10강

알고리즘과 자료구조

큐와 스택 1

서울과학기술대학교 신일훈 교수

한국방송통신대학교 프리임칼리지

학습목표

1. 큐의 개념 및 연산을 이해한다.
2. 큐를 구현할 수 있다.
3. 원형 연결리스트를 구현할 수 있다.
4. 다양한 큐 및 큐의 활용분야를 이해한다.

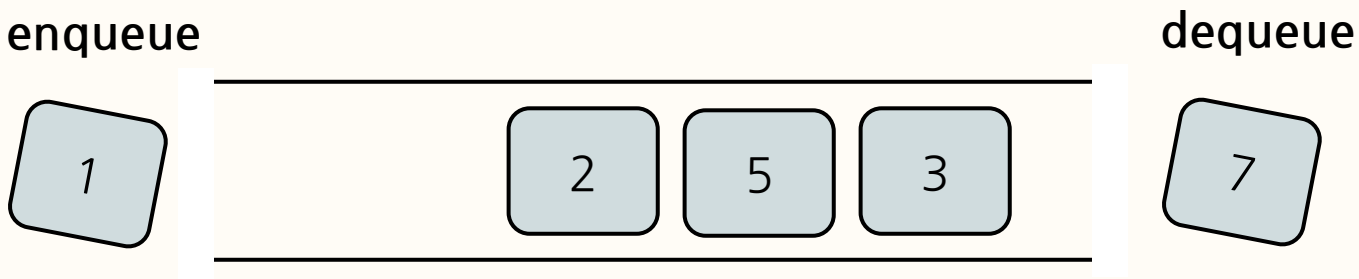
01

큐 개념

1. 큐(queue) 개념

개념

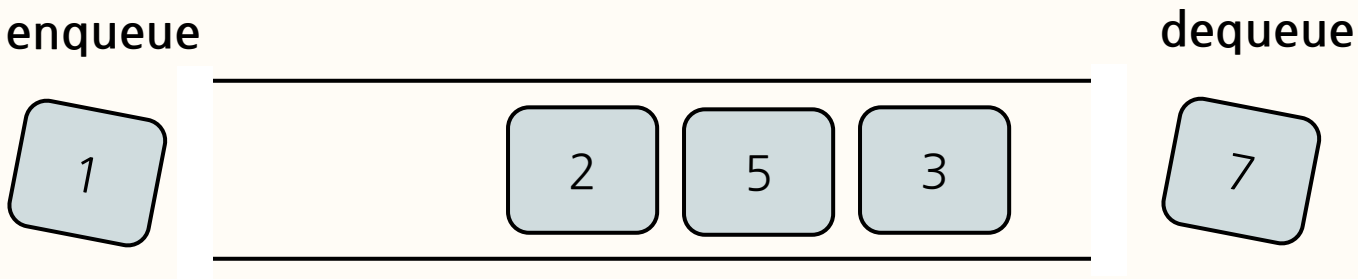
- 연결 리스트와 유사한 선형 자료구조
- 양쪽이 뚫려 있는 파이프의 형태
- 새로운 아이템의 추가는 큐의 후면(또는 전면)으로만 가능
- 아이템의 제거는 큐의 전면(또는 후면)으로만 가능



1. 큐(queue) 개념

❑ 특성

- FIFO(First In First Out)의 특성



1. 큐(queue) 개념

❑ 주요 연산

- 추가(enqueue)
- 제거(dequeue)
- 검색
- 크기반환

1. 큐(queue) 개념

❑ 활용

- CPU 스케줄링 (FCFS, round-robin, …)
- 그래프 탐색 (너비우선탐색: breath first search)
- …

02

큐 구현

2. 큐 구현

- 원형 양방향 연결리스트로 구현 가능
  - enqueue() : insert_back()
  - dequeue() : delete_front()
  - …

2. 큐 구현

클래스 Queue 정의

- 멤버 변수
  - queue : 아이템들을 저장할 연결리스트 객체. 객체 생성 시에 빈 리스트로 초기화
  - count : queue에 저장된 아이템의 개수

2. 큐 구현

- 클래스 Queue 정의
  - 메서드
    - 생성자
    - enqueue()
    - dequeue()
    - get_size()
    - print_queue()
    - search()

2. 큐 구현

클래스 Queue 정의 (생성자)

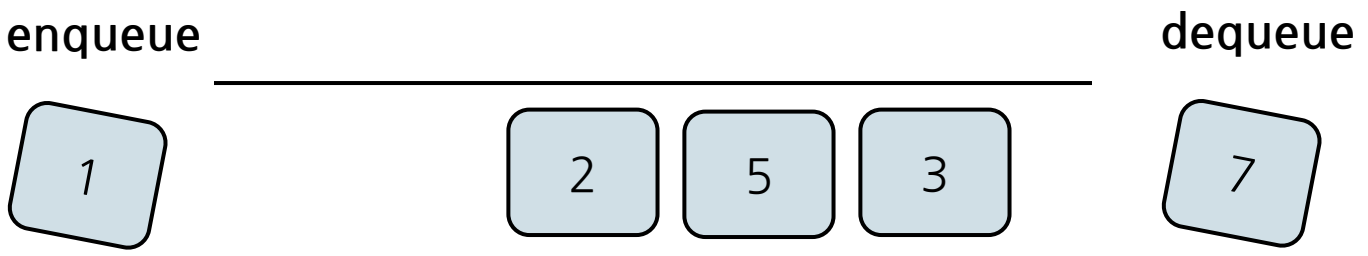
```
import CList


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

2. 큐 구현

클래스 Queue 정의 (enqueue())

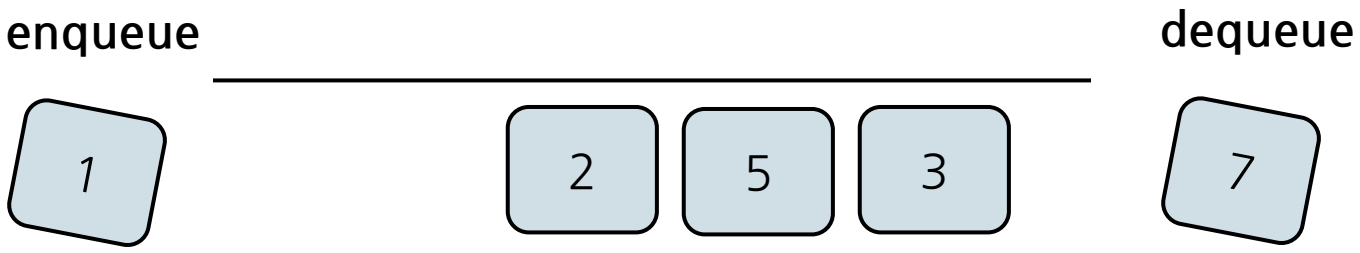
```
class Queue :
    def enqueue(self, item):
        self.queue.insert_back(item)
        self.count += 1
```



2. 큐 구현

클래스 Queue 정의 (dequeue())

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



2. 큐 구현

클래스 Queue 정의 (print())

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

2. 큐 구현

클래스 Queue 정의 (get_size())

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

2. 큐 구현

Queue 테스트

```
if __name__ == '__main__':
    q = Queue()

    q.enqueue('mango')
    q.enqueue('apple')
    q.enqueue('orange')
    q.print ()
```

2. 큐 구현

Queue 테스트

```
if _name_ == '_main_':
    ...
    for i in range(4):
        print(q.dequeue())

    q.enqueue('mango')
    q.enqueue('apple')
    q.print()
```

2. 큐 구현

Queue 테스트 실행

```
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
```

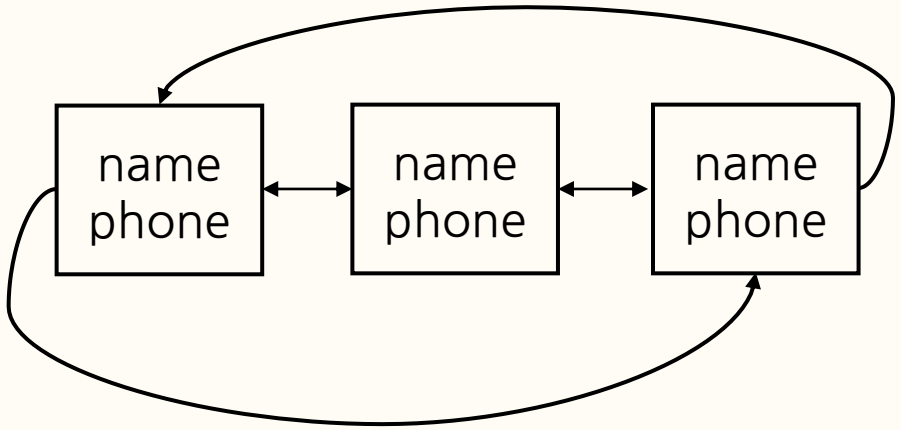
03

원형 양방향 연결 리스트 구현

3. 원형 양방향 연결 리스트 구현

개념

- 모든 노드들이 양쪽 방향 링크를 통해 선형으로 연결됨



- tail 노드의 next 링크는 head 노드를 가리킴
- head 노드의 prev 링크는 tail 노드를 가리킴

3. 원형 양방향 연결 리스트 구현

클래스 CNode 정의

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

3. 원형 양방향 연결 리스트 구현

- 클래스 CList 정의
  - 멤버 변수
    - head (리스트의 첫 노드를 가리킴)
  - 메서드
    - 생성자
    - insert_back()
    - delete_front()
    - print_list()

3. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (생성자)

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

3. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (insert_back())

```
class CList:
    def insert_back(self, item):
        cnode = CNode(item, None, None)
        if (self.head == None):
            cnode.next = cnode
            cnode.prev = cnode
            self.head = cnode
```

3. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (insert_back())

```
class CList:
    def insert_back(self, item):
        ...
        else:
            first = self.head
            last = first.prev
            cnode.next = first
            cnode.prev = last
            first.prev = cnode
            last.next = cnode
```

3. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (delete_front())

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

3. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (delete_front())

```
class CList:
    def delete_front(self):
        ...
        target = self.head
        if (target.next == target):
            self.head = None
            item = target.item
            del(target)
            return item
```

3. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (delete_front())

```
class CList:
    def delete_front(self):
        ...
        else:
            first = target.next
            last = target.prev
            first.prev = last
            last.next = first
            self.head = first
```

3. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (delete_front())

```
class CList:
    def delete_front(self):
        ...
        else:
            ...
            item = target.item
            del(target)
            return item
```

3. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (print_list())

```
class CList:
    def print_list(self):
        if self.head == None:
            print('empty')
            return
```

3. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (print_list())

```
class CList:
    def print_list(self):
        ...
        p = self.head
        while p.next != self.head :
            print(p.item, ' <=> ', end= ' ' )
            p = p.next
        print(p.item)
        return
```

3. 원형 양방향 연결 리스트 구현

CList 테스트

```
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()
```

3. 원형 양방향 연결 리스트 구현

Clist 테스트 실행

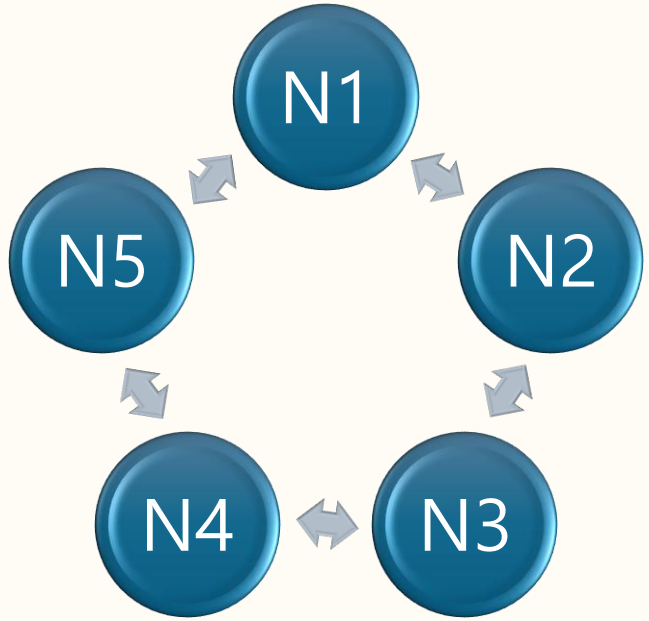
```
mango  <=>  orange  <=>  apple
mango
orange  <=>  apple
orange
apple
apple
empty
None
empty
```

04

기타큐

4. 기타 큐

- 원형 큐
  - 크기가 고정됨
  - front & rear
  - message mailbox 등



4. 기타 큐

- DEQue (Double Ended Queue), 덱
  - 전면과 후면에서 모두 삽입, 삭제 가능
    - FIFO
    - LIFO

4. 기타 큐

❑ 우선순위 큐

- 아이템들이 각기 다른 우선순위를 가질 수 있음.
- dequeue시 가장 우선순위가 높은 아이템을 꺼냄

05

활용분야

5. 활용분야

- CPU의 태스크 스케줄링(Task Scheduling)
  - FCFS
  - Round Robin

5. 활용분야

- 네트워크 프린터 큐
  - FCFS

5. 활용분야

- 콜 센터의 전화 서비스 처리
  - FCFS

5. 활용분야

❑ 그래프의 너비우선탐색 (Breath-First Search) 구현

- 너비우선탐색

1. 처음 방문할 노드를 설정
2. 현재 노드에서 연결된 노드들을 모두 방문하고, 이후 현재 노드에서 연결된 노드 중 하나를 새로운 현재 노드로 설정
3. 모든 노드를 방문할 때까지 2번을 반복.

정리하기

- 큐의 개념
- 큐의 구현
- 원형 양방향 연결 리스트 구현
- 기타 큐 및 큐의 활용분야 이해

11 강

다음 시간 안내 ▶▶▶

큐와 스택2