알고리즘과 자료구조 워크북

교과목명 : 알고리즘과 자료구조
차시명: 10차시 큐와 스택

◈ 담당교수: 신일훈 **(서울과학기술대학교)**

◉ 세부목차
- 큐의 개념 및 연산
- 큐 구현
- 원형 양방향 연결리스트 구현

학습에 앞서

■ 학습개요

큐와 스택은 대표적인 선형 자료구조들이다. 큐는 양쪽 방향이 뚫려 있는 파이프 구조이며, 아이템의 추가가 후면으로 일어나고 아이템의 제거는 전면에서 발생하여 선입선출의 특성을 갖는다. 본 강에서는 큐의 개념 및 주요 연산을 이해하고 원형 연결리스트를 활용하여 큐를 구현한다.

■ 학습목표

1	큐의 개념 및 연산을 이해한다.
2	큐를 구현할 수 있다.
3	원형 양방향 연결리스트를 구현할 수 있다.

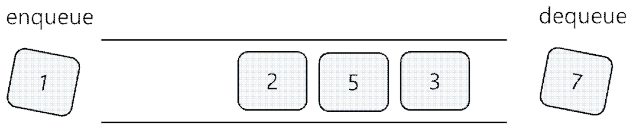
■ 주요용어

용어	해설
큐	대표적인 선형 자료구조로서, 양쪽 방향이 뚫려 있는 파이프 구조이며, 아이템의 추가가 후면으로 일어나고 아이템의 제거는 전면에서 발생하여 선입선출의 특성을 갖는다.
선입선출(FIFO)	먼저 들어온 아이템이 가장 먼저 나온다는 의미의 용어이다. 큐의 대표적인 특성이다.

학습하기

1. 큐의 개념
   - 연결 리스트와 유사한 선형 자료구조
   - 양쪽이 뚫려 있는 파이프의 형태
   - 새로운 아이템의 추가는 큐의 후면(또는 전면)으로만 가능

- 아이템의 제거는 큐의 전면(또는 후면)으로만 가능



- FIFO(First In First Out)의 특성
- 추가, 제거, 검색, 크기 반환 등이 가능함
- CPU 스케줄링(FCFS, round-robin, …) 또는 그래프 탐색(너비우선탐색: breath first search) 등을 수행할 때 활용됨

2. 큐 구현

가. 클래스 Queue 정의

- 원형 양방향 연결리스트를 활용하여 구현.
- enqueue(): insert_back()으로 구현
- dequeue(): delete_front()로 구현
- 멤버 변수
  - queue : 아이템들을 저장할 연결리스트 객체이며, 객체 생성 시에 빈 리스트로 초기화된다.
  - count : queue에 저장된 아이템의 개수
- 메서드
  - 생성자
  - enqueue()
  - dequeue()
  - get_size()
  - print_queue()
  - search()

나. 클래스 Queue 생성자 코드

```
import CList

class Queue:
        def __init__(self):
                self.queue = CList.CList()
                self.count = 0
```

- 원형 연결리스트 클래스가 CList.py에 구현되어 있다고 가정함.
- CList.py에 필요한 메서드가 모두 구현되어 있다고 가정한다.
- 큐 구현에 CList를 활용할 것이므로 해당 파일을 import 한다.
- 큐가 생성될 때 비어 있는 상태이므로 queue를 빈 원형연결리스트로 생성하고 count를 0으로 초기화한다.

다. 클래스 Queue의 enqueue() 메서드 코드

- 큐의 후면에 삽입을 수행하는 메서드이다.
- insert_back() 메서드는 원형 연결리스트의 후면에 삽입하는 메서드이다.
- insert_back() 메서드를 호출하여 후면에 아이템을 삽입하고 count를 1 증가시킨다.

```
        def enqueue(self, item):
                self.queue.insert_back(item)
                self.count += 1
```

라. 클래스 Queue의 dequeue() 메서드 코드

- 모든 노드들의 item 값들을 출력하는 메서드이다.

```
        def dequeue(self):
                if (self.count > 0):
                        self.count -= 1
                        item = self.queue.delete_front()
                        return item
                 return None
```

- self.head가 0이면 None을 반환한다.
- self.head가 0보다 크면, count를 1 감소시킨다.
- delete_front()는 원형 연결리스트의 전면에서 아이템을 제거하여 반환하는 메서드이다.
- delete_front()를 호출하여 전면의 아이템을 제거하고 해당 아이템을 반환한다.

마. 클래스 Queue의 print() 메서드 코드

```
        def print(self):
                self.queue.print_list()
```

- 원형 연결리스트의 모든 노드들을 출력하는 print_list()를 호출하여 큐의 아이템들을 출력한다.

바. 클래스 Queue의 get_size() 메서드 코드

- 큐에 저장된 아이템의 개수를 반환하는 메서드이다.

```
        def get_size(self):
                return self.count
```

- self.count에는 아이템의 개수가 저장되어 있으므로 이를 반환한다.

사. 테스트 코드 및 실행 결과

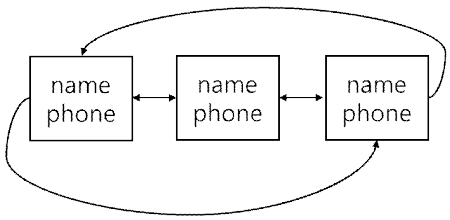
```
if __name__ == '__main__':
        q = Queue()
        q.enqueue('mango')
        q.enqueue('apple')
        q.enqueue('orange')
        q.print ()
        for i in range(4):
                print(q.dequeue())
        q.enqueue('mango')
        q.enqueue('apple')
        q.print()
```

- 비어 있는 큐를 생성한 후, mango, apple, orange를 차례로 삽입한다.
- 큐를 출력하면 가장 처음에 추가한 mango가 제일 먼저 나와야 한다.
- for문을 반복하며 dequeue()를 4번 호출하면, mango, apple, orange가 차례로 제거되고 마지막에는 삭제가 실패한다.
- 이후 enqueue() 메서드를 호출하여 mango, apple을 삽입하고 큐를 출력하면 mango, apple의 순서로 출력되어야 한다.

- 또한 이 테스트 코드는 __main__이 만족될 때 실행되므로, 이 파이썬 파일을 import할 때는 실행되지 않는다.
- 다음은 이에 대한 실행 결과이다.

```
In [46]: runfile('D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/자료구조/Queue.py',
wdir='D:/data/lecture/파이썬으로배우는자료구조와알고리즘/
code/자료구조')
mango  <=>  apple  <=>  orange
mango
apple
orange
None
mango  <=>  apple
```

3. 원형 양방향 연결리스트 구현
- 데이터를 저장하는 노드(node)와 노드를 연결하는 링크로 구성됨
- 모든 노드들이 양쪽 방향 링크를 통해 원형으로 연결됨



- 가장 마지막 노드인 tail 노드의 next 링크는 첫 번째 노드인 head 노드를 가리킴.
- head 노드의 prev 링크는 tail 노드를 가리킴

가. 클래스 CNode 정의
- 리스트를 구성하는 하나의 노드에 해당
- 멤버 변수
  - item: 노드의 데이터를 저장
  - next: 다음 노드를 가리키는 링크
  - prev: 이전 노드를 가리키는 링크
- 메서드
  - 생성자

나. 클래스 CNode 코드

```
class CNode:
        def __init__(self, item, prev=None, next=None):
                self.item = item
                self.prev = prev
                self.next = next
```

- 생성자는 세 개의 인자, item, prev, next를 전달받아 이들 값으로 멤버 변수들을 초기화함.

다. 클래스 CList 정의
- 양방향 리스트를 나타냄
- 멤버 변수
  - head: 첫째 노드를 가리킴
- 메서드
  - 생성자

- insert_front()
- delete_front()
- print_list()

라. 클래스 CList 생성자 코드

```
class CList:
        def __init__(self):
                self.head = None
```

- 리스트가 생성될 때 비어 있는 상태이므로 head를 None으로 초기화함.

마. 클래스 CList의 insert_back() 메서드 코드

- 후면 삽입을 수행하는 메서드이다.

```
        def insert_back(self, item):
                cnode = CNode(item, None, None)
                if (self.head == None):
                        cnode.next = cnode
                        cnode.prev = cnode
                        self.head = cnode
                else:
                        first = self.head
                        last = first.prev
                        cnode.next = first
                        cnode.prev = last
                        first.prev = cnode
                        last.next = cnode
```

- 추가할 item을 인자로 전달받고, CNode를 호출하여 item을 저장할 노드를 생성한다.
- 기존에 리스트가 비어 있는 경우에는 self.head를 생성한 노드로 설정하고 생성한 노드의 next, prev는 모두 자신을 가리키도록 하고 종료한다.
- 리스트가 비어있지 않은 경우에는 생성한 노드의 next가 기존의 첫째 노드(first)를 가리키도록 한다.
- 생성한 노드의 prev는 기존의 마지막 노드(last)를 가리키도록 한다.
- first의 prev는 생성한 노드를 가리키고, last의 next도 생성한 노드를 가리키도록 한다.
- 생성된 노드는 마지막 노드이므로 self.head는 수정되지 않는다. (이를 제외한 나머지 코드는 insert_front()와 동일함.)

바. 클래스 CList의 delete_front() 메서드 코드

- 첫째 노드를 삭제하는 메서드이다.

```
    def delete_front(self):
        if self.head == None:
            return None

        target = self.head
        if (target.next == target):
            self.head = None
            item = target.item
            del(target)
            return item
        else:
            first = target.next
            last = target.prev
            first.prev = last
            last.next = first
            self.head = first
            item = target.item
            del(target)
            return item
```

- 리스트가 비어 있는 경우에는 None을 반환한다.
- self.head가 첫째 노드를 가리키므로 이를 target에 저장한다.
- target이 유일한 노드이면, self.head를 None으로 하고 target의 아이템을 반환한다. 이 때 target이 차지하는 메모리를 해제할 수 있다. (del() 함수)
- target이 유일한 노드가 아니면, 기존의 둘째 노드인 first의 prev가 기존의 마지막 노드인 last를 가리키도록 하고 last의 next는 first를 가리키도록 한다. => target 노드가 리스트에서 분리됨.
- target의 아이템을 반환한다. 이 때 target이 차지하는 메모리를 해제할 수 있다. (del() 함수)

.

사. 클래스 CList의 print_list() 메서드 코드

- 모든 노드들의 item 값들을 출력하는 메서드이다.

```
    def print_list(self):
        if self.head == None:
            print('empty')
            return
        p = self.head
        while p.next != self.head :
            print(p.item, ' <=> ', end= ' ' )
            p = p.next
        print(p.item)
        return
```

- self.head가 만약 None이면 리스트가 비어 있으므로 empty를 출력하고 바로 종료한다.
- self.head가 첫째 노드를 가리키므로 이를 시작 노드로 해서 노드가 더 이상 없을 때까지 노드에 저장된 item을 출력한다.
- 출력 후에는 타겟 노드(p)를 다음 노드(p.next)로 수정하여 작업을 반복한다.
- 현재 노드의 다음 노드가 존재할 때는 (if문), item을 출력하고 나서 화살표도 함께 출력한다.

아. 테스트 코드 및 실행 결과

```
if __name__ == '__main__':
        c = CList()
        c.insert_back('mango')
        c.insert_back('orange')
        c.insert_back('apple')
        c.print_list()
        for count in range(4) :
                print(c.delete_front())
                c.print_list()
```

- 비어 있는 리스트를 생성한 후, mango, orange, apple을 차례로 후면에 삽입한다.
- 리스트를 출력하면 가장 먼저 추가한 mango가 제일 먼저 나와야 한다.
- for 문을 반복하며 delete_front()를 4번 호출하면, mango, orange, apple이 차례로 제거되고 마지막 호출은 실패해야 한다.
- 다음은 이에 대한 실행 결과이다.

```
mango  <=>  orange  <=>  apple
mango
orange  <=>  apple
orange
apple
apple
empty
None
empty
```

연습문제

1. 선형 자료구조로서 FIFO의 특성을 갖는 자료구조는?.

정답 : 큐

해설 : 큐는 선형 자료구조로서 후면으로 데이터가 추가되고 전면에서 데이터가 제거되므로 FIFO의 특성을 갖는다.

2. 원형 연결리스트로 큐를 구현한 경우, 큐에 새로운 아이템을 추가하는 enqueue() 메서드의 최악의 시간복잡도를 구하시오.

정답 : O(1)

해설 : 아이템의 추가가 후면에서 발생하며, 원형 연결리스트를 사용하면 tail 노드를 한번에 찾을 수 있다. 따라서 큐에 저장되어 있는 아이템의 개수와 관계없이 필요한 연산의 수가 일정하므로 시간복잡도는 O(1)이다.

3. 원형 연결리스트로 큐를 구현한 경우, enqueue() 메서드의 알고리즘을 의사코드 형태로 표현하시오.

정답 :

```
def enqueue(self) :
        self.queue.insert_back()
```

```
        self.count += 1          # 없어도 됨
```

해설 : 후면에 아이템을 추가하므로 insert_back() 메서드를 호출하면 된다. 만약 큐의 현재 아이템 개수를 관리하고 있다면, 해당 변수를 1 증가하면 된다.

정리하기

1. 큐는 선형 자료구조로서 후면으로 데이터가 추가되고 전면에서 데이터가 제거되므로 FIFO의 특성을 갖는다.
2. 큐를 구현할 수 있다.
3. 큐는 FCFS, round robin 과 같은 CPU 스케줄링 정책을 구현할 때나 그래프의 너비우선탐색을 구현할 때 활용된다.

참고자료

- 파이썬과 함께하는 자료구조의 이해, 양성봉, 2021, 생능출판

다음 차시 예고

- 선형자료구조 중 하나인 스택에 대해 학습한다.