ANSI 알고리즘 스터디

01 스택 & 큐

01 자료 구조란

02 스택 (Stack)

03 큐 (Queue)

ANSi 아주대학교 알고리즘 소학회 Analyze N' Solve It

NAVER D2

01 자료 구조란?

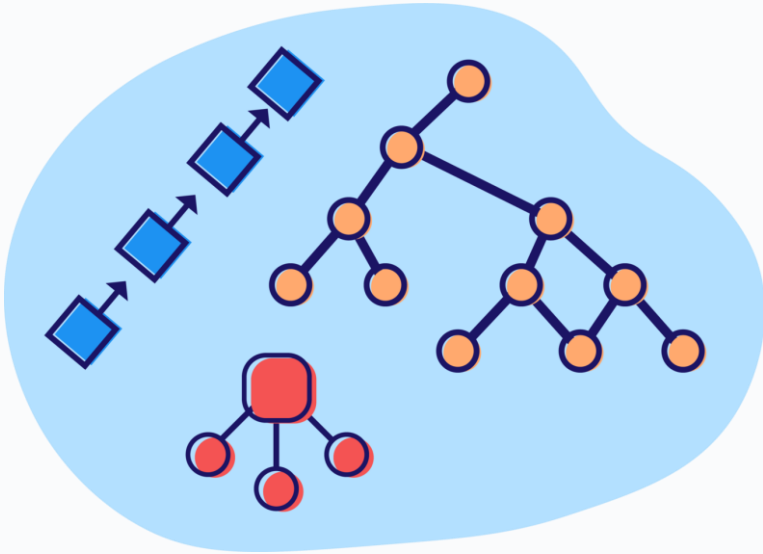
자료 구조란 무엇인가?

- 자료 구조(Data structure)는 자료(Data)에서 각 원소(Element)들마다 존재하는 논리적 관계의 표현

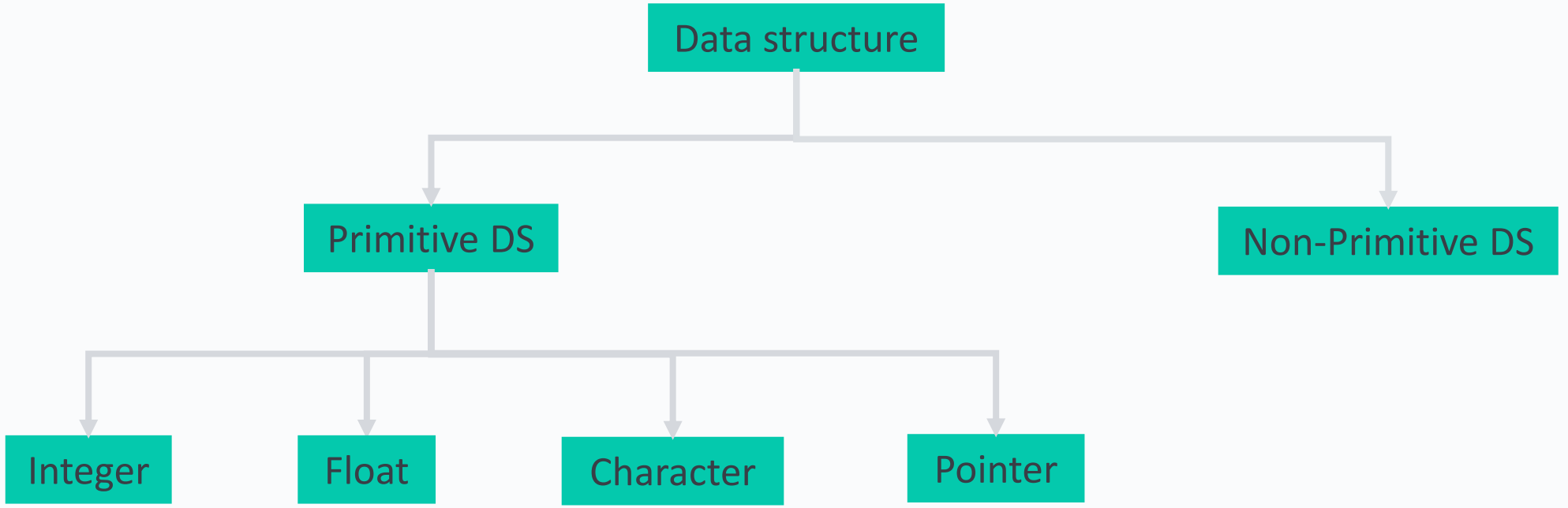
↓

- 자료 구조는 단지 자료들을 저장하기 위함이 아니라, 자료들 간의 관계도 고려한다!

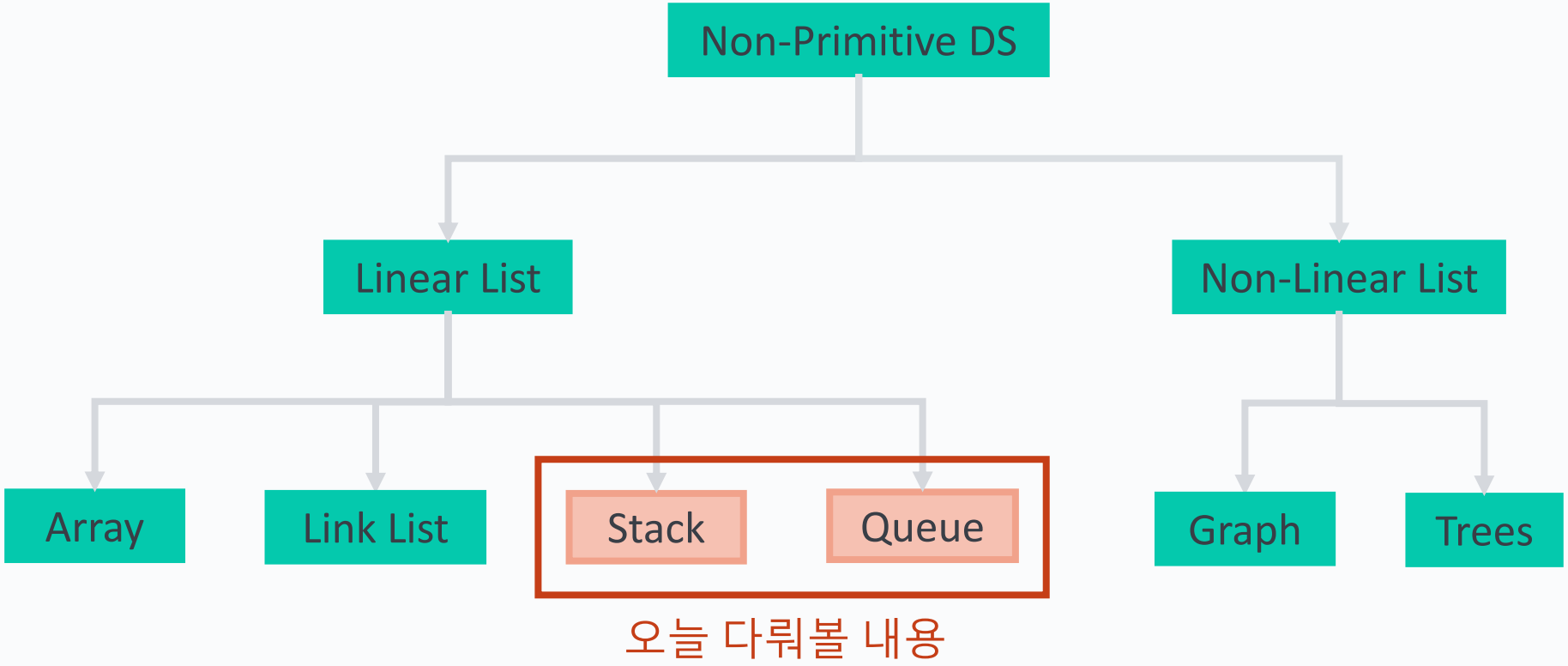
문제를 해결하기 위해서는 어떤 자료 구조를 사용할지 최우선적으로 고려해야한다



자료 구조의 종류


Data structure
Primitive DS
Non-Primitive DS
Integer
Float
Character
Pointer

자료 구조의 종류


Non-Primitive DS
Linear List
Non-Linear List
Array
Link List
Stack
Queue
Graph
Trees
오늘 다뤄볼 내용

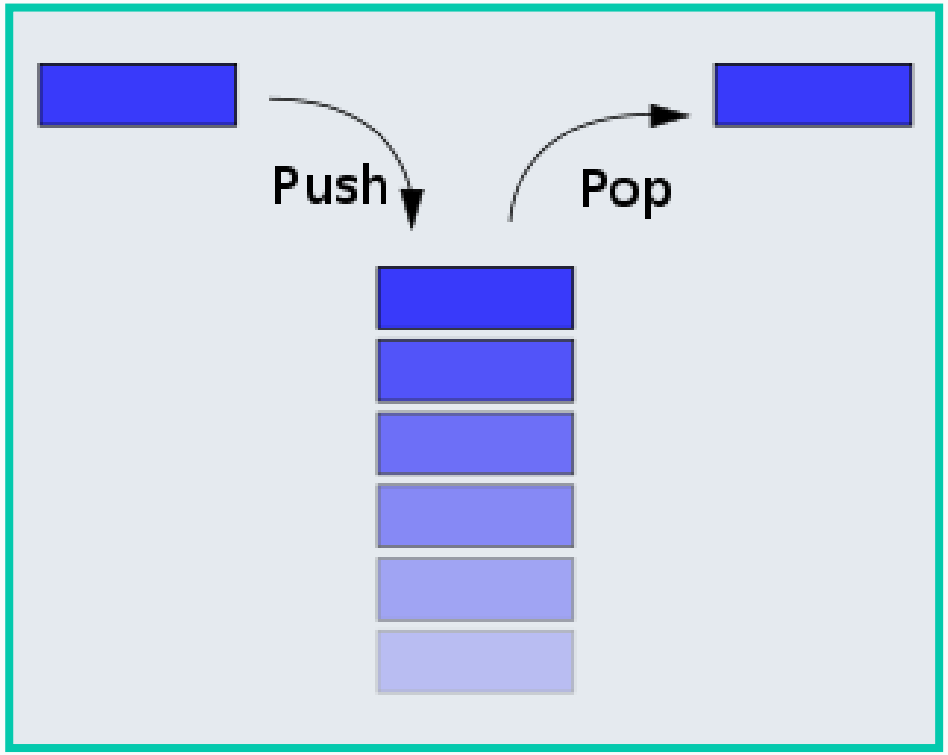
02 스택 (Stack)

스택 (Stack)

- LIFO – Last In First Out
- Push – 원소를 collection에 추가
- Pop – 제거되지 않은 가장 최근에 추가한 원소를 제거
- Top – 제거되지 않은 가장 최근에 추가한 원소

Time complexity of Stack

Push (Insertion)	$O(1)$
Pop (Deletion)	$O(1)$

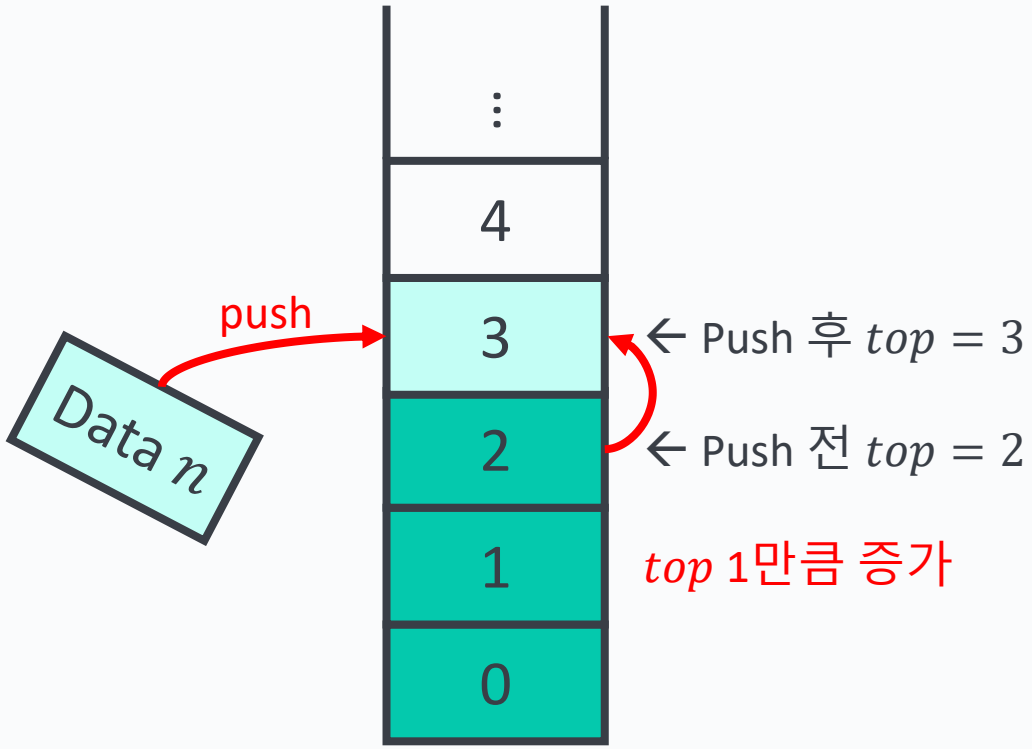


스택 (Stack) – push

- 배열로 구현한 스택 ($array[]$)
- top 인덱스의 초기값 = -1
- 데이터 n을 push하면 →

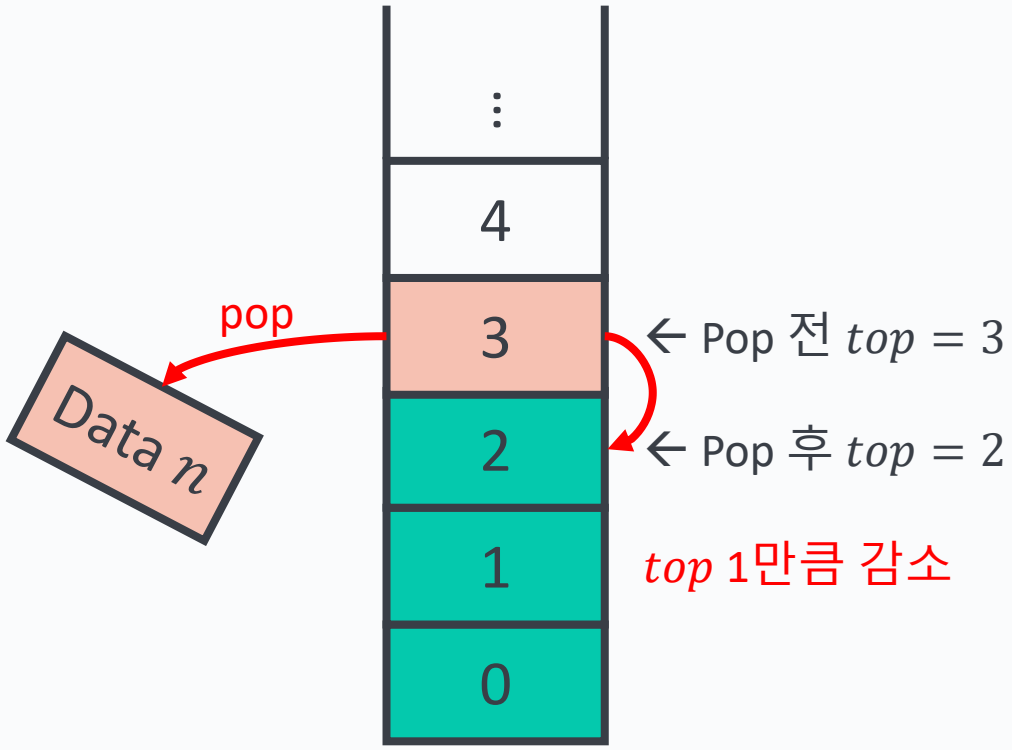
$$array[++top] = n$$

- top 원소 접근 - $array[top]$



스택 (Stack) – pop

- pop하면 – $--top$
- 단 top 이 -1 일 때, (스택에 원소가 없을 때) pop 호출 시 예외처리



스택 (Stack) STL

- C++의 STL(Standard Template Library)로 구현된 스택
- 스택을 사용하려면 <stack> header 를 include해야 함
- 실행 결과를 예측해 보자

- 자세한 내용은 → https://en.cppreference.com/w/cpp/container/stack

```cpp
#include <stdio.h>
#include <stack>
using namespace std;
int main () {
    stack<int> s;
    s.push(1);
    s.push(2);
    s.push(3);
    while (!s.empty()) {
        printf("%d\n", s.top());
        s.pop();
    }
    return 0;
}
```

#10828 스택 (icpc.me/10828)

- Push, pop, size, empty, top 등 스택의 연산을 사용해보는 문제
- 스택에 원소가 없을 때 pop이나 top 시도하면 -1을 출력 해야함

- 정답 코드 (C++, STL 사용): http://boj.kr/71b5378b32dc46e0affbb1ecf5acaa8e

앞으로 큐나 스택과 같은 자료구조를 사용할 때는 직접 구현하지 않고 STL을 사용합니다. STL에 익숙해지고 싶으시다면 예제문제를 꼭 풀어보세요

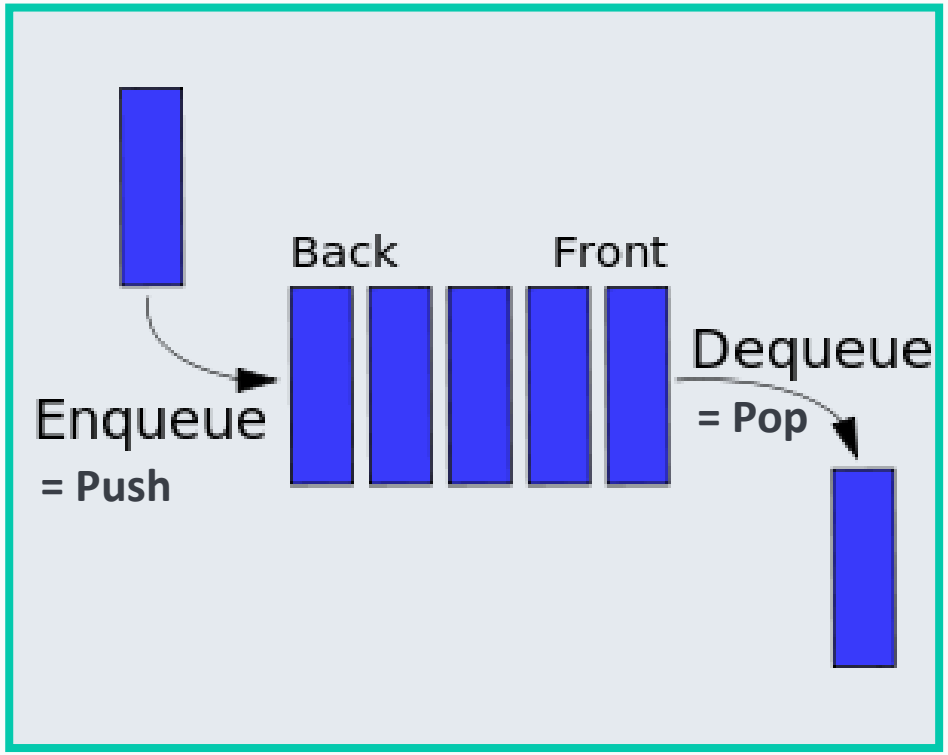
03 큐 (Queue)

큐 (Queue)

- FIFO – First In First Out
- Push – 원소를 collection에 추가
- Pop – 제거되지 않은 가장 오래전에 추가한 원소를 제거
- Front – 제거되지 않은 가장 오래전에 추가한 원소

Time complexity of Queue

Push (Insertion)	$O(1)$
Pop (Deletion)	$O(1)$

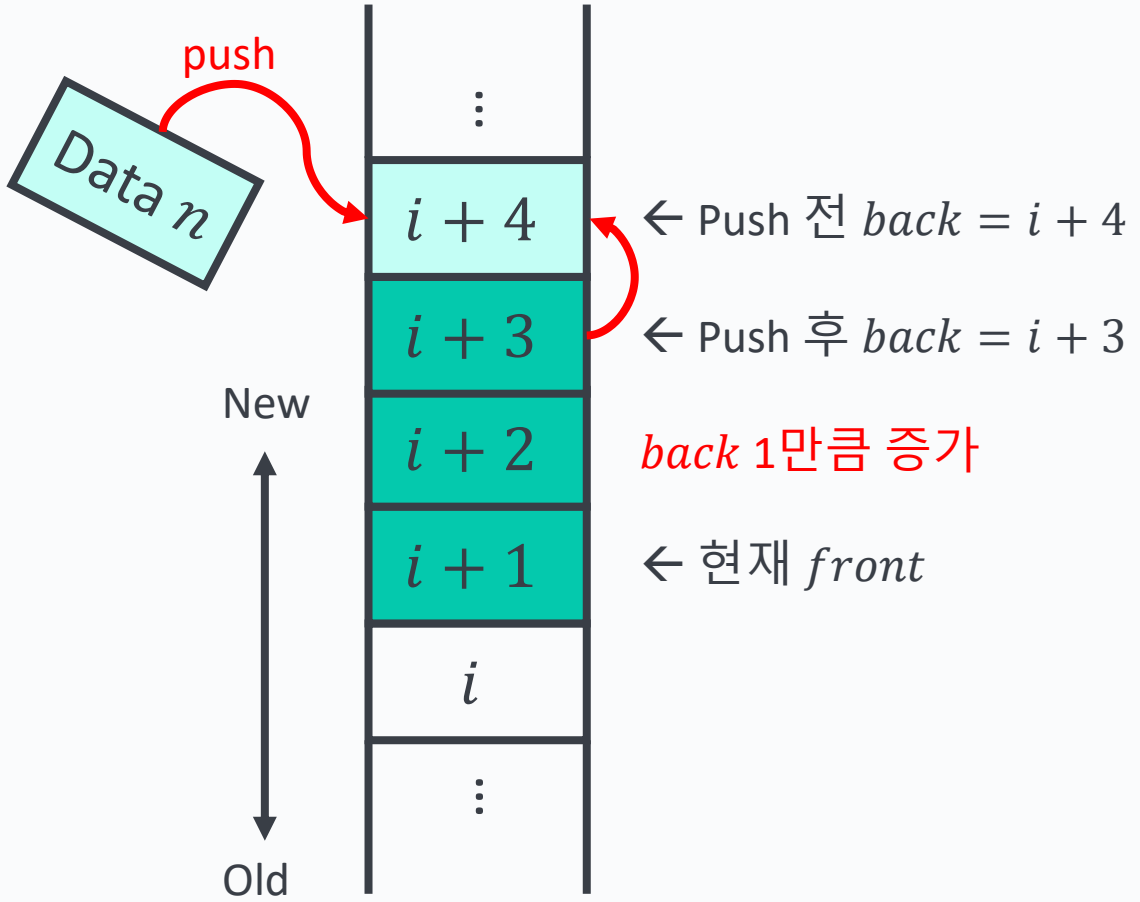


큐 (Queue) – push

- 배열로 구현한 큐 ($array[]$)
- $front$ 인덱스의 초기값 = 0
- $back$ 인덱스의 초기값 = -1
- 데이터 n을 push하면 →

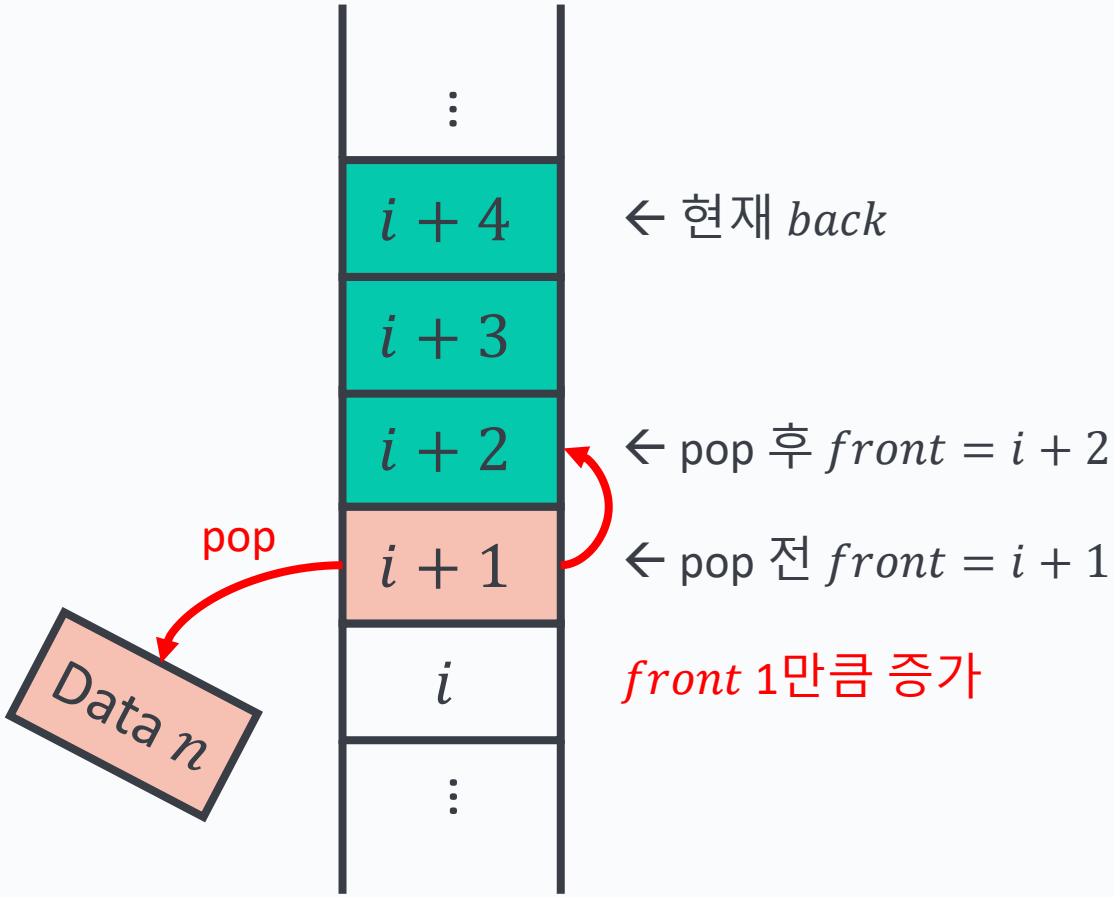
$$array[++back] = n$$

- $front$ 원소 접근 - $array[front]$



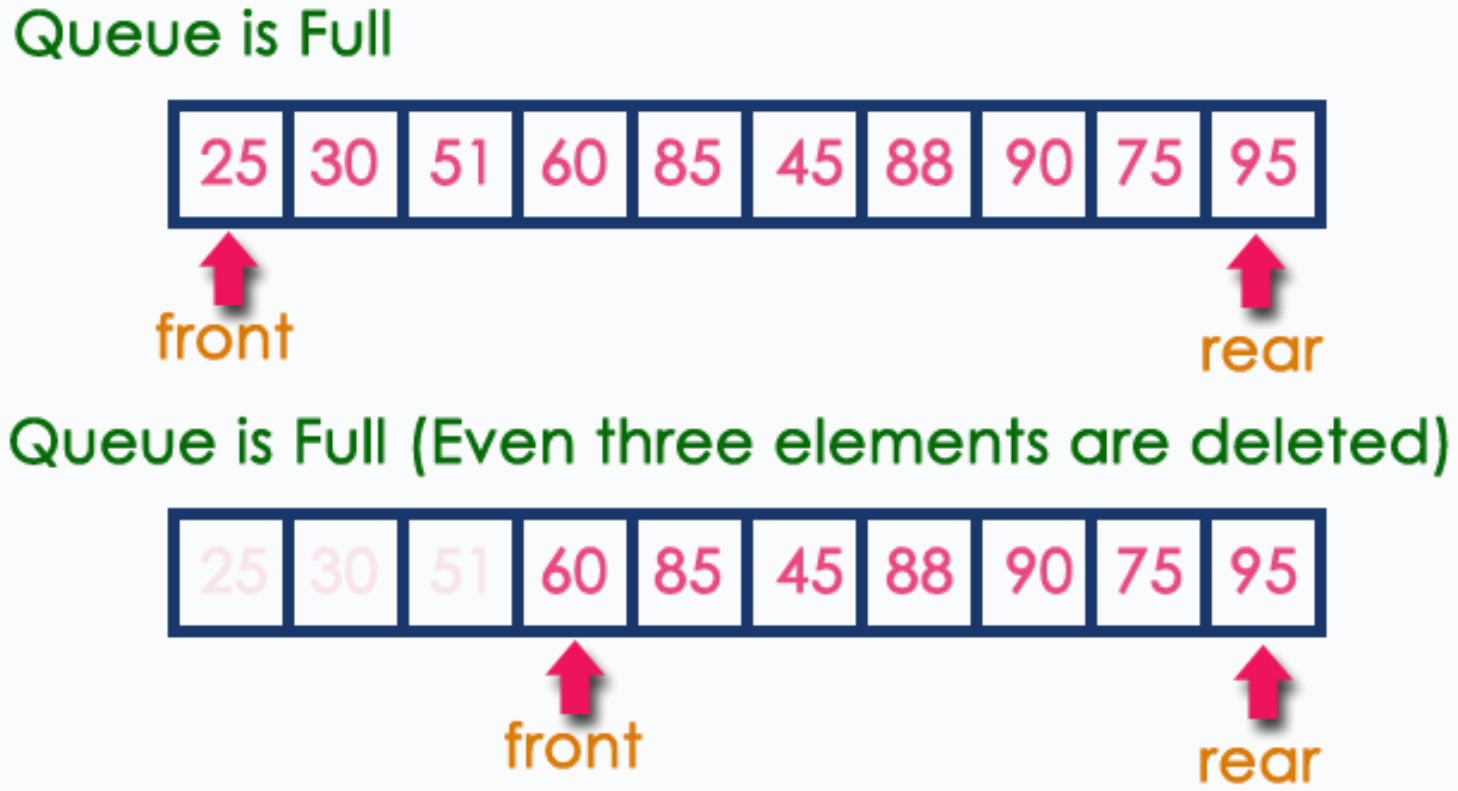
큐 (Queue) – pop

- pop하면 $front++$
- 단 $front > back$ 일 때, (큐에 원소가 없을 때) pop 호출 시 예외처리



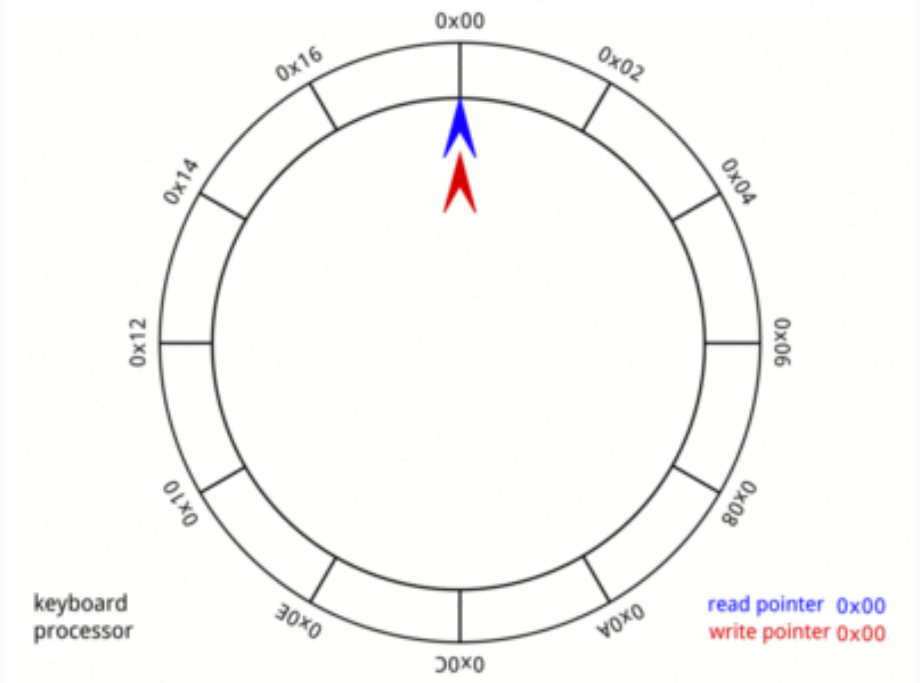
ANSi

문제점...


Queue is Full
25 30 51 60 85 45 88 90 75 95
front
rear
Queue is Full (Even three elements are deleted)
25 30 51 60 85 45 88 90 75 95
front
rear

Circular buffer 를 이용한 큐(Queue)

- 나머지 연산(Modular)을 이용하면 쉽게 구현 가능
- 항상 overflow는 조심하자... → 저장된 원소의 개수를 저장하면 편하다.



큐 (Queue) STL

- C++의 STL(Standard Template Library)로 구현된 큐
- 스택을 사용하려면 <queue> header 를 include해야 함
- 실행 결과를 예측해 보자

- 자세한 내용은 → https://en.cppreference.com/w/cpp/container/queue

```cpp
#include <stdio.h>
#include <queue>
using namespace std;
int main () {
    queue<int> q;
    q.push(1);
    q.push(2);
    q.push(3);
    while (!q.empty()) {
        printf("%d\n", q. front());
        q.pop();
    }
    return 0;
}
```

#10845 큐 (icpc.me/10845) & #18258 큐 2 (icpc.me/18258)

- push, pop, size, empty, front, back등 큐의 연산을 사용해보는 문제
- 큐에 원소가 없을 때 pop, front, back을 시도하면 -1을 출력 해야함
- '큐 2' 문제는 '큐' 문제보다 제한시간이 2배 더 많지만, 주어지는 명령의 수가 200배 더 많음

- '큐' 정답코드: http://boj.kr/ee170b4bd8034cd3a6da5675d65eb6e7
- '큐 2' 는 직접 도전해보자. 시간초과가 난다면 질문할 것.
  - (잘 구현된 queue를 사용하고, printf(), scanf()와 같이 충분히 빠른 입출력 사용할 것)

References

- [Data Structures Tutorials - Circular Queue with an example | Program (btechsmartclass.com)]

수고하셨습니다

다음 강의는 DFS & BFS입니다

ANSi