ILE3-015 List

2차원 선형 데이터 구조

List의 Functions

- Get (front, rear, using Index)
- Insert (front, rear, using Index)
- Delete (front, rear, using Index)

Array List

Memory에서 연속된 구간을 이용
Python의 기본 List는 Array List
Index를 이용한 참조가 많고, rear에만 데이터가 추가/삭제 되는 경우 유리

- Get 연산은 모두 `O(1)`
- Insert는 rear에 대한 연산은 `O(1)`, 기타 연산은 `O(n)`
- Delete는 rear에 대한 연산은 `O(1)`, 기타 연산은 `O(n)`
- Reserve된 영역을 넘어가는 경우, Memory Copy가 일어나 Insert rear도 `O(n)`이 될 수 있음

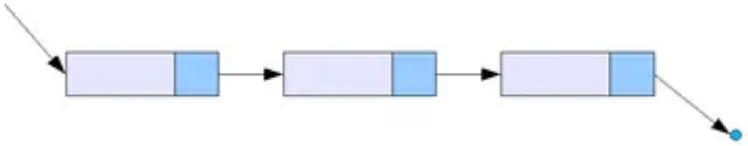
Linked List

값과 함께 다른 값을 참조할 수 있는 Reference로 구성된 2차원 선형 데이터 구조

Single Linked List

단방향으로만 연결이 되어 있는 Linked List
Front, Rear에 대한 연산이 많고, Index를 이용한 참조가 적은 경우 유리
현재 위치에서 다음 데이터에 대해서만 참조/작업이 있는 경우 유리

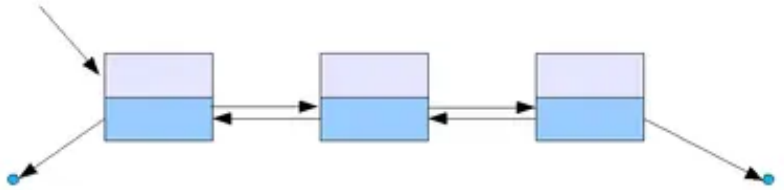
- Front, rear에 대한 연산은 모두 `O(1)`
- 현재 위치에서 다음 Index에 대한 연산은 모두 `O(1)`
- 현재 위치에서 이전 Index에 대한 연산은 모두 `O(n)`
- Index를 이용한 연산은 모두 `O(n)`



Double Linked List

양방향으로 연결이 되어 있는 Linked List
현재 위치에서 양방향으로 참조/작업이 있는 경우 유리

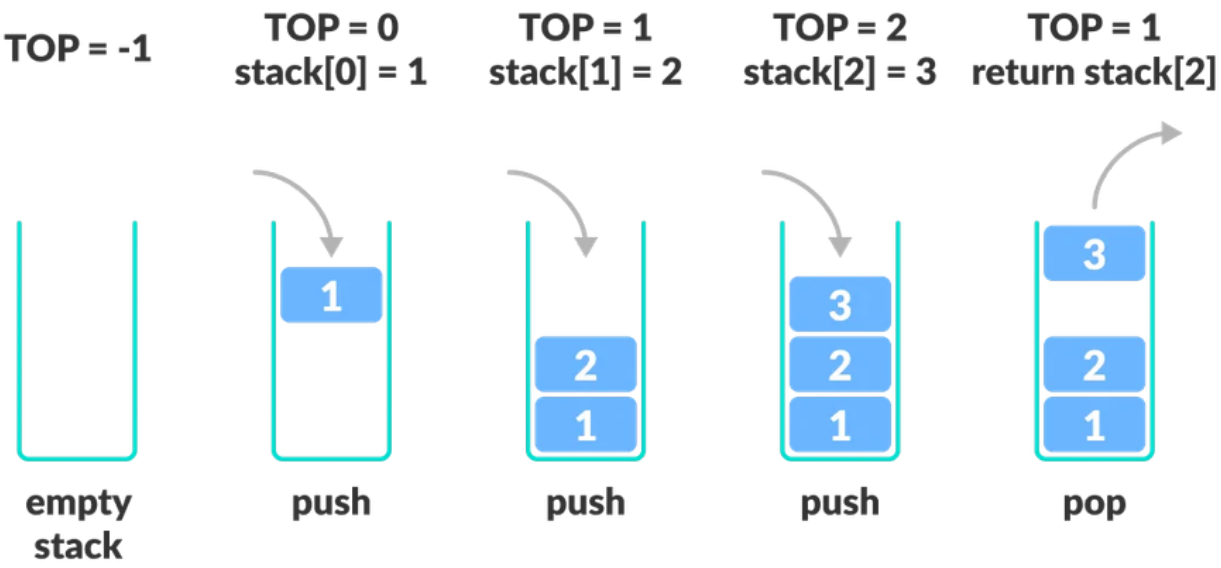
- Front, rear에 대한 연산은 모두 `O(1)`
- 현재 위치에서 다음 Index에 대한 연산은 모두 `O(1)`
- 현재 위치에서 이전 Index에 대한 연산은 모두 `O(1)`
- Index를 이용한 연산은 모두 `O(n)`



List의 활용

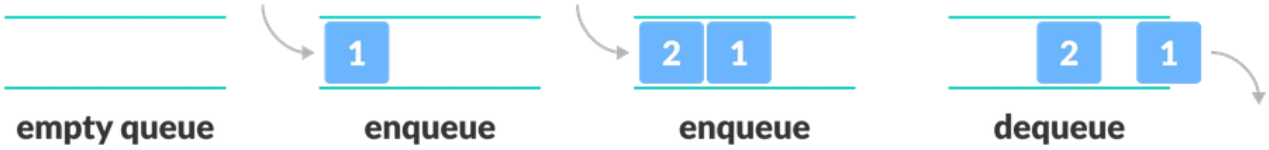
Stack

LIFO (Last In, First Out)를 위한 데이터 구조



Queue

FIFO (First In, First Out)를 위한 데이터 구조



과제

Single Linked List를 구현하고, python에 내장된 list와 0번째 Index에 데이터를 1천만번 Insert하는 시간을 비교하세요

Python 내장 List에서 0번째 Index에 1천만번 Insert

```
l = []
for i in range(10000000):
    l.insert(0, i)
```