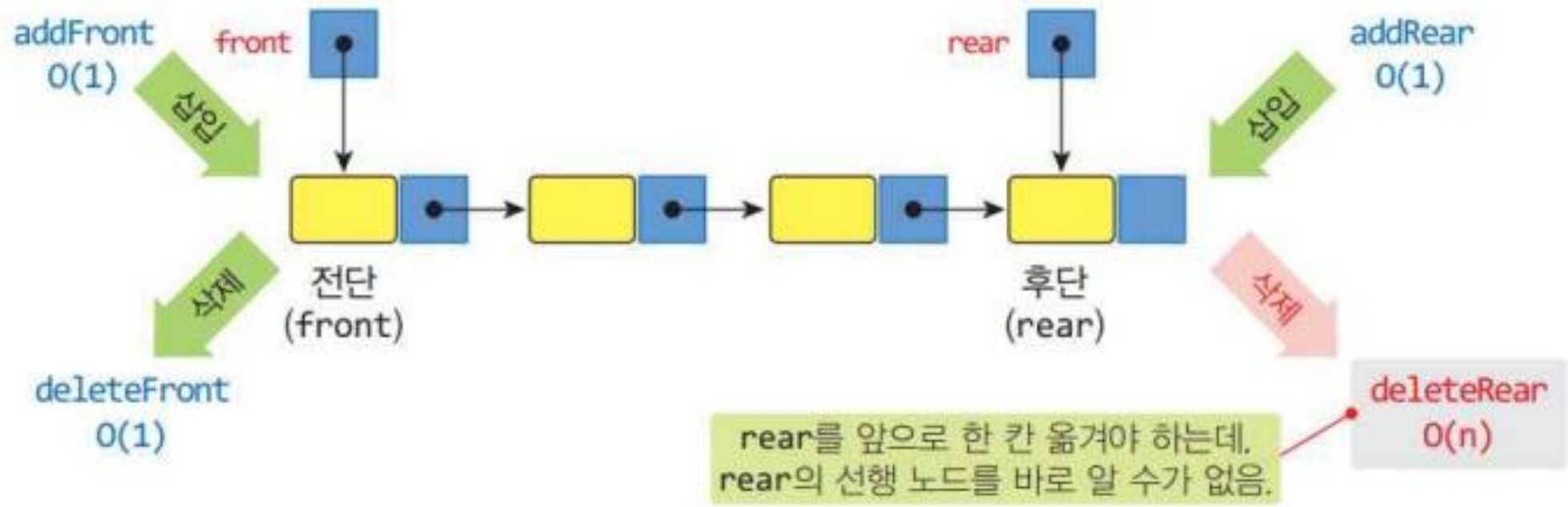
3. 원형 연결 리스트로 구현하는 큐



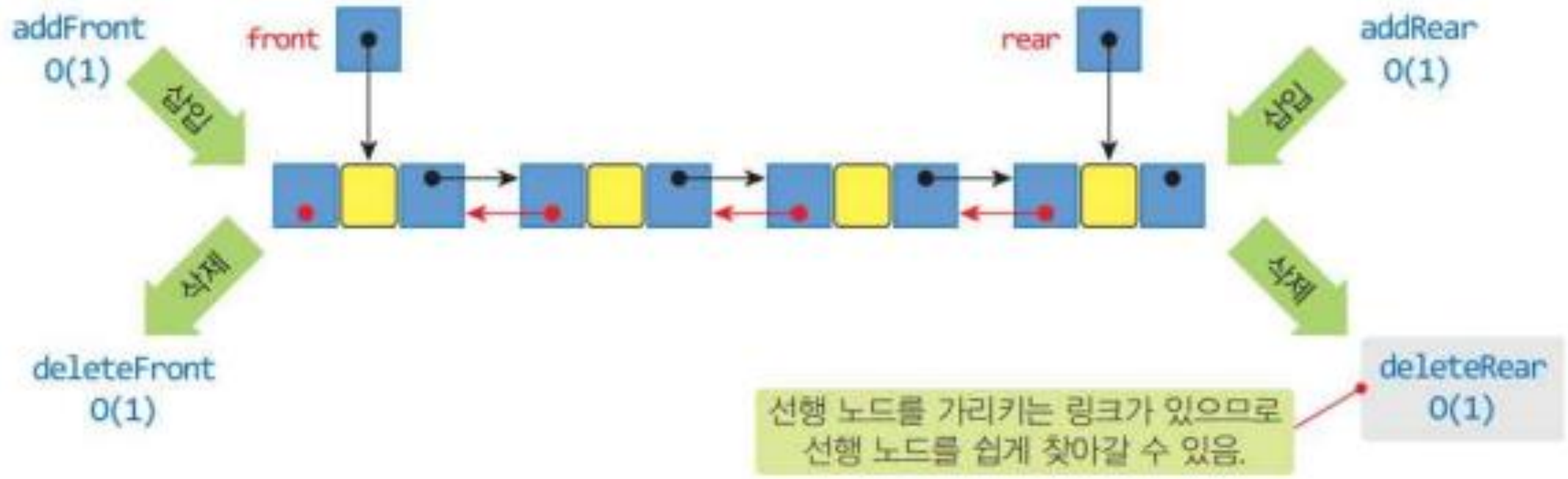
- tail을 사용하는 것이 rear 와 front에 바로 접근할 수 있다는 점에서 훨씬 효율적

덱을 구현하려면?

- 단순연결리스트로 구현한 덱



덱은 이중 연결 노드를 사용



자료구조, 왜 알아야 하지?

- 자료구조, 데이터를 관리하는 방법은 컴퓨터, 프로그래밍과 떼어낼 수 없는 매우 매우 중요한 개념

- 프로그래머는 자신이 사용하는 기술의 작동 원리와 성능을 파악해야 할 의무가 있음
  → 근본있는(올바른) 프로그래밍이 가능, CS 면접이 존재하는 이유

- 그래프 탐색, 스케줄링 등 중요 알고리즘들의 기초가 됨

- 일단 시험에 나온다.

자료구조를 편하게 사용하는 방법

howToUseModule.py

자료구조를 사용하는 예제

- 괄호 https://www.acmicpc.net/problem/9012
- 카드2 https://www.acmicpc.net/problem/2164
- 프린터 큐 https://www.acmicpc.net/problem/1966