Algorithm Study

스택, 큐

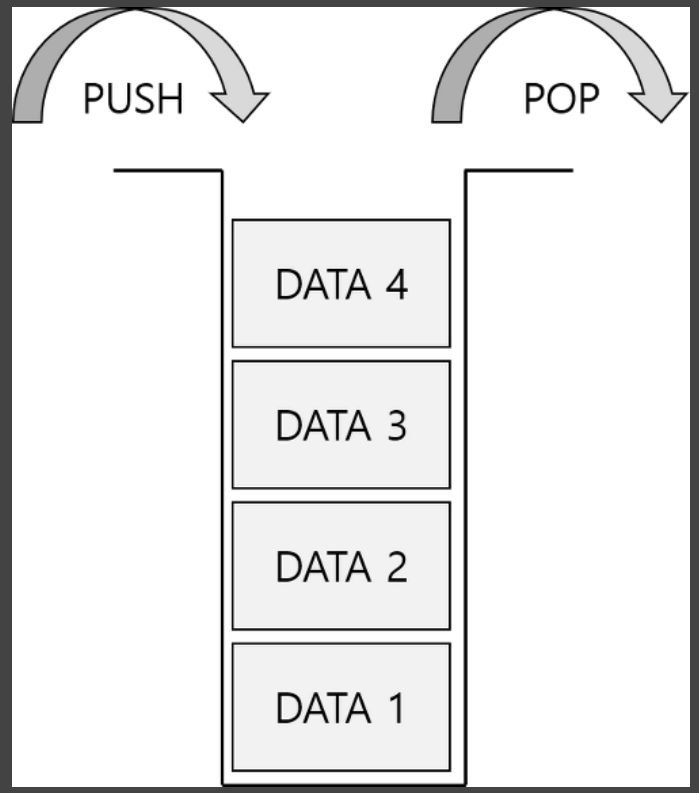
2022.09.17

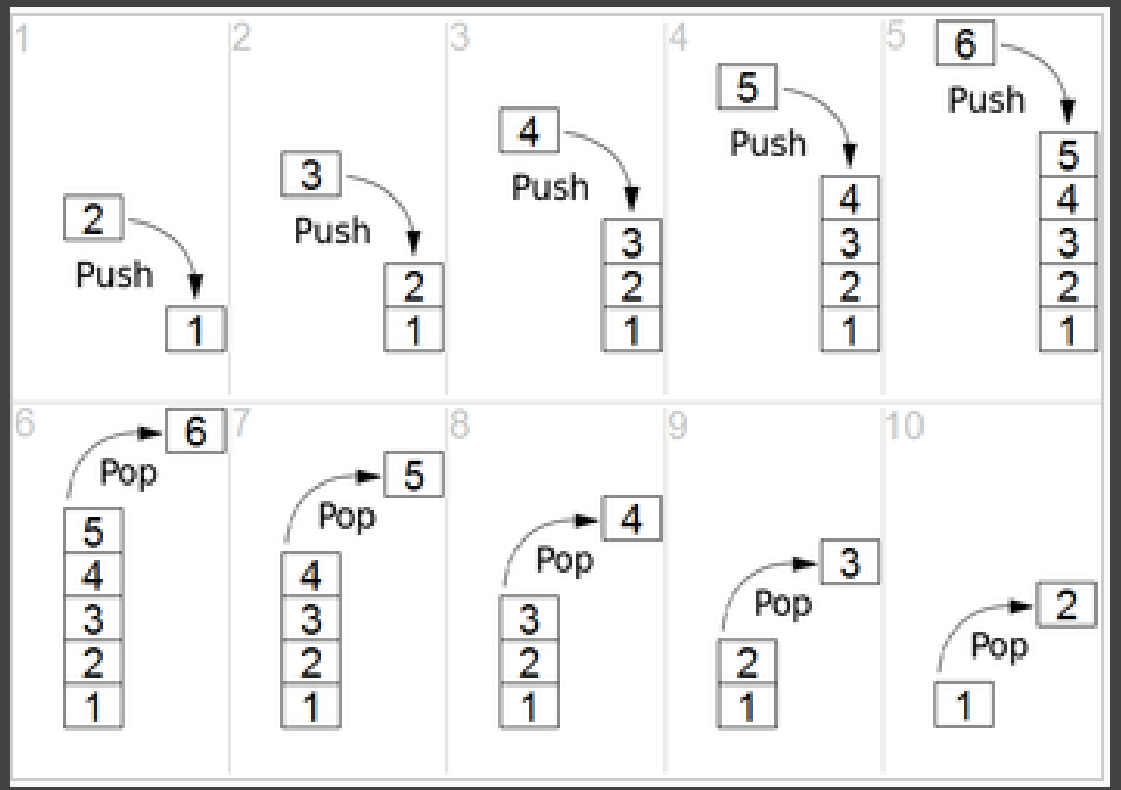
방지호

목차 Contents

(1) Stack 개념

LIFO





(1) Stack 개념

Class를 만들어 stack 구현하는 방법

```
MAX_ITEMS = 100

class StackType:
    def __init__(self):
        self.info = []
        self.Top = -1

    def is_empty(self):
        return (self.Top == -1)

    def is_full(self):
        return (self.Top == MAX_ITEMS-1)

    def push(self, item):
        self.Top += 1
        self.info.append(item)

    def pop(self):
        self.Top -= 1
        self.info.pop()

    def top(self):
        if (self.Top == -1):
            return "Stack is Empty"
        return self.info[self.Top]
```

Stack 구현의 핵심 : item의 위치를 point

(1) Stack 개념

Python 내장 함수를 사용해서 stack 구현하는 방법

stack - push

: append 메소드를 이용해 리스트의 가장 마지막에 원소를 넣음.

```
stack = []
stack.append(1)
stack.append(2)
stack.append(3)
stack.append(4)
```

stack - pop

: pop 메소드를 이용하여 리스트의 가장 마지막 원소를 제거

```
stack = [1,2,3,4]
top = stack.pop()

print(top)
```

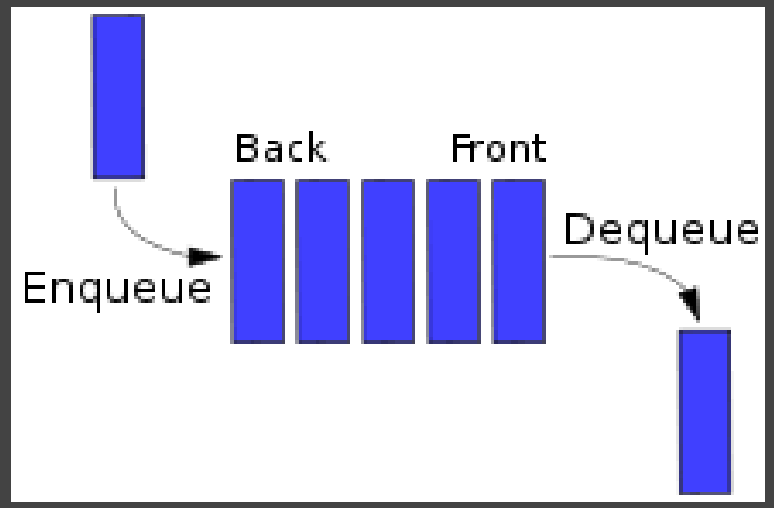
stack - top

: append 메소드를 이용해 리스트의 가장 마지막에 원소를 넣음.

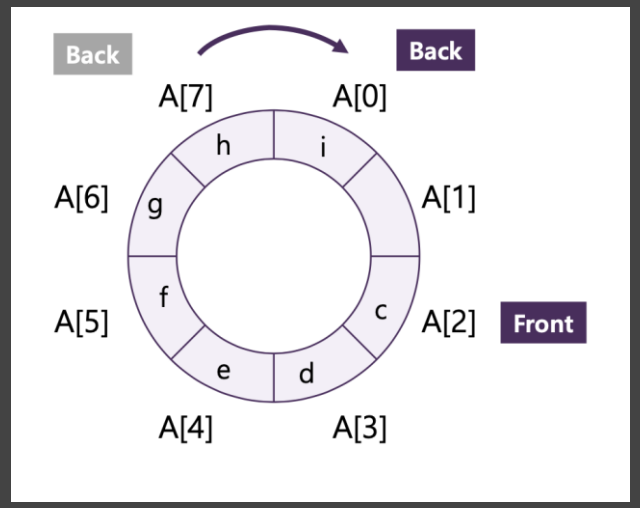
```
stack = [1,2,3,4]
top = stack[-1]

print(top)
```

(2) Queue 개념

FIFO

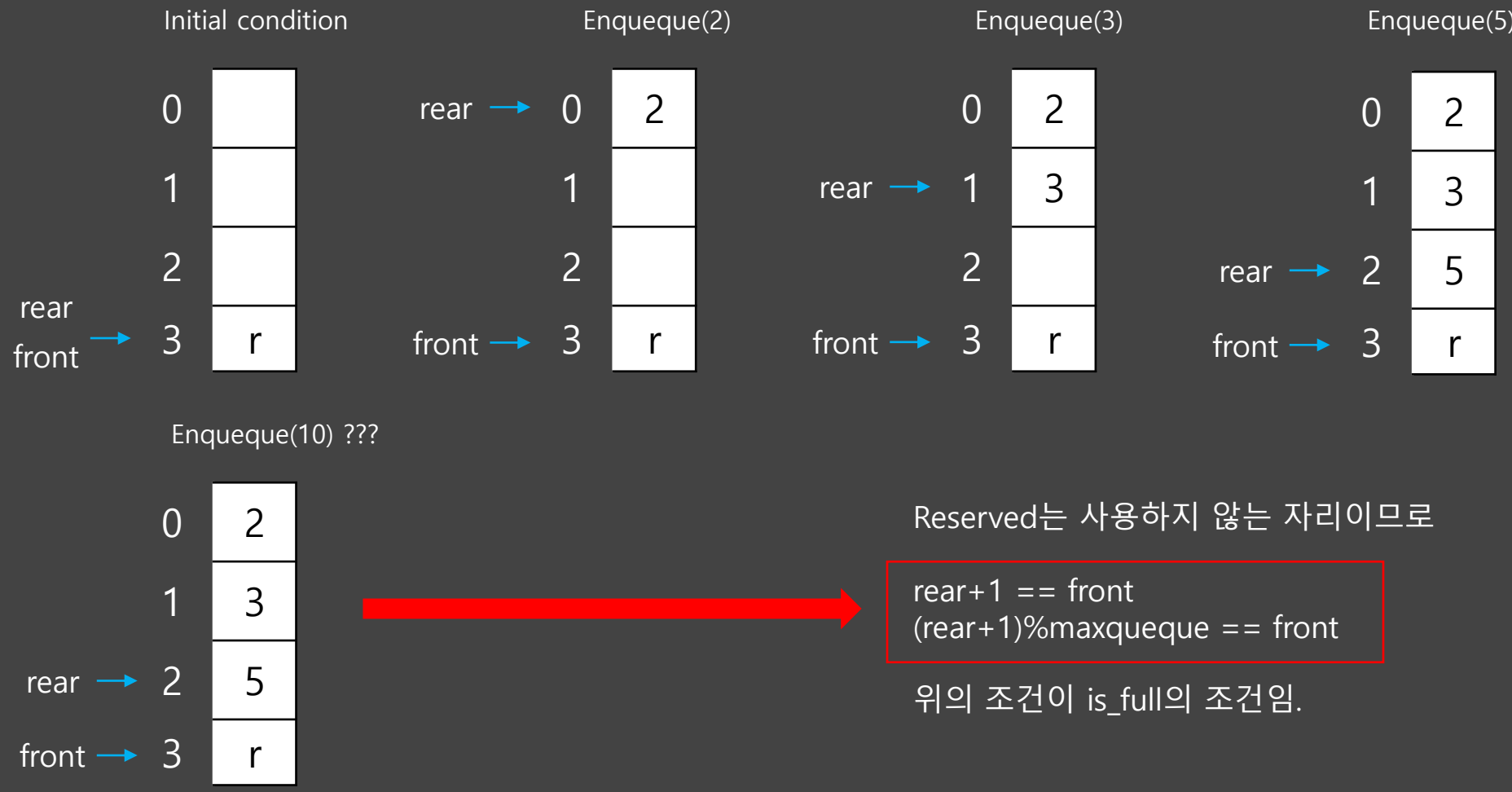


→ Stack과 달리 Queue에서는
Back과 front가 계속 커지기만 하므로
문제가 발생할 수 있음.



→ Circular Queue 방식으로
문제를 해결함.

(2) Queue 개념 (is_full)

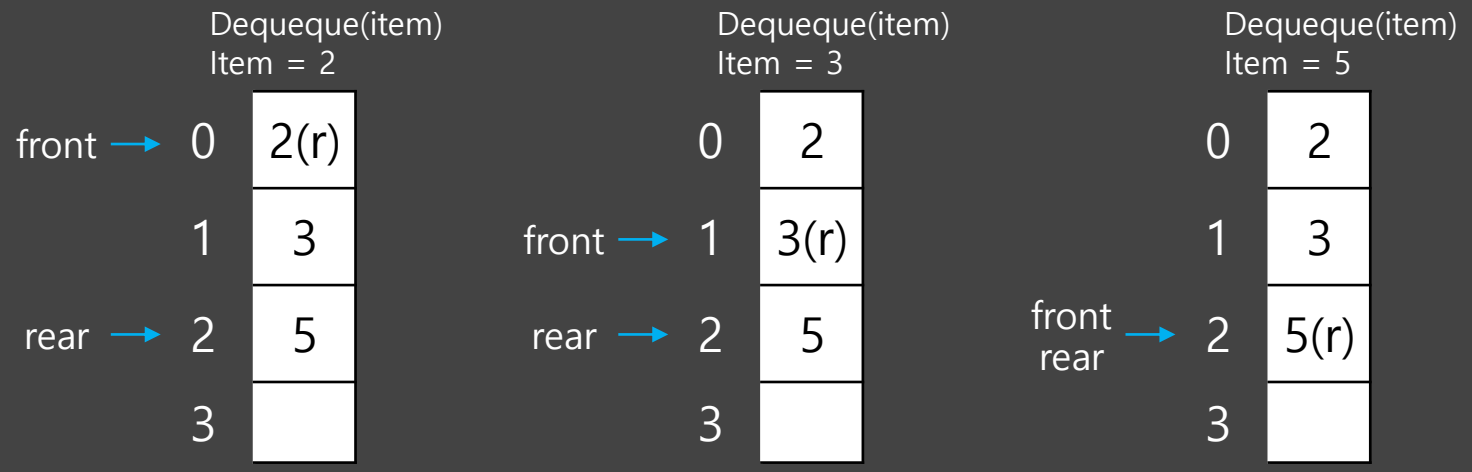


Reserved는 사용하지 않는 자리이므로

rear+1 == front
(rear+1)%maxqueque == front

위의 조건이 is_full의 조건임.

(2) Queue 개념 (is_empty)



rear == front

위의 조건이 is_empty의 조건임.

(2) Queue 개념

Class를 만들어 queue 구현하는 방법

```
MAX_ITEMS = 100

class QueueType():
    def __init__(self):
        self.info = []
        self.rear = MAX_ITEMS-1
        self.front = MAX_ITEMS-1

    def enqueue(self, data):
        self.rear = (self.rear + 1) % MAX_ITEMS
        self.info.insert(self.rear, data)

    def dequeue(self):
        self.front = (self.front + 1) % MAX_ITEMS
        return self.info[self.front]

    def is_empty(self):
        return (self.rear == self.front)

    def is_full(self):
        return ( (self.rear + 1) % MAX_ITEMS == self.front)

    def make_empty(self):
        self.rear = MAX_ITEMS-1
        self.front = MAX_ITEMS-1
```

Index의 시작을 0으로 만들기 위함.

(2) Queue 개념

Python의 list를 사용해서 queue 구현하는 방법 (데이터가 오른쪽에서 들어오고 왼쪽으로 나감.)

queque - Enqueque

: append 메소드를 이용

```
queque = []
queque.append(1)
>> [1]
queque.append(2)
>> [1, 2]
Queque.append(3)
>> [1, 2, 3]
queque.append(4)
>> [1, 2, 3, 4]
```

queque - Dequeuqe

: pop 메소드를 이용

```
queque = [1, 2, 3, 4]
queque.pop(0)
>> 1
queque.pop(0)
>> 2
Queque.pop(0)
>> 3
Queque.pop(0)
>> 4
```

(2) Queue 개념

Python의 list를 사용해서 queue 구현하는 방법 (데이터가 왼쪽에서 들어오고 오른쪽으로 나감.)

queque - Enqueque

: insert 메소드를 이용

```
queque = []
queque.insert(0, 1)
>> [1]
queque.insert(0, 2)
>> [2, 1]
Queque.insert(0, 3)
>> [3, 2, 1]
Queque.insert(0, 4)
>> [4, 3, 2, 1]
```

queque - Dequeuqe

: pop 메소드를 이용

```
queque = [4, 3, 2, 1]
queque.pop()
>> 1
queque.pop()
>> 2
Queque.pop()
>> 3
Queque.pop()
>> 4
```

(2) Queue 개념

Collections 모듈의 deque를 사용해서 queue 구현하는 방법 (데이터가 오른쪽에서 들어오고 왼쪽으로 나감.)

```
>> from collections import deque
>> queue = deque([1, 2, 3])
```

queque - Enqueque

: append 메소드를 이용

```
queque.append(4)
>> [1, 2, 3, 4]
queque.append(5)
>> [1, 2, 3, 4, 5]
```

queque - Dequeuqe

: pop 메소드를 이용

```
queque.popleft()
>> 1
queque.popleft()
>> 2
```

```
>> queue
deque([3, 4, 5])
```

(2) Queue 개념

queue 모듈의 Queue 클래스를 사용해서 queue 구현하는 방법

```
>> from queue import Queue
>> que = Queue()
```

queque - Enqueque

: put 메소드를 이용

```
queque.put(1)
queque.put(2)
Queque.put(3)
Queque.put(4)
```

queque - Dequeuqe

: get 메소드를 이용

```
queque.get()
>> 1
queque.get()
>> 2
Queque.get()
>> 3
```

(3) 문제집

Week 5 : 스택, 큐

https://www.acmicpc.net/workbook/view/12714

전부 풀이 업로드
너무 오래 걸리는 문제는 풀지 않기