Stack 스택

Queue 큐

Deque 덱

알고리즘 문제해결 강의 2회차

홍익대학교 함윤주

00 들어가기 전에

강사 소개

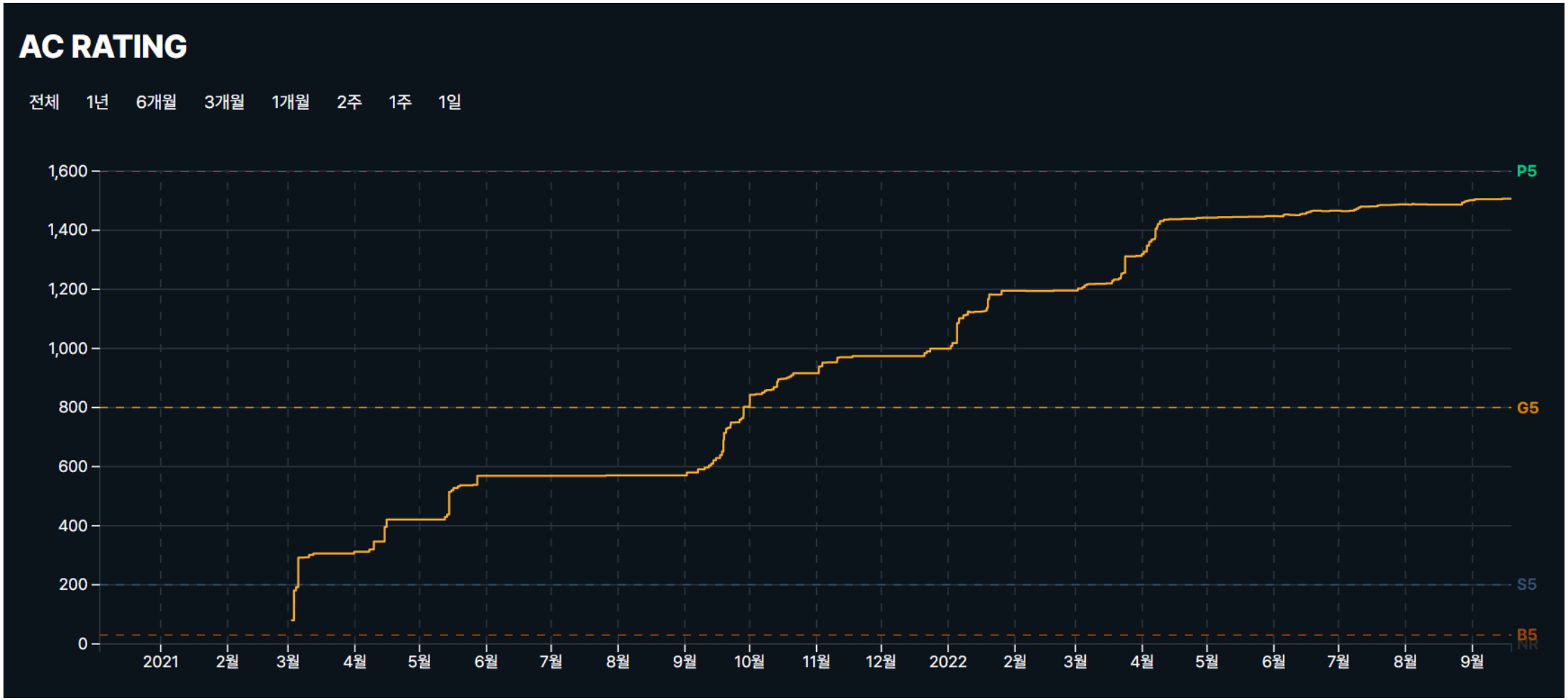
홍익대학교 컴퓨터공학과 20학번

BOJ : ohhamma (오함마)
플래가 가고 싶은 골린이

2022 ICPC Sinchon Winter Algorithm Camp Contest
- Novice 7th place

SUAPC 2022 Summer
- 28th place

OO 들어가기 전에


AC RATING
전체
1년
6개월
3개월
1개월
2주
1주
1일
1,600
1,400
1,200
1,000
800
600
400
200
0
P5
G5
S5
B5
NR
2021
2월
3월
4월
5월
6월
7월
8월
9월
10월
11월
12월
2022
2월
3월
4월
5월
6월
7월
8월
9월

00 들어가기 전에

자료구조란?

update, insert, access가 발생하는 data를
효율적으로 저장하고 관리하기 위한 도구

1. 선형 자료구조

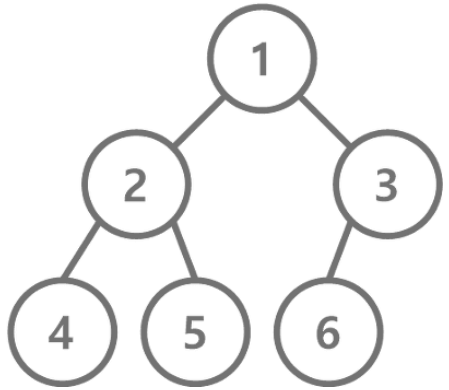
자료들 간의 앞뒤 관계가 1:1의 선형관계

→ 배열, 연결리스트, 스택, 큐, 덱

2. 비선형 자료구조

자료들 간의 앞뒤 관계가 1:n 또는 n:n의 관계

→ 그래프, 트리



00 들어가기 전에

학습 목표

1. 선형 자료구조 스택, 큐, 덱의 특징을 안다.

2. 문제를 읽고 어떤 선형 자료구조를 사용해야 하는지 판단할 수 있다.

목차
Contents

01 스택

02 큐

03 스택 vs 큐

04 덱

05 적절한 자료구조 선택하기

06 특징 정리

07 랜덤 문제 연습

08 연습문제

01 스택

개념 설명

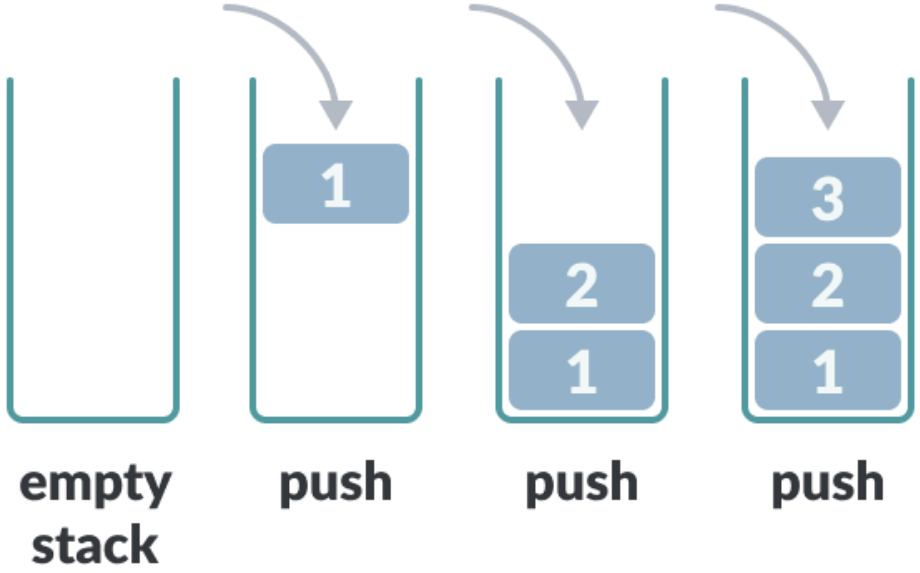
학습목표

1. 선형 자료구조 **스택**, 큐, 덱의 특징을 안다.

stack

n. 무더기, 더미

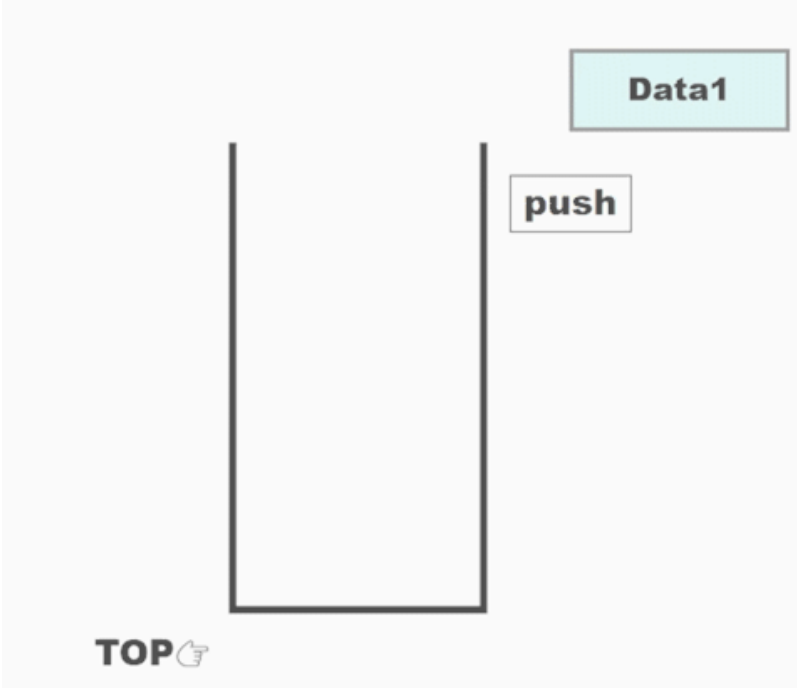
v. 쌓다, 쌓아서 채우다



LIFO

Last In First Out

후입선출



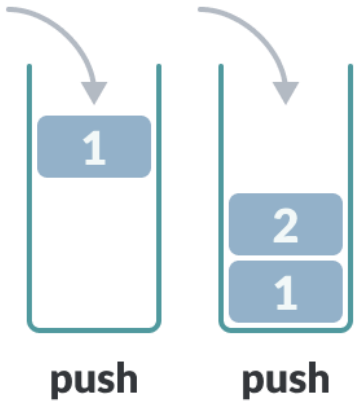
01 스택

주요 기능

push

push(x)

원소 x 추가

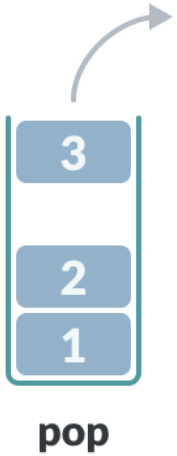


pop

pop()

가장 마지막에

넣은 원소 제거

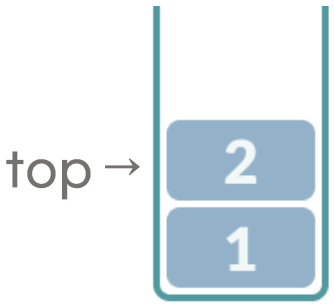


top

top()

가장 마지막에

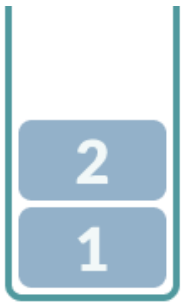
넣은 원소 반환



size

size()

원소 개수 반환



size = 2

empty

empty()

비어있으면

true

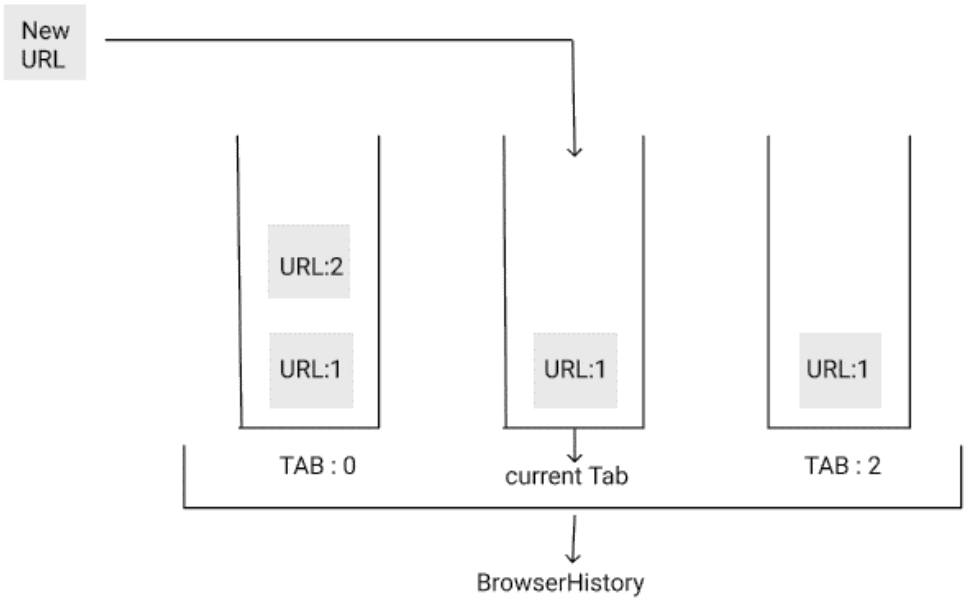
empty stack

(size = 0)

01 스택

실생활 예시

프링글스

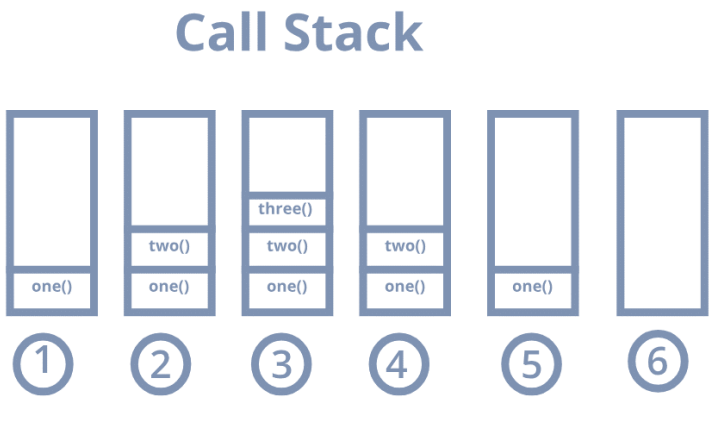


웹 브라우저 기록

```
function one(){
  two();
}

function two() {
  three();
}

function three() {
  console.trace("Call
          Stack");
}
```



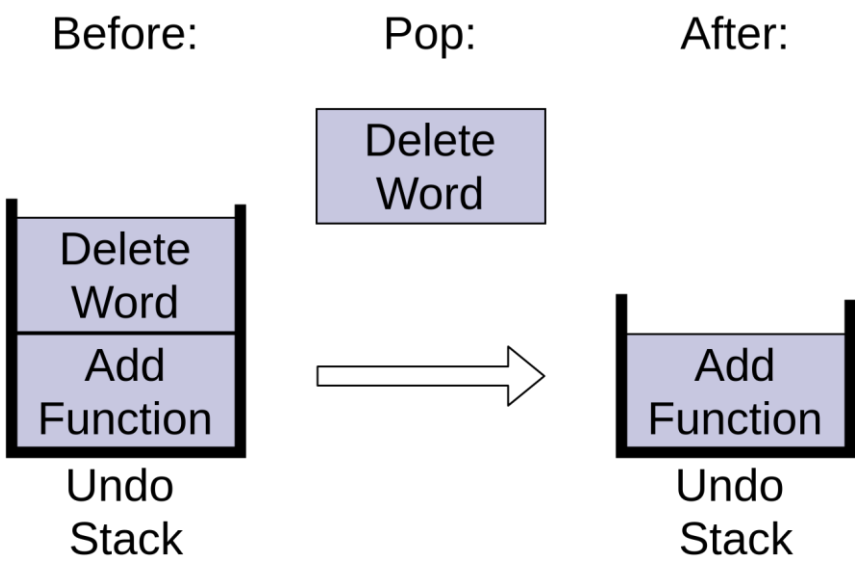
함수 호출

01 스택

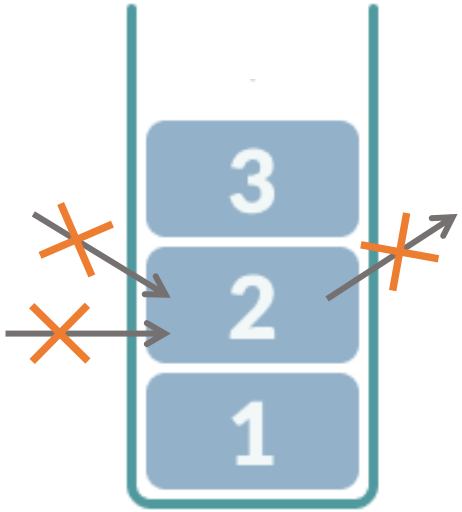
주로 언제 사용?

역추적, 실행 취소(ctrl+Z)

웹 브라우저 기록, 함수 호출의 연장선



중간 원소를 삽입/삭제/탐색 하지 않는 경우



01 스택

주로 언제 사용?

후위 표기법 계산

후위 표기법이란?

피연산자 뒤에 연산자가 오는 형식

중위표기법	후위표기법	전위표기법
A + B	A B +	+ A B
A + B – C	A B + C –	+ A – B C
A + B * C – D	A B C * + D –	– + A * B C D
(A + B) / (C – D)	A B+ C D – /	/ + A B – C D

→ 괄호를 없애는 과정이 스택과 유사

여는괄호 '('를 모두 스택에 넣어놓고
닫는괄호 ')'가 나올 때마다 스택 pop

(((A + ((B * C) / D)) + (E * F)) + G)

BC*

BC*D/

ABC*D/+

EF*

ABC*D/+EF*+

ABC*D/+EF*+G+

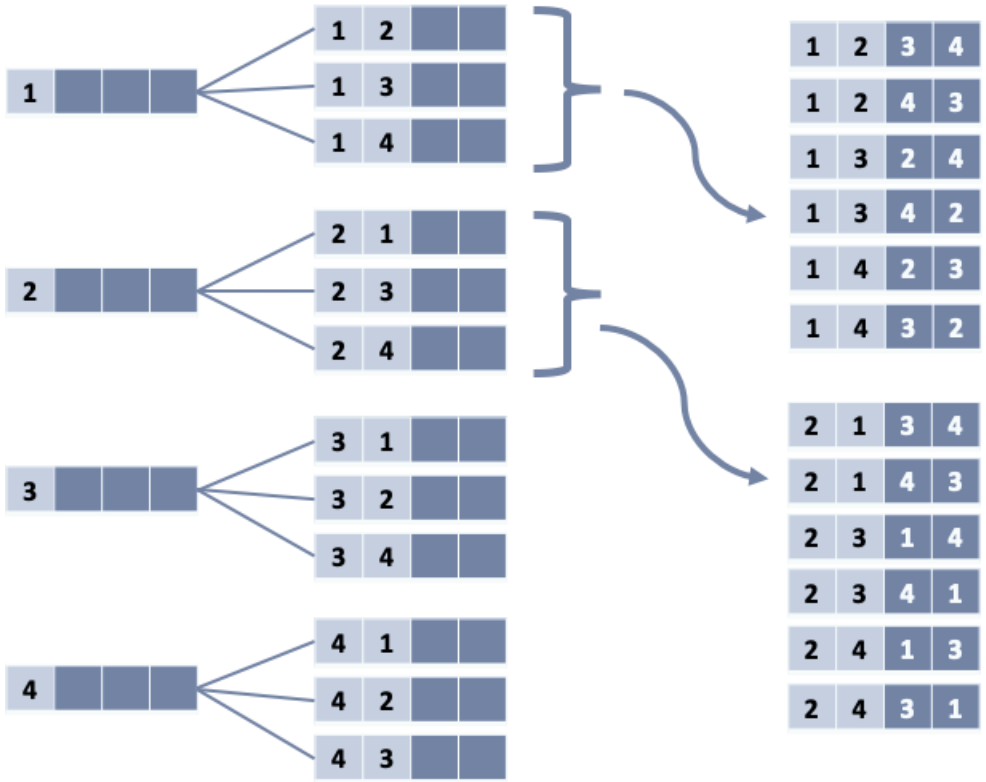
01 스택

활용 응용

재귀 알고리즘

함수 내부에서 자기 자신을 다시 호출해 작업을 수행하는 방식

ex) 순열 만들기



01 스택

문제 유형

monotone stack

단조로운 스택: 오름/내림차순을 유지한 스택

구현 아이디어

top과 새롭게 들어오는 수를 비교, 경우에 따라 pop하며 진행

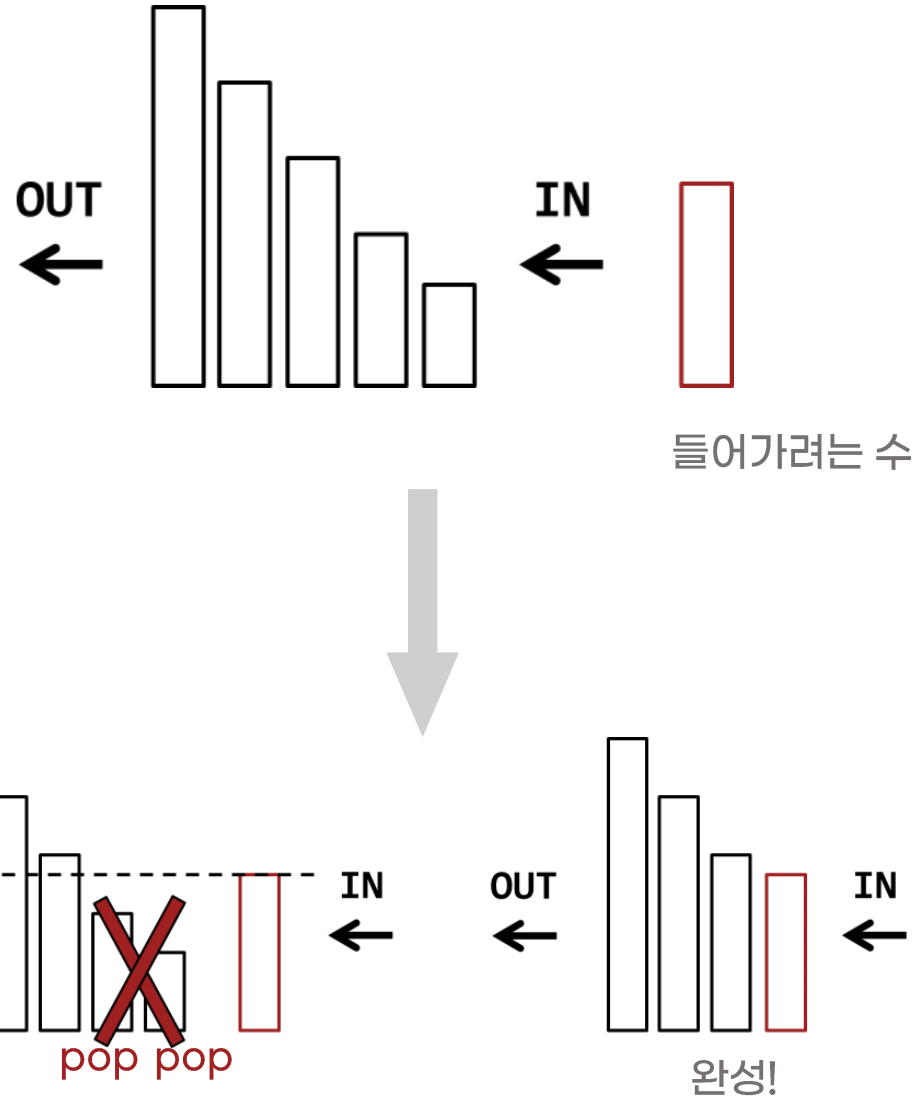
1. 중복없는 오름차순

   들어가려는 수 ≤ top인 경우 pop 반복

2. 중복없는 내림차순

   들어가려는 수 ≥ top인 경우 pop 반복

중복없는 내림차순 구현 과정



02 큐

개념 설명

큐가 빨리 안잡히네 ..

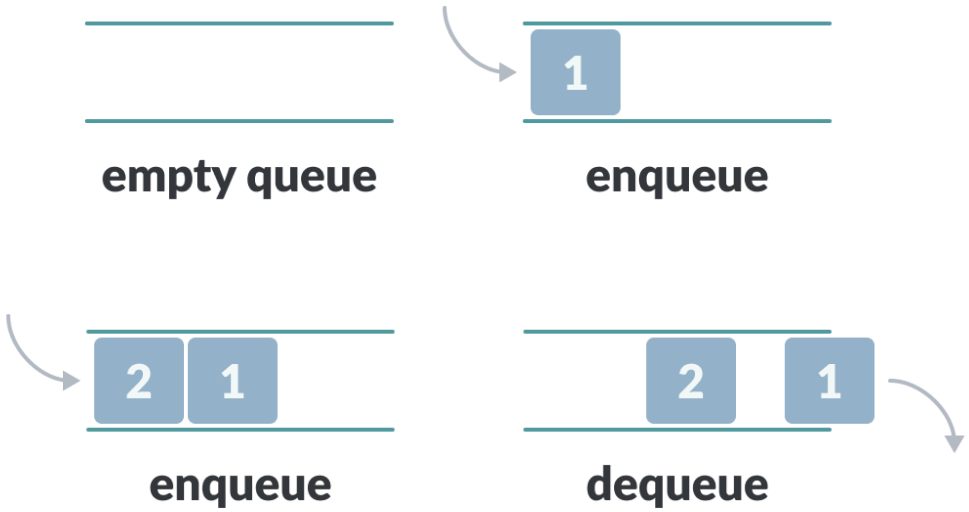
학습목표

1. 선형 자료구조 스택, 큐, 덱의 특징을 안다.

queue

n. 줄, 대기 행렬

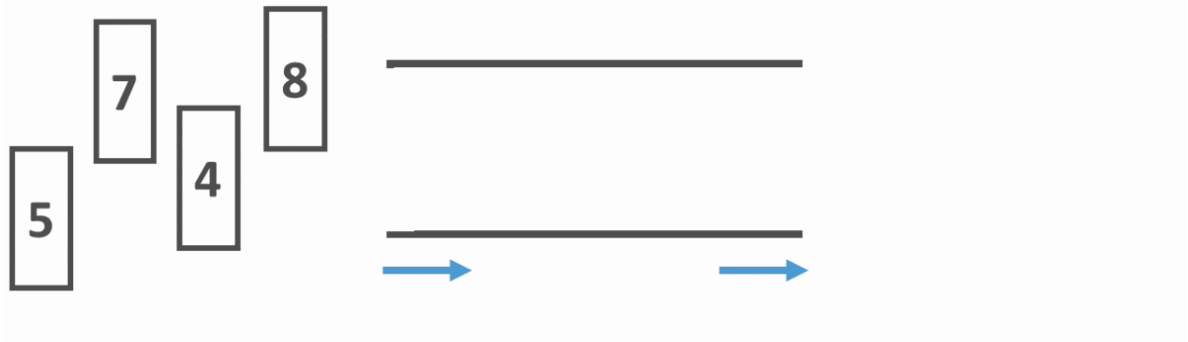
v. 줄을 서서 기다리다



FIFO

First In First Out

선입선출



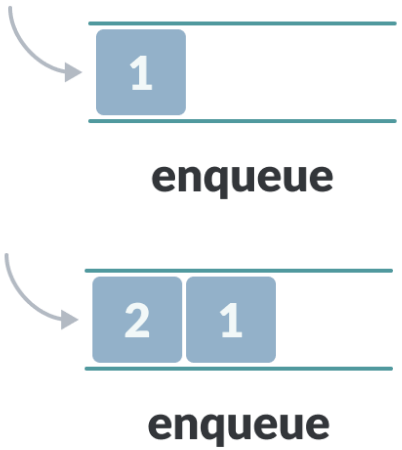
02 큐

주요 기능

push

push(x)

원소 x 추가



pop

pop()

가장 먼저

넣은 원소 제거

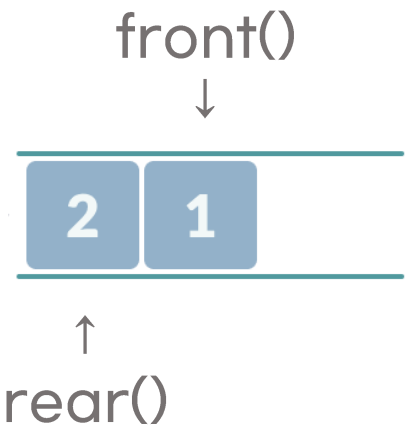
front

rear

front() / rear()

가장 먼저/마지막에

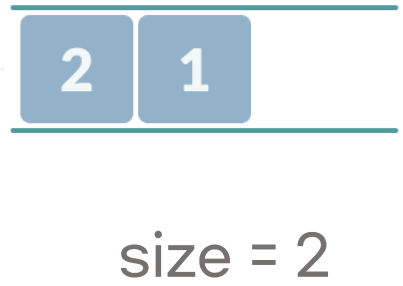
넣은 원소 반환



size

size()

원소 개수 반환

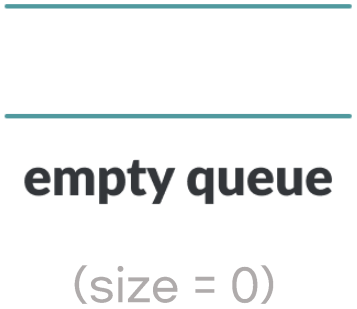


empty

empty()

비어있으면

true



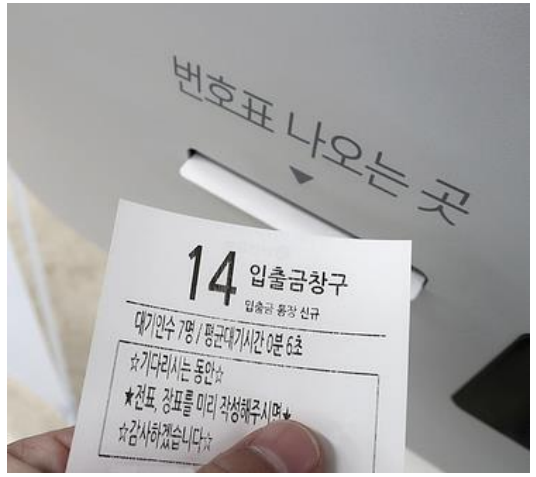
02 큐

실생활 예시

정수기 옆
종이컵 디스펜서

줄서기

은행창구 번호표 대기



운영체제의 태스크 스케쥴링

- **예제**

Process	Burst Time
P_1	24
P_2	3
P_3	3

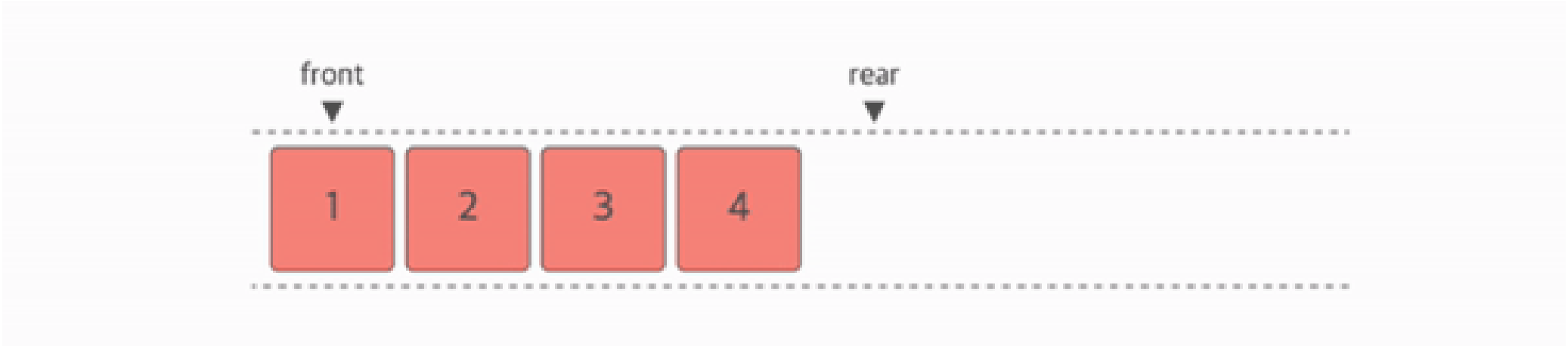
- **(1) P_1, P_2, P_3 순서로 요청하였을 때**

  - 대기 시간 : $P_1 = 0$, $P_2 = 24$, $P_3 = 27$
  - 평균 대기 시간 : (0 + 24 + 27) / 3 = 17

프린터 출력

02 큐

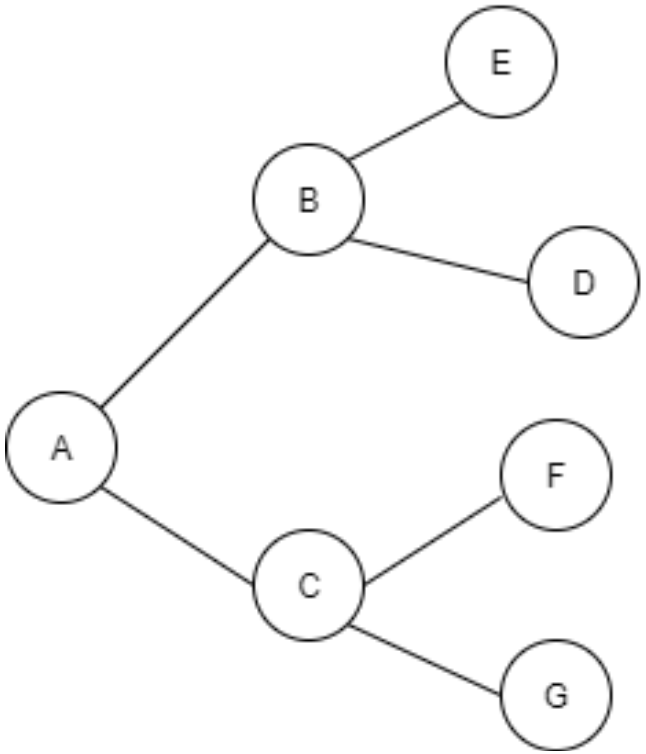
주로 언제 사용?

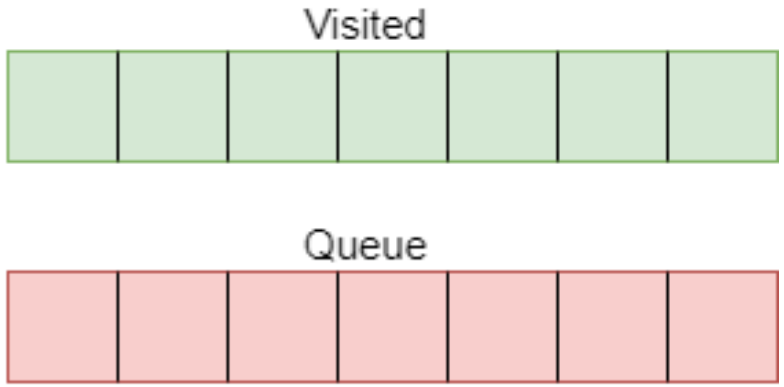


입력된 순서대로 데이터를 처리하는 경우

02 큐

활용 응용

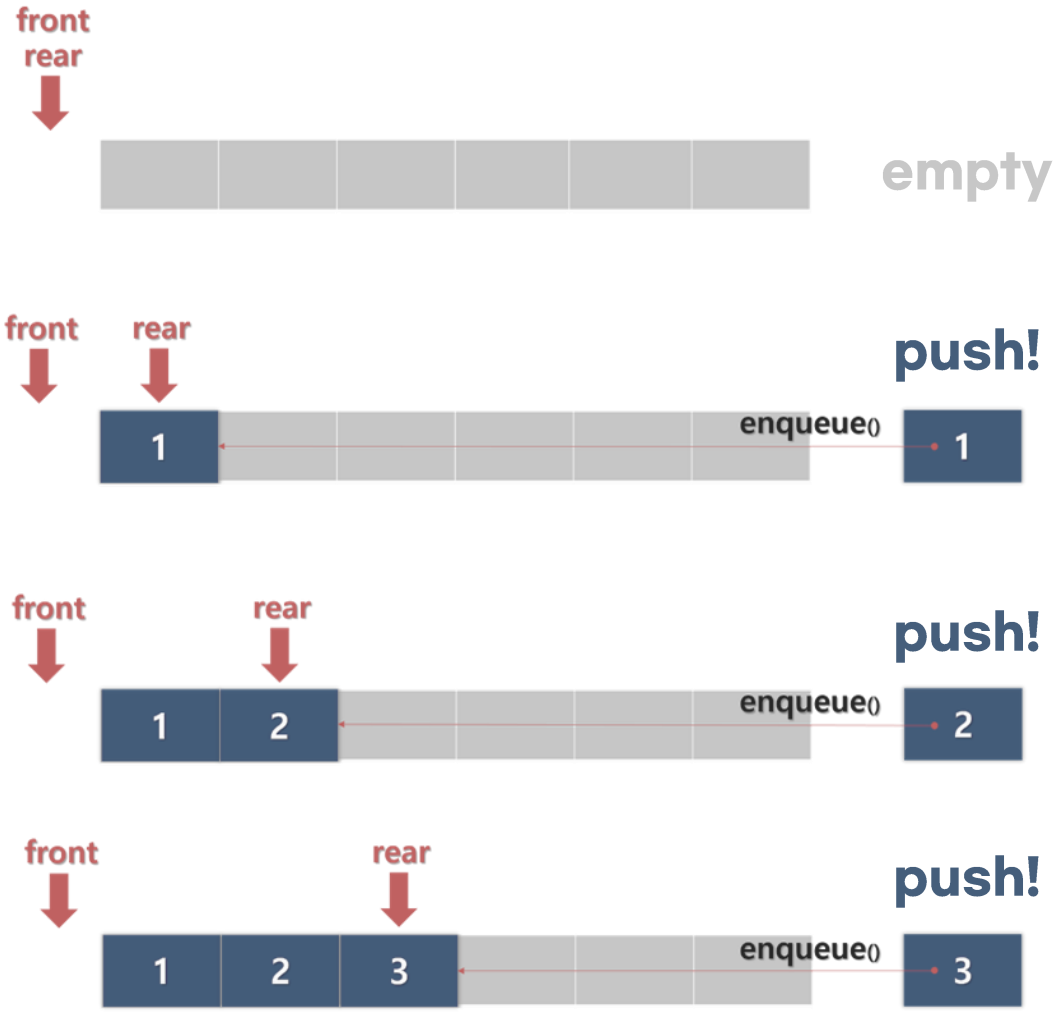


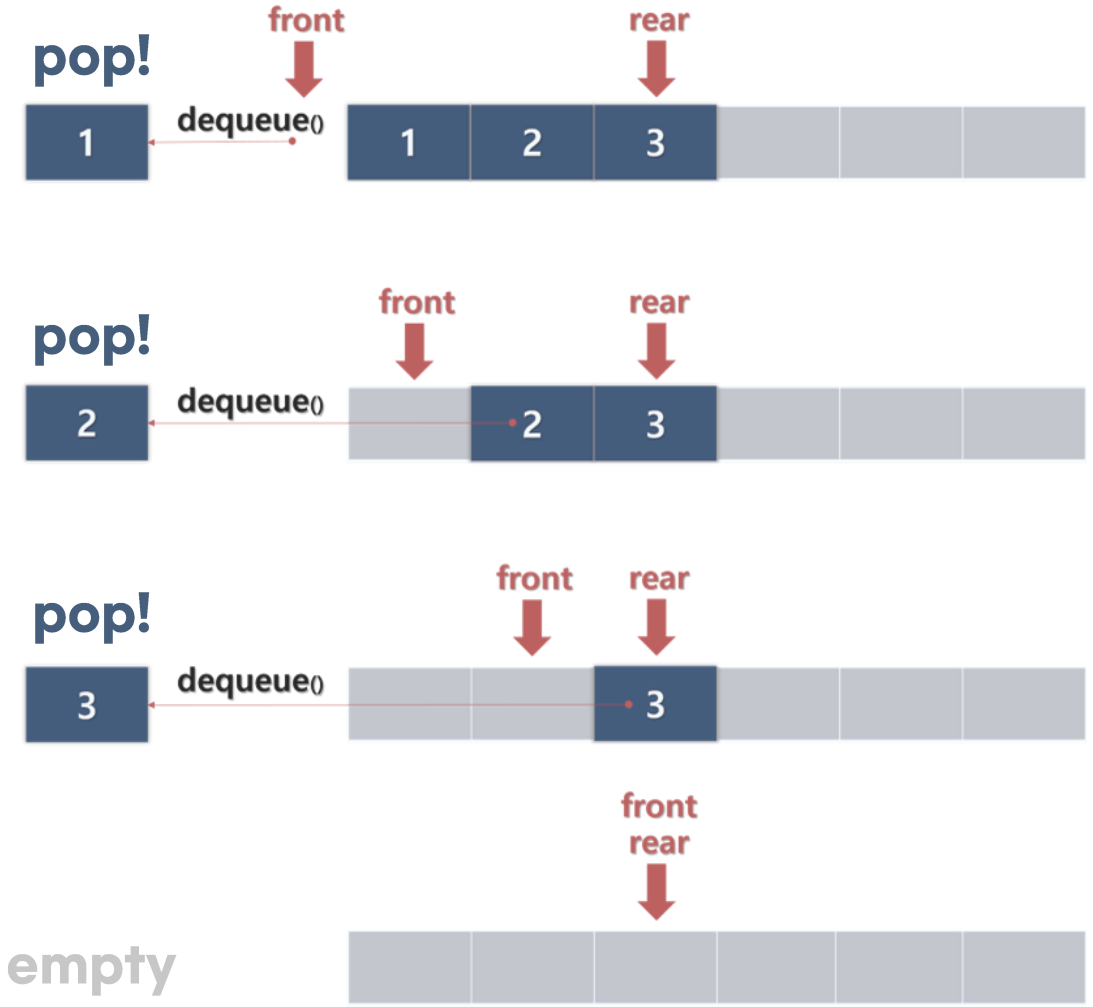


너비 우선 탐색 (BFS) 알고리즘

02 큐

선형 큐의 한계





데이터가 없을 때에도 비어있는 공간 활용할 수 없음

→ 비효율적

02 큐

선형 큐의 한계

한 번 더 **pop**한다면?

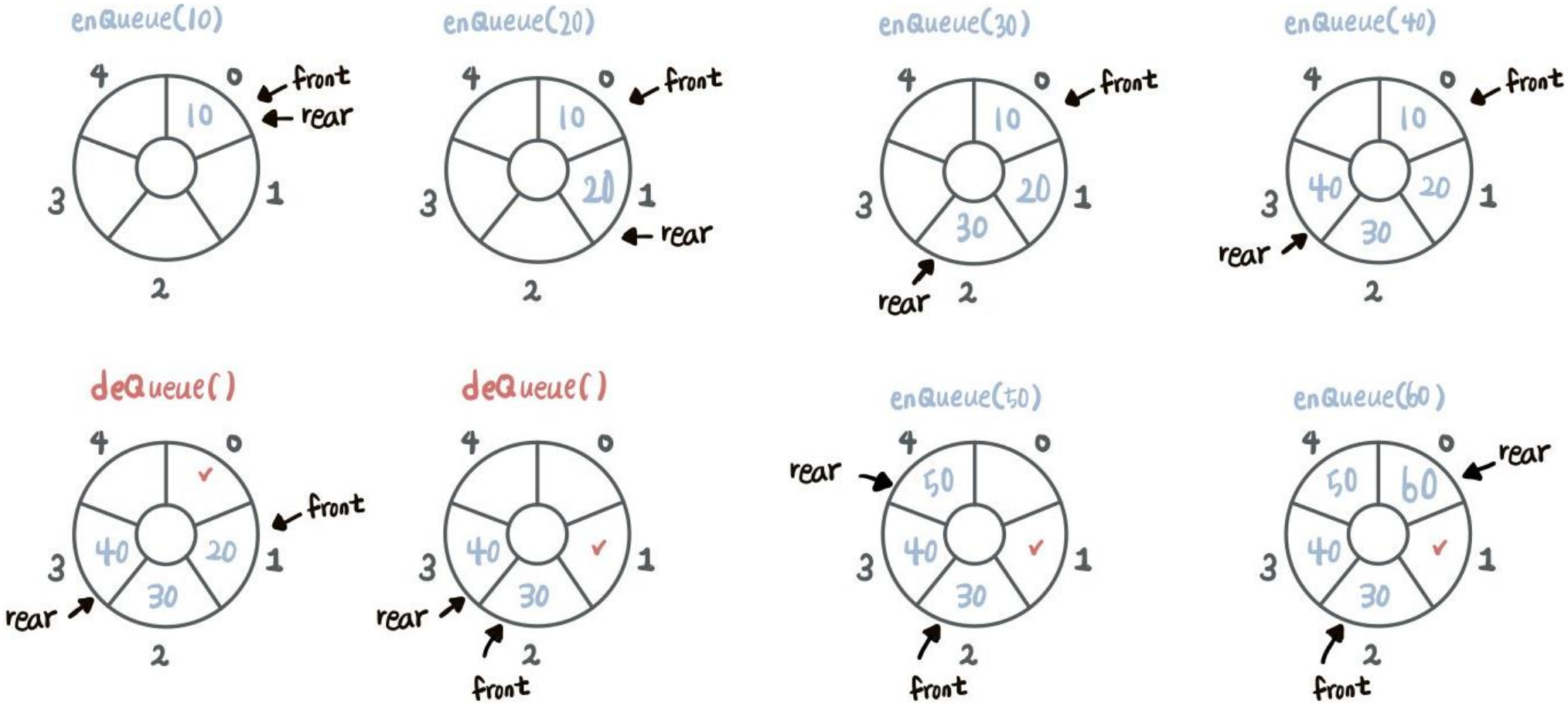
! error !

empty 상태에서 pop을 하면 (rear < front)

→ 오류 발생

02 큐

원형 큐



원형 큐를 사용하여 메모리도 줄일 수 있음

02 큐

문제 유형

어떤 문제들은 문제 접근 방법에서 큐를 사용할 수 있으나

자료구조 개념 특성상 단독으로 쓰이는 경우가 드묾

~ 쉬는시간 ~

10분 후 다시 시작합니다

03 스택 vs 큐

공통점

학습목표

1. 선형 자료구조 **스택**, **큐**, 덱의 특징을 안다.

장점

데이터 삽입 & 삭제 빠름

O(1)

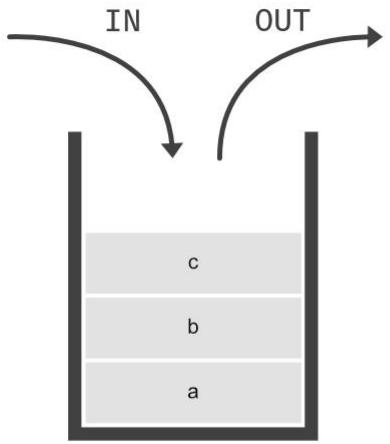
top/front/rear 원소 접근 빠름

단점

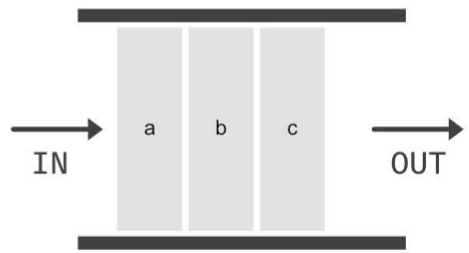
top/front/rear

특정 원소만 접근 가능,
이외에는 random access 불가능

중간 데이터를 삽입·삭제·탐색하기 어려움



스택(Stack)



큐(Queue)

03 스택 vs 큐

차이점

BOJ 24511 queuestack

문제

한가롭게 방학에 놀고 있던 도현이는 갑자기 재밌는 자료구조를 생각해냈다. 그 자료구조의 이름은 queuestack이다.

queuestack의 구조는 다음과 같다. 1번, 2번, ... , N번의 자료구조(queue 혹은 stack)가 나열되어있으며, 각각의 자료구조에는 한 개의 원소가 들어있다.

queuestack의 작동은 다음과 같다.

- x_0을 입력받는다.
- x_0을 1번 자료구조에 삽입한 뒤 1번 자료구조에서 원소를 pop한다. 그때 pop된 원소를 x_1이라 한다.
- x_1을 2번 자료구조에 삽입한 뒤 2번 자료구조에서 원소를 pop한다. 그때 pop된 원소를 x_2이라 한다.
- ...
- x_{N-1}을 N번 자료구조에 삽입한 뒤 N번 자료구조에서 원소를 pop한다. 그때 pop된 원소를 x_N이라 한다.
- x_N을 리턴한다.

도현이는 길이 M의 수열 C를 가져와서 수열의 원소를 앞에서부터 차례대로 queuestack에 삽입할 것이다. 이전에 삽입한 결과는 남아 있다. (예제 1 참고)

queuestack에 넣을 원소들이 주어졌을 때, 해당 원소를 넣은 리턴값을 출력하는 프로그램을 작성해보자.

03 스택 vs 큐

차이점

각 상태에 대한 큐스택 내부를 표현하면 다음과 같다.

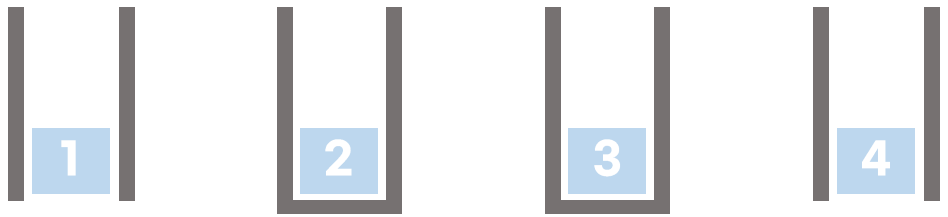
- 초기 상태 : $[1, 2, 3, 4]$
- 첫 번째 원소 삽입 : $[2, 2, 3, 1]$
- 두 번째 원소 삽입 : $[4, 2, 3, 2]$
- 세 번째 원소 삽입 : $[7, 2, 3, 4]$

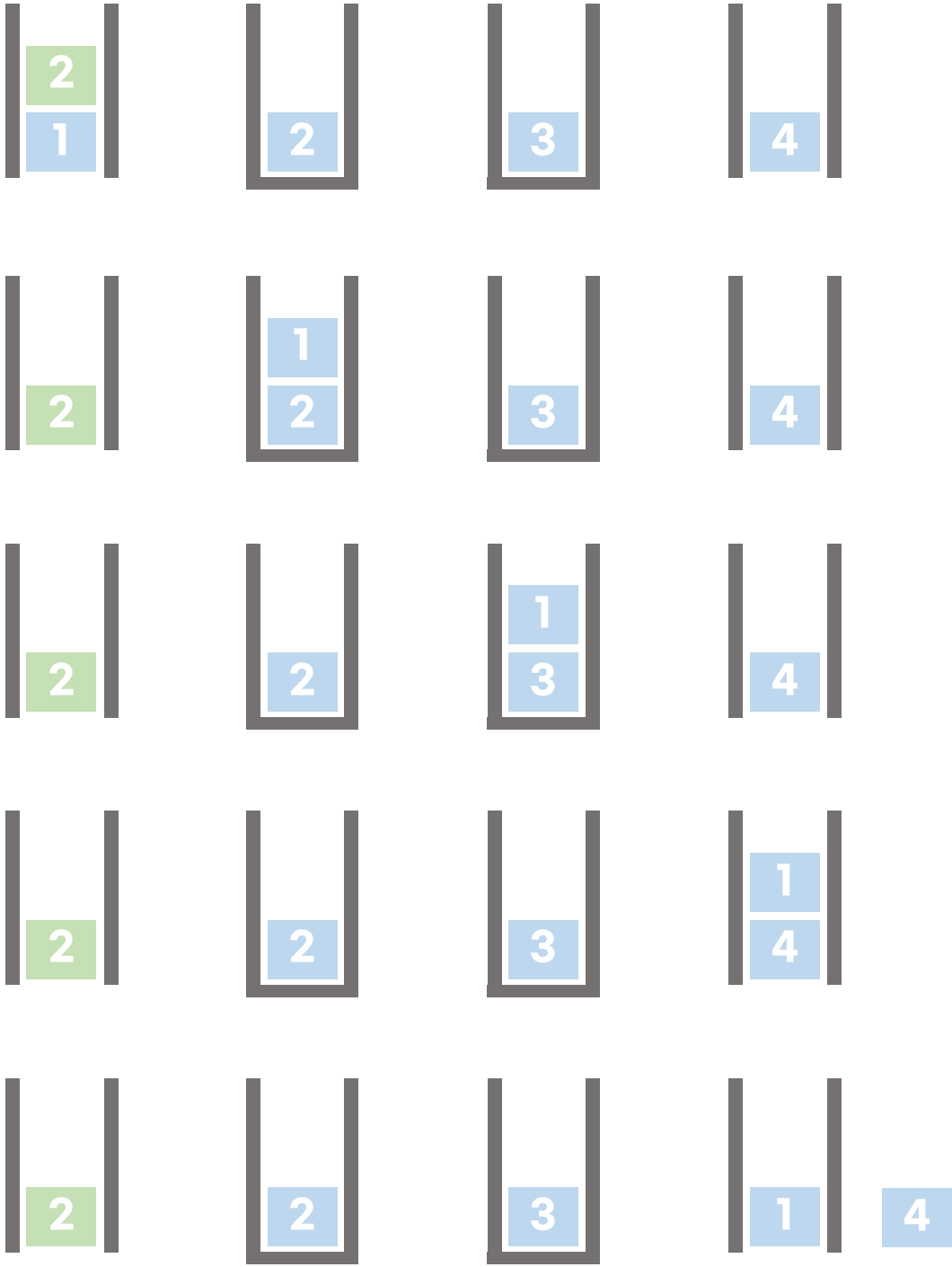
예제 입력 1 복사

```
4
0 1 1 0
1 2 3 4
3
2 4 7
```

0: 큐 / 1: 스택

초기상태

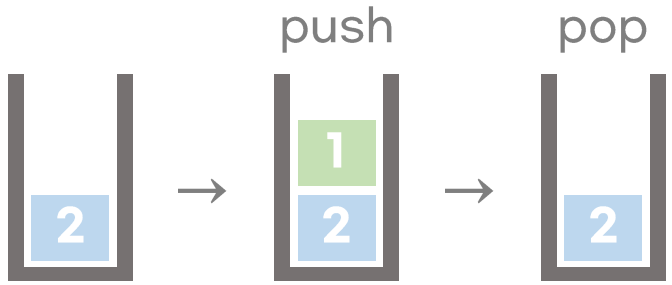




03 스택 vs 큐

차이점

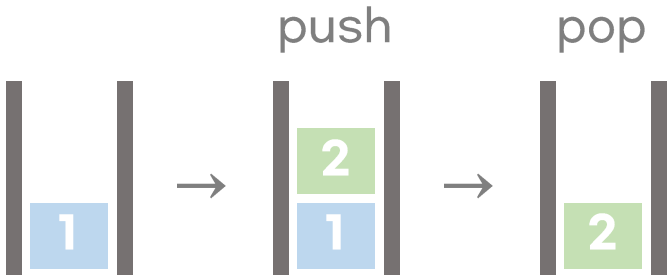
스택



LIFO

vs

큐



FIFO

04 덱

개념 설명

학습목표

1. 선형 자료구조 스택, 큐, 덱의 특징을 안다.

deque double-ended queue

ex) 카드 덱(deck)

양쪽에서 삽입, 삭제 모두 가능

04 덱

주요 기능

push_front

push_back

push_front(x)
원소 x 맨앞에 추가

push_back(x)
원소 x 맨뒤에 추가

pop_front

pop_back

pop_front(x)
맨앞의 원소 제거

pop_back(x)
맨뒤의 원소 제거

front

rear

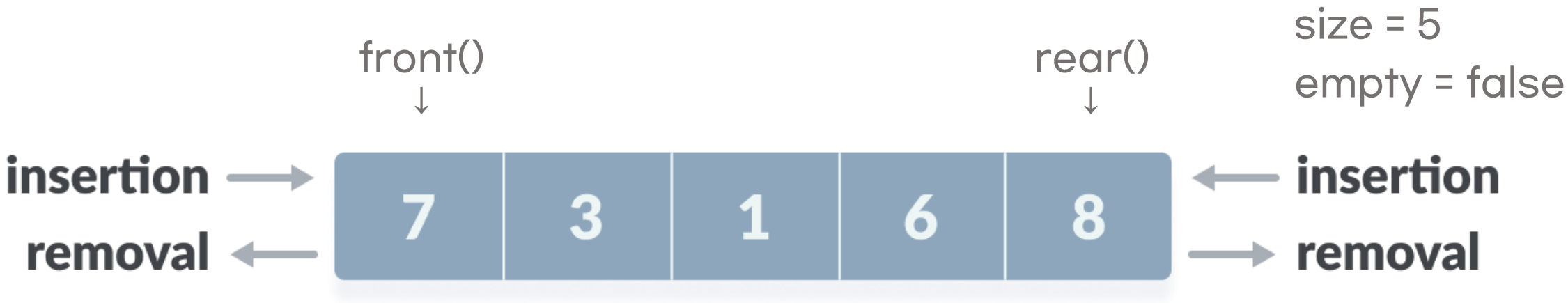
front() / rear()
맨앞/맨뒤의
원소 반환

size

size()
원소 개수 반환

empty

empty()
비어있으면
true



04 덱

실생활 예시

pop_front

도박에서 카드를 나눠줄때 보통 위에꺼 분배

아닌 경우를 밑장빼기라고 함

pop_back

push_front

노래방 우선 예약 시스템

cf) 기본 예약 시스템은 큐

push_back

04 덱

장단점

장점

데이터 삽입 & 삭제 빠름 — cf) 스택, 큐

가변적인 크기 — 새로운 원소 삽입/삭제할때
메모리 공간 자동으로 확장/축소

메모리가 연속적으로 존재하지x
cf) vector

인덱스 접근 가능 — operator[]을 사용해 데이터 접근

단점

최소 메모리 공간이 큼

중간 데이터를 삽입·삭제하기 어려움

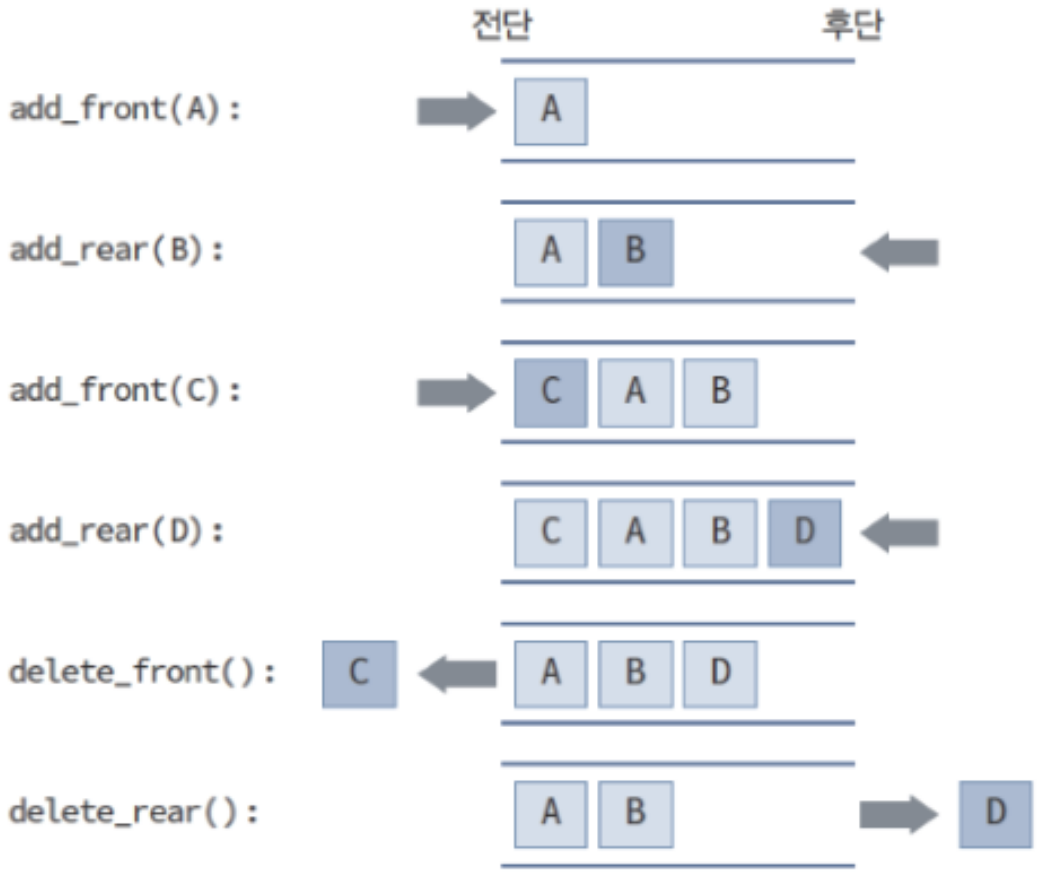
04 덱

주로 언제 사용?

주로 앞과 뒤에서
삽입/삭제 일어날 때

데이터 개수가 가변적일 때

검색을 거의 하지 않고
데이터에 임의 접근할 때



04 덱

활용 응용

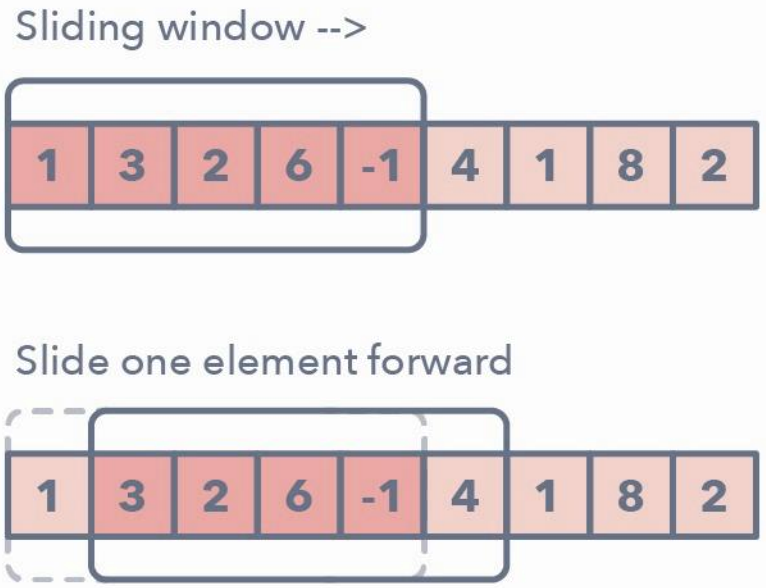
선형 공간을 2회 이상 반복적으로 탐색해야 할 경우

O(N^2) → O(N) 부분 배열 활용

sliding window

고정된 길이의 윈도우를 이동시키면서
윈도우 내의 데이터를 이용

ex) 일정범위의 구간 내에서
최솟값이나 최댓값을 구할 때



구간의 양 끝에서만
자료의 추가와 삭제가 일어남

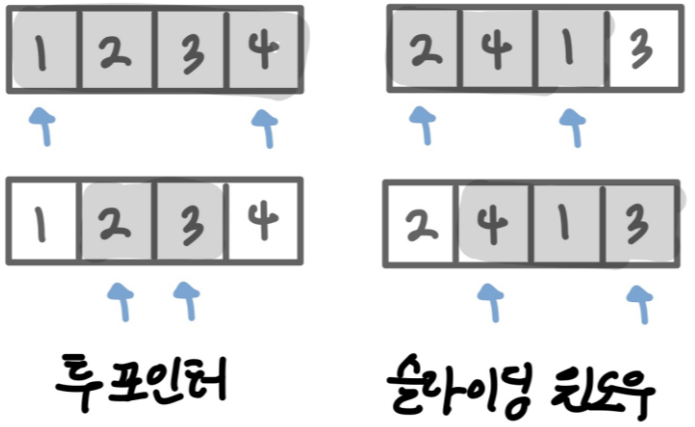
→ 덱

two pointer

start & end

부분 범위의 길이가 가변적

주로 정렬된 배열에서 이용



04 덱

활용 응용

monotone queue

단조로운 큐: 오름/내림차순을 유지한 큐

구현 아이디어

back과 새롭게 들어오는 수를 비교, 경우에 따라 pop_back하며 진행

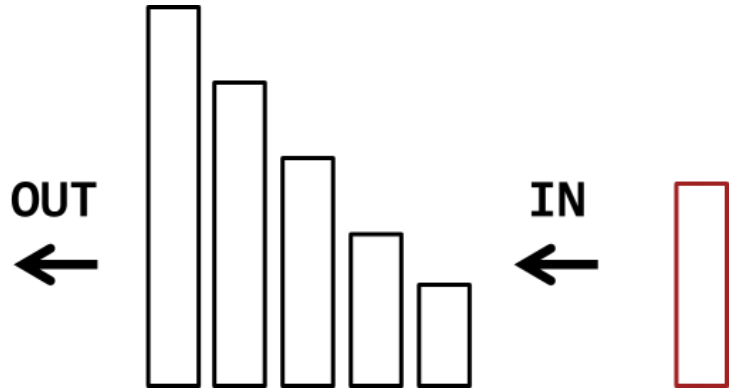
1. 중복없는 오름차순

들어가려는 수 ≤ back인 경우 pop_back 반복

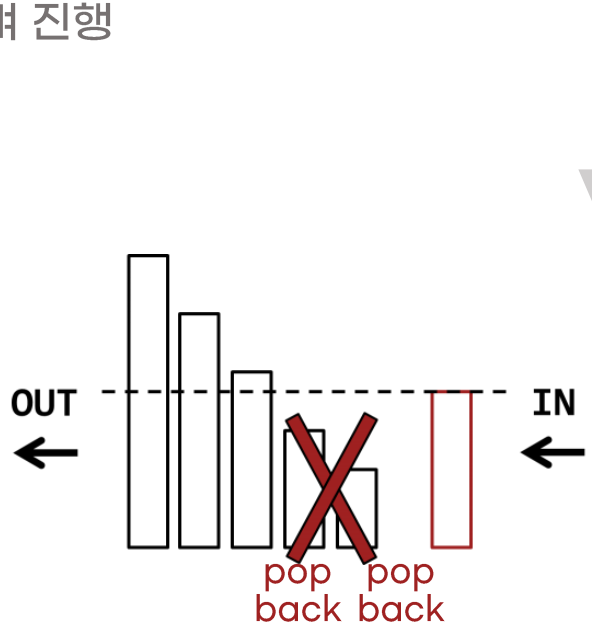
2. 중복없는 내림차순

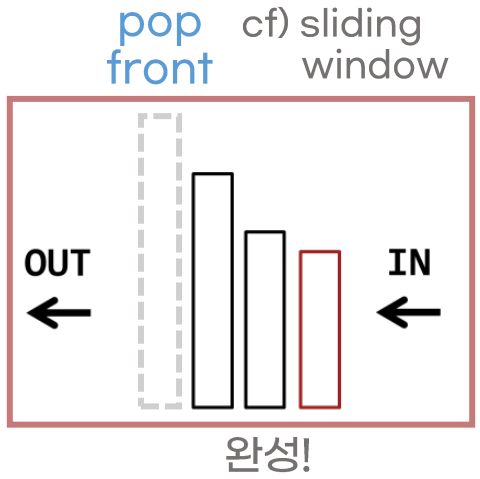
들어가려는 수 ≥ back인 경우 pop_back 반복

중복없는 내림차순 구현 과정



들어가려는 수





완성!

05 적절한 자료구조 선택하기

학습목표

2. 문제를 읽고 어떤 선형 자료구조를 사용해야 하는지 판단할 수 있다.

Q. 문제를 읽고 어떤 선형 자료구조를 선택해야 할까?

강사진의 의견 1

1. 전체 문제를 푸는 과정에서 나오는 부분 문제를 해결할 때 써야하는 적절한 자료구조 찾기
2. 메인 솔루션 자체가 특정 자료구조의 개형을 이용해야 하는 경우 고려

i) insert, delete가 한 쪽 끝에서만 발생하는가? 혹은 그렇게 되도록 문제를 재해석, 변형할 수 있는가?
→ 스택

ii) 양쪽 끝에서 insert, delete가 서로 반대방향에서 발생하는가? 혹은 ~
→ 큐

iii) 양쪽 끝에서만 insert, delete가 모두 발생하는가? 혹은 ~
→ 덱

05 적절한 자료구조 선택하기

학습목표

2. 문제를 읽고 어떤 선형 자료구조를 사용해야 하는지 판단할 수 있다.

Q. 문제를 읽고 어떤 선형 자료구조를 선택해야 할까?

강사진의 의견 2

1. 우선 배열 사용 고려 (머리속에서 시뮬레이션 돌려보기)
2. 메모리 과다 사용 or 원소 insert, delete 과정이 반복되는 경우 스택, 큐, 덱 사용 고려

1) 큐를 강제하는 상황이라고 판단될 때
→ 큐

2) 스택을 사용했을 때 풀이가 편해진다고 판단될 때
→ 스택

3) 앞과 뒤에서 insert, delete가 자주 이루어지면
→ 덱

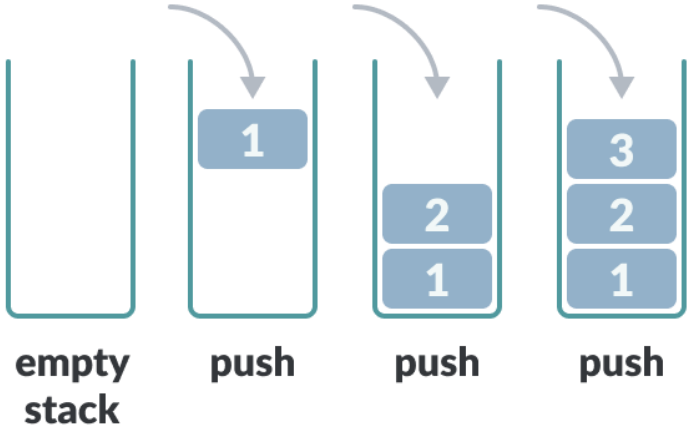
06 특징 정리

학습목표

1. 선형 자료구조
   스택, **큐**, **덱**의 특징을 안다.

원소
삽입/삭제
O(1)

스택
LIFO



큐
FIFO

덱
double-ended queue

07 랜덤 문제 연습

1158 요세푸스 문제

학습목표

2. 문제를 읽고 어떤 선형 자료구조를 사용해야 하는지 판단할 수 있다.

문제

요세푸스 문제는 다음과 같다.

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

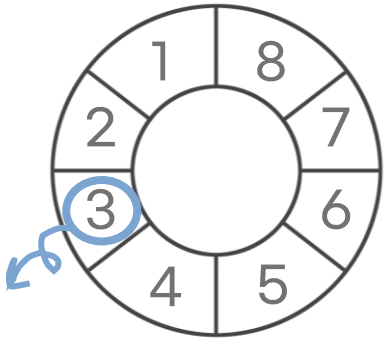
N과 K가 주어지면 (N, K)-요세푸스 순열을 구하는 프로그램을 작성하시오.

입력

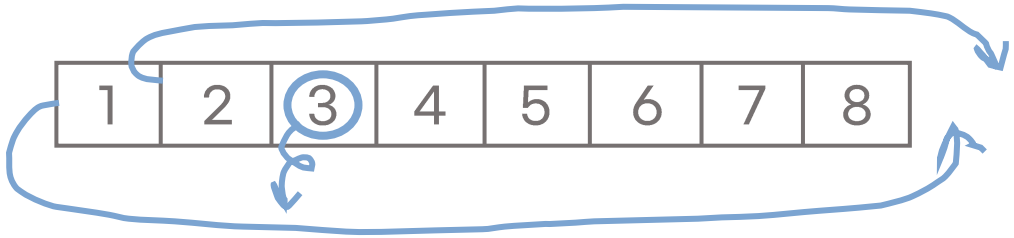
첫째 줄에 N과 K가 빈 칸을 사이에 두고 순서대로 주어진다. (1 ≤ K ≤ N ≤ 5,000)

1. 원에서 중간에 위치한 사람 제거
   → 배열 사용 가능

2. 원이라고 생각했을 때



큐
사용 가능



pop_front

push_back

07 랜덤 문제 연습

5430 AC

학습목표

2. 문제를 읽고 어떤 선형 자료구조를 사용해야 하는지 판단할 수 있다.

문제

선영이는 주말에 할 일이 없어서 새로운 언어 AC를 만들었다. AC는 정수 배열에 연산을 하기 위해 만든 언어이다. 이 언어에는 두 가지 함수 R(뒤집기)과 D(버리기)가 있다.

함수 R은 배열에 있는 수의 순서를 뒤집는 함수이고, D는 첫 번째 수를 버리는 함수이다. 배열이 비어있는데 D를 사용한 경우에는 에러가 발생한다.

함수는 조합해서 한 번에 사용할 수 있다. 예를 들어, "AB"는 A를 수행한 다음에 바로 이어서 B를 수행하는 함수이다. 예를 들어, "RDD"는 배열을 뒤집은 다음 처음 두 수를 버리는 함수이다.

배열의 초기값과 수행할 함수가 주어졌을 때, 최종 결과를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 테스트 케이스의 개수 T가 주어진다. T는 최대 100이다.

각 테스트 케이스의 첫째 줄에는 수행할 함수 p가 주어진다. p의 길이는 1보다 크거나 같고, 100,000보다 작거나 같다.

다음 줄에는 배열에 들어있는 수의 개수 n이 주어진다. ($0 \le n \le 100,000$)

다음 줄에는 $[x_1,...,x_n]$과 같은 형태로 배열에 들어있는 정수가 주어진다. ($1 \le x_i \le 100$)

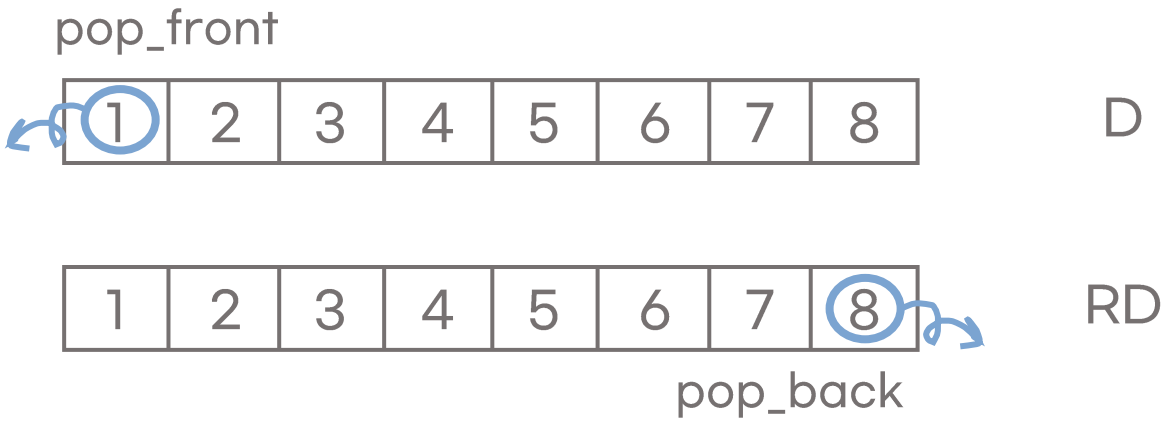
전체 테스트 케이스에 주어지는 p의 길이의 합과 n의 합은 70만을 넘지 않는다.

1. 배열을 사용해볼까?
→ 뒤집을 때마다 O(N)

2. 원소가 순서대로 들어오니까 큐?
→ front, rear을 매번 바꾸기 귀찮음

덱
사용 가능

3. 그럼 앞뒤로 삭제가 가능한 덱?
→ 뒤집을 때마다 앞뒤가 바뀌었다고 판단하면 됨



07 랜덤 문제 연습

9012 괄호

학습목표

2. 문제를 읽고 어떤 선형 자료구조를 사용해야 하는지 판단할 수 있다.

문제

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. 한 쌍의 괄호 기호로 된 "( )" 문자열은 기본 VPS 이라고 부른다. 만일 x 가 VPS 라면 이것을 하나의 괄호에 넣은 새로운 문자열 "(x)"도 VPS 가 된다. 그리고 두 VPS x 와 y를 접합(concatenation)시킨 새로운 문자열 xy도 VPS 가 된다. 예를 들어 "(())()"와 "((()))" 는 VPS 이지만 "((()", "(())())" , 그리고 "(()" 는 모두 VPS 가 아닌 문자열이다.

여러분은 입력으로 주어진 괄호 문자열이 VPS 인지 아닌지를 판단해서 그 결과를 YES 와 NO 로 나타내어야 한다.

입력

입력 데이터는 표준 입력을 사용한다. 입력은 T개의 테스트 데이터로 주어진다. 입력의 첫 번째 줄에는 입력 데이터의 수를 나타내는 정수 T가 주어진다. 각 테스트 데이터의 첫째 줄에는 괄호 문자열이 한 줄에 주어진다. 하나의 괄호 문자열의 길이는 2 이상 50 이하이다.

올바른 괄호 문자열(VPS)은 다음과 같이 정의한다.

1. 빈 문자열은 VPS이다.
2. s 가 VPS라면 (s) 도 VPS이다.
3. s 와 t 가 VPS라면 st 도 VPS이다.
4. 모든 VPS는 위 세 가지 규칙으로만 만들 수 있다.

예를 들어 " (())() ", " ()()() ", " (()()) " 등은 VPS이고, " ())(() ", " (", " (()()() " 등은 VPS가 아니다.

(((A + ((B * C) / D)) + (E * F)) + G)

')' 가 나올 때마다
가장 마지막에 들어간 '(' 삭제
→ LIFO
→ 스택

07 랜덤 문제 연습

9012 괄호

학습목표

2. 문제를 읽고 어떤 선형 자료구조를 사용해야 하는지 판단할 수 있다.

문제

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. 한 쌍의 괄호 기호로 된 "( )" 문자열은 기본 VPS 이라고 부른다. 만일 x 가 VPS 라면 이것을 하나의 괄호에 넣은 새로운 문자열 "(x)"도 VPS 가 된다. 그리고 두 VPS x 와 y를 접합(concatenation)시킨 새로운 문자열 xy도 VPS 가 된다. 예를 들어 "(())()"와 "((()))" 는 VPS 이지만 "((()(", "(())())" , 그리고 "(()" 는 모두 VPS 가 아닌 문자열이다.

여러분은 입력으로 주어진 괄호 문자열이 VPS 인지 아닌지를 판단해서 그 결과를 YES 와 NO 로 나타내어야 한다.

입력

입력 데이터는 표준 입력을 사용한다. 입력은 T개의 테스트 데이터로 주어진다. 입력의 첫 번째 줄에는 입력 데이터의 수를 나타내는 정수 T가 주어진다. 각 테스트 데이터의 첫째 줄에는 괄호 문자열이 한 줄에 주어진다. 하나의 괄호 문자열의 길이는 2 이상 50 이하이다.

')' 가 나올 때마다 가장 마지막에 들어간 '(' 삭제

```
stack st

loop for i from 0 to len
    if (i == '(')
        st.push(i)
    else if (i == ')')
        if (st.empty())
            print("NO")
            end loop
        else
            st.pop();
end loop

if (st.empty())
    print("YES")
else
    print("NO")
```

'(' 개수와 ')' 개수가 일치하지 않으면 NO

i) ')' 가 더 많은 경우
→ 스택이 먼저 비어 버림

ii) '(' 가 더 많은 경우
→ 마지막에 스택이 비어 있지 x

08 연습문제 (K = 7)

20301	반전 요세푸스
18115	카드 놓기
2493	탑
3078	좋은 친구
1863	스카이라인 쉬운거
13335	트럭
11003	최솟값 찾기
19591	독특한 계산기
17298	오큰수
9935	문자열 폭발
10975	데크 소트 2
1464	뒤집기3
14713	앵무새
1725	히스토그램

감사합니다