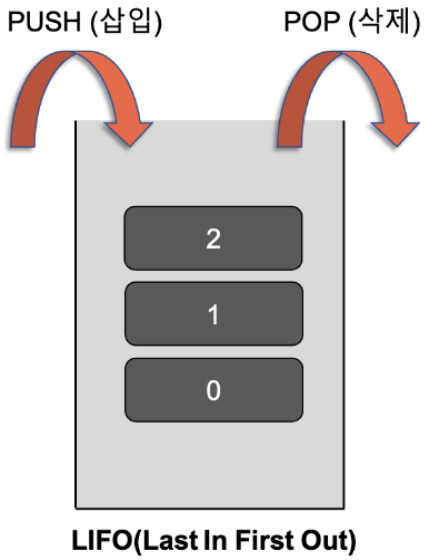
알고리즘 스터디 3 - 스택, 큐, 덱

📌스택

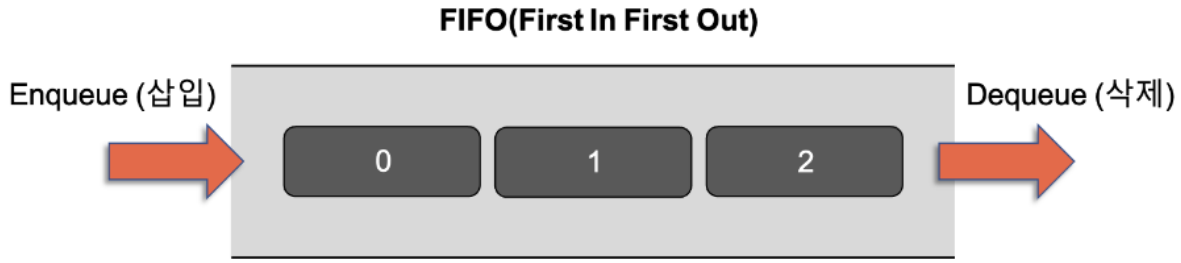


- 데이터를 차곡차곡 쌓아 올린 형태의 자료구조
  - 정해진 방향으로만 쌓을 수 있다
  - top으로 정한 곳을 통해서만 접근 가능
  - 새로 삽입된 자료는 top이 가리키는 가장 맨위에 쌓이게 되고, 자료를 삭제할때도 top을 통해서 삭제가 가능하다.
- 가장 마지막에 삽입된 자료가 가장 먼저 삭제되는 구조
- 그래프의 깊이 우선 탐색(DFS)에서 사용한다.
- 재귀적 함수를 호출할 때 사용한다.

스택 선언

```
#include <stack>  // stack이 들어있는 헤더파일
stack<int> s;     //int형 스택 선언
stack<char>s;     //char형 스택 선언
```

📌큐



- 스택과 다르게 먼저 들어온 것이 먼저 나간다.
- 큐의 한쪽 끝은 프론트(front)로 정하여 삭제 연산만 수행하고, 다른 한쪽 끝은 리어(rear)로 정하여 삽입 연산만 수행한다.
- 그래프의 넓이 우선 탐색(BFS)에서 사용한다.

큐 선언

```
#include <queue>  // queue가 들어있는 헤더파일
queue<int> s;     //int형 큐 선언
queue<char>s;     //char형 큐 선언
```

멤버함수

멤버함수	
q.empty()	비어있으면 true 아니면 false를 리턴
q.size()	원소의 수를 리턴
q.front()	맨 앞 원소 리턴
q.back()	맨 뒤 원소 리턴
q.push(a)	맨 뒤에 원소 a를 추가
q.pop()	맨 앞 원소 삭제

📌덱

- 앞과 뒤에서 데이터를 처리할 수 있는 양방향 자료형
  - 데이터의 삽입과 삭제가 front와 back에서 이루어 질 수 있다.
- 스택처럼 써도 되고 큐처럼 써도 된다.
- vector의 단점을 보완하기 위해 여러개의 메모리 블록을 할당하고 하나의 블록처럼 여기는 기능을 제공한다.
  - 메모리가 부족할때마다 일정한 크기의 새로운 메모리 블록을 할당한다. 이전 원소를 복사하지 않는다.
  - vector는 새로운 원소가 추가 될때 메모리 재할당 후 이전 원소를 복사하는 방식으로 삽입시 성능 저하의 단점이 있다.

덱큐 선언

```cpp
#include <deque>
using namespace std;

int main() {

    /* 생성자 */
    deque<int> dq;
    //deque<int> dq(5); // size = 5만큼 int의 기본값 0으로 초기화
    //deque<int> dq(5, 2); // size = 5 만큼 2로 초기화
```