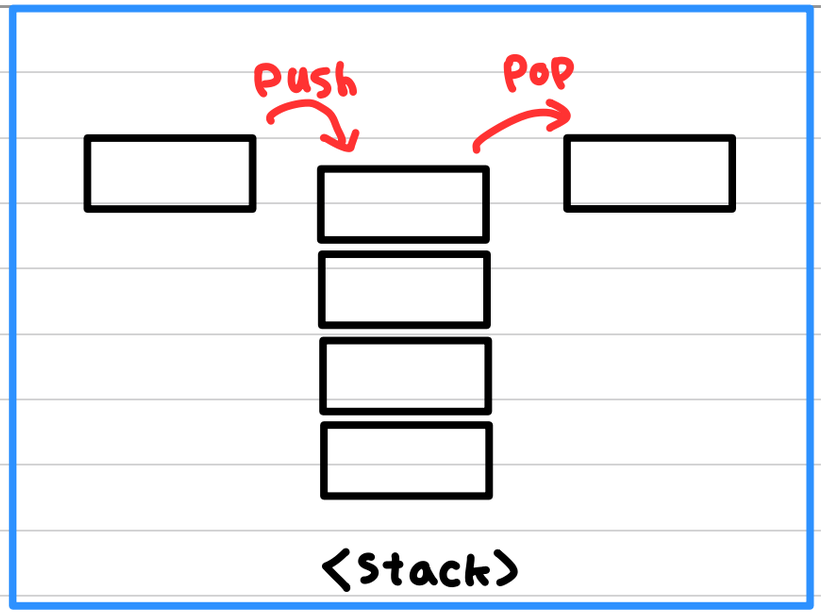
자료구조

I. 스택 (Stack)

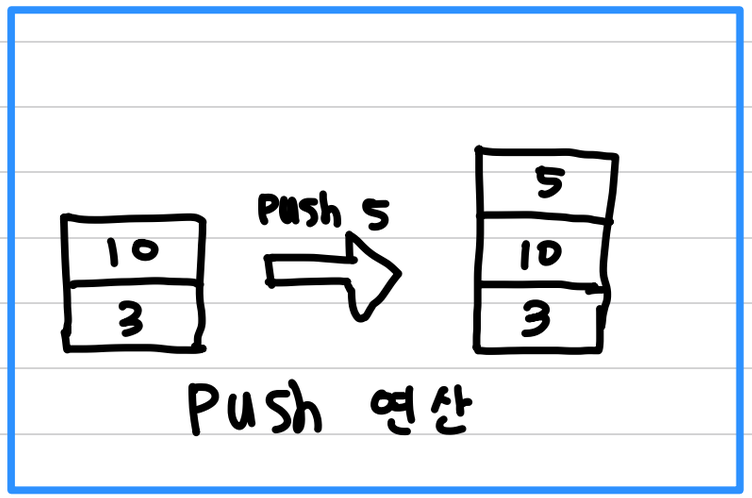
- 한쪽 끝에서만 자료를 넣고 뺄 수 있는 자료구조
- 마지막으로 넣은 것이 가장 먼저 나오기 때문에
  Last In First Out (LIFO) 라고도 한다



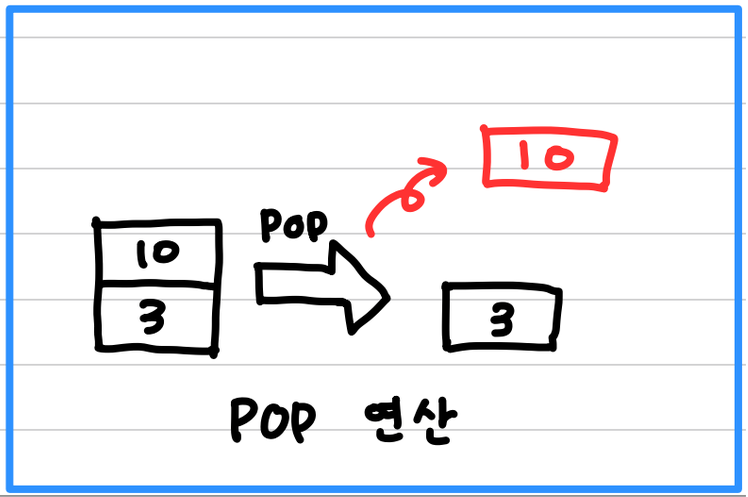
<Stack>

1) 스택의 연산

- Push : 스택에 자료를 넣는 연산
- Pop . 스택에서 자료를 빼는 연산
- top : 스택의 가장 위에 있는 자료를 보는 연산
- empty: 스택이 비어있는지 확인하는 연산
- size: 스택에 저장된 자료의 개수를 알아보는 연산



Push 연산



Pop 연산

* 관련 문제
  - BOJ 10828
  - BOJ 9093
  - BOJ 9012
  - BOJ 1874
  - BOJ 1406

2) 스택의 구현

1. 라이브러리를 사용 → java.util.Stack
2. 일차원 배열로 구현

```
4 usages
int[] stack = new int[10000];
8 usages
int size = 0;
```

```
void push (int data) {
    stack[size] = data;
    size++;
}
```

```
int pop() {
    int data = stack[size - 1];
    stack[size - 1] = 0;
    size--;
    return data;
}
```

```
int size() {
    return size;
}
```

```
boolean isEmpty() {
    if (size == 0)
        return true;
    else
        return false;
}
```

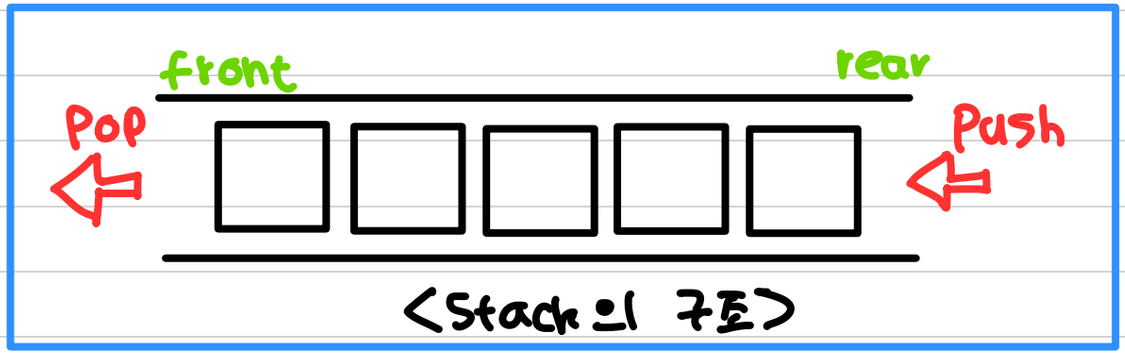
```
int top() {
    return stack[size - 1];
}
```

Ⅱ. 큐 (Queue)

- 한쪽 끝에서만 자료를 넣고 다른 한쪽 끝에서만 뺄 수 있는 자료구조
- 먼저 넣은 것이 가장 먼저 나오기 때문에 First In First Out (FIFO)라고도 한다

1) 큐의 연산

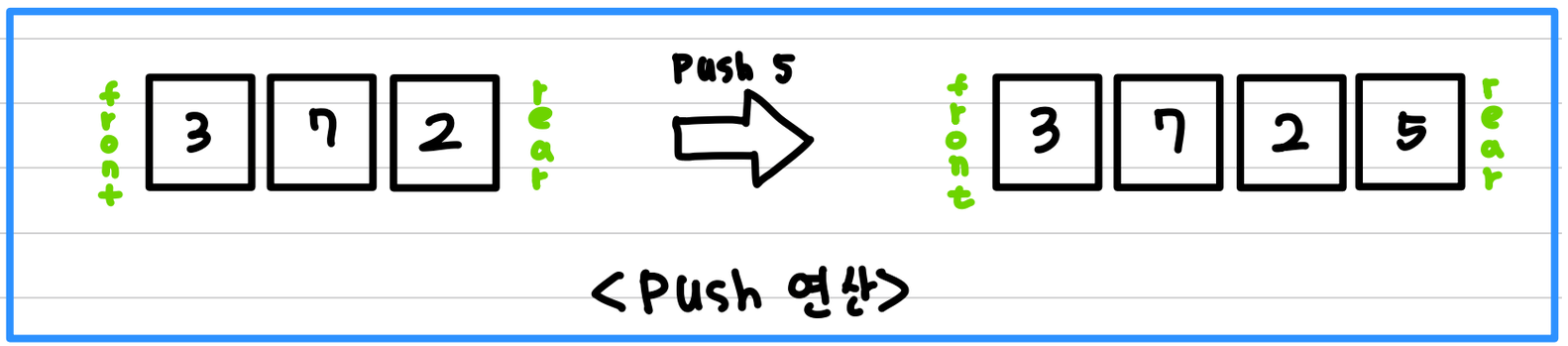
- push : 큐에 자료를 넣는 연산
- pop : 큐에서 자료를 빼는 연산
- front : 큐의 가장 앞에 있는 자료를 보는 연산
- back : 큐의 가장 뒤에 있는 자료를 보는 연산
- empty : 큐가 비어있는지 아닌지를 알아보는 연산
- size : 큐에 저장되어있는 자료의 개수를 알아보는 연산



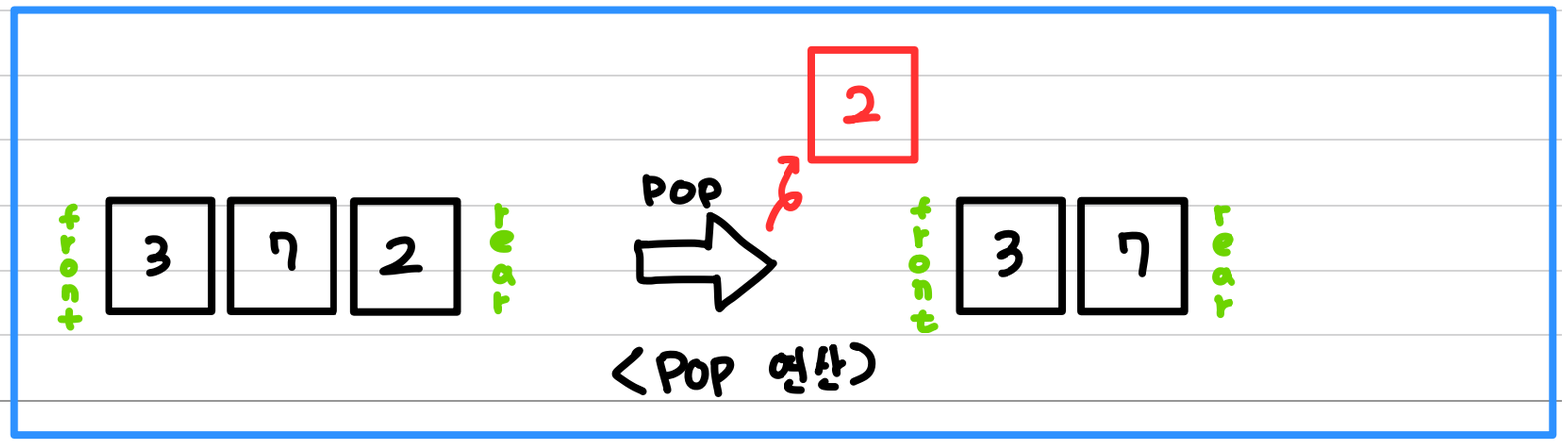
<Stack의 구조>

*관련 문제

- BOJ 10845
- BOJ 1158



<Push 연산>



<Pop 연산>

2) 큐의 구현

1. 라이브러리를 사용한다 → java.util.Queue (강추)
2. 일차원 배열을 사용한 구현(한번 정도 추천)

```
5 usages
int[] queue = new int[10000];
6 usages
int begin = 0;
5 usages
int end = 0;
```

```
void push (int data) {
    queue[end] = data;
    end++;
}
```

```
int pop() {
    int data = queue[begin];
    queue[begin] = 0;
    begin--;
    return data;
}
```

```
int front() {
    return queue[begin];
}
```

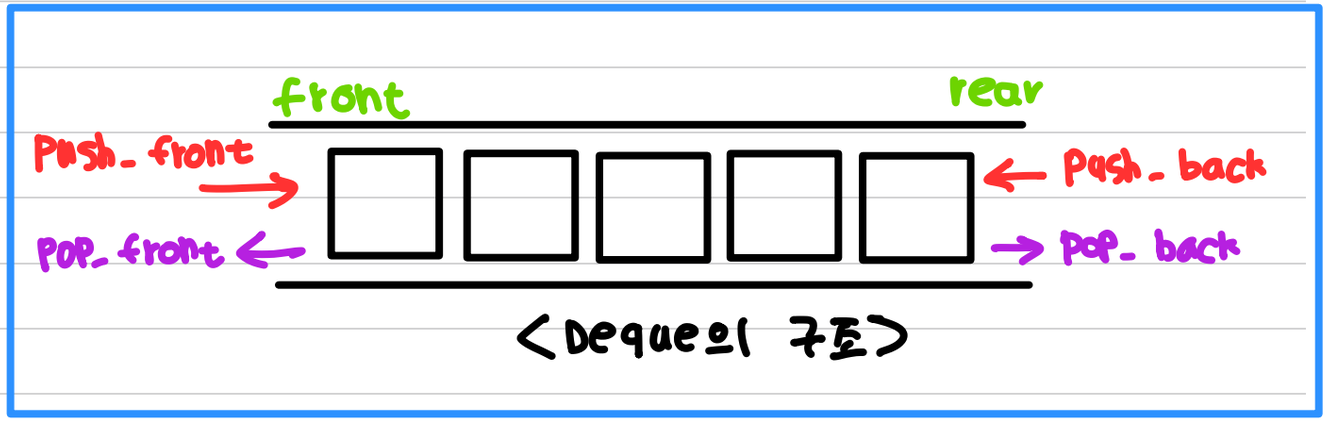
```
int back() {
    return queue[end];
}
```

```
boolean isEmpty() {
    if(begin == end)
        return true;
    else
        return false;
}
```

```
int size() {
    return end - begin;
}
```

Ⅲ. 덱(Deque)

- 양 끝에서만 자료를 넣고 양 끝에서 뺄 수 있는 자료구조
- Double-ended queue 의 약자이다



<Deque의 구조>

1) 덱의 연산

- Push_front : 덱의 앞에 자료를 넣는 연산
- Push_back : 덱의 뒤에 자료를 넣는 연산
- Pop-front : 덱의 앞에서 자료를 빼는 연산
- Pop-back : 덱의 뒤에서 자료를 빼는 연산
- front : 덱의 가장 앞에 있는 자료를 보는 연산
- back : 덱의 가장 뒤에 있는 자료를 보는 연산

2) 덱의 구현

1. 라이브러리 사용 → java.util.Deque

* 관련 문제

- BOJ 10866