09강

알고리즘과 자료구조

양방향 연결리스트

서울과학기술대학교 신일훈 교수

학습목표

1. 양방향 연결 리스트 개념 및 연산을 이해한다.
2. 양방향 연결 리스트를 구현할 수 있다.
3. 원형 양방향 연결 리스트 개념을 이해한다.
4. 원형 양방향 연결 리스트를 구현할 수 있다.

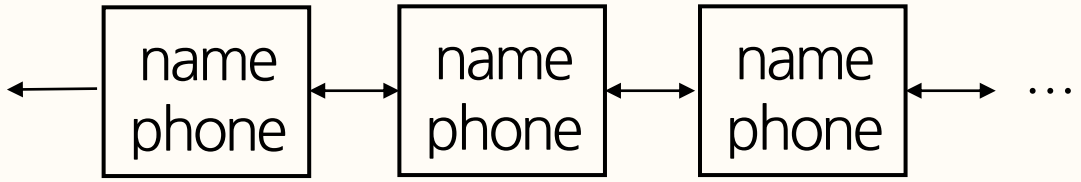
01

양방향 연결 리스트 개념

1. 양방향 연결 리스트 개념

❑ 개념

- 데이터를 저장하는 노드(node)와 노드를 연결하는 링크로 구성됨
- 모든 노드들이 양쪽 방향 링크를 통해 선형으로 연결됨



- 임의의 노드를 찾으면 나머지 노드들도 링크를 통해 접근 가능
- 검색, 삽입, 삭제 연산 등이 가능함

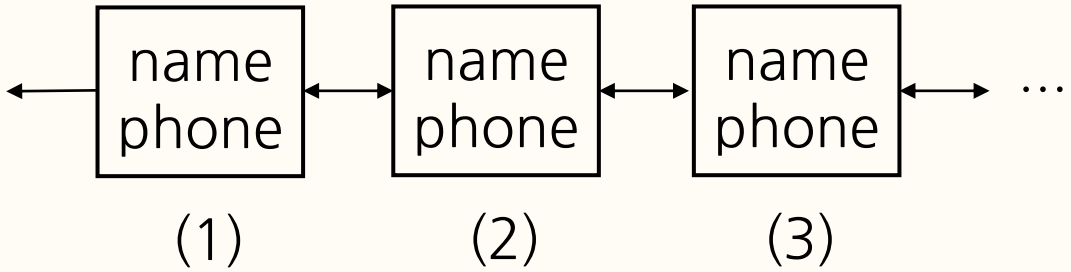
02

양방향 연결 리스트 연산

2. 양방향 연결 리스트 연산

❑ 검색

- 임의의 노드를 접근할 수 있어야 함 (일반적으로 head 노드).
- 타겟 노드를 발견할 때까지, 가장 앞 노드부터 마지막 노드까지 차례로 검사

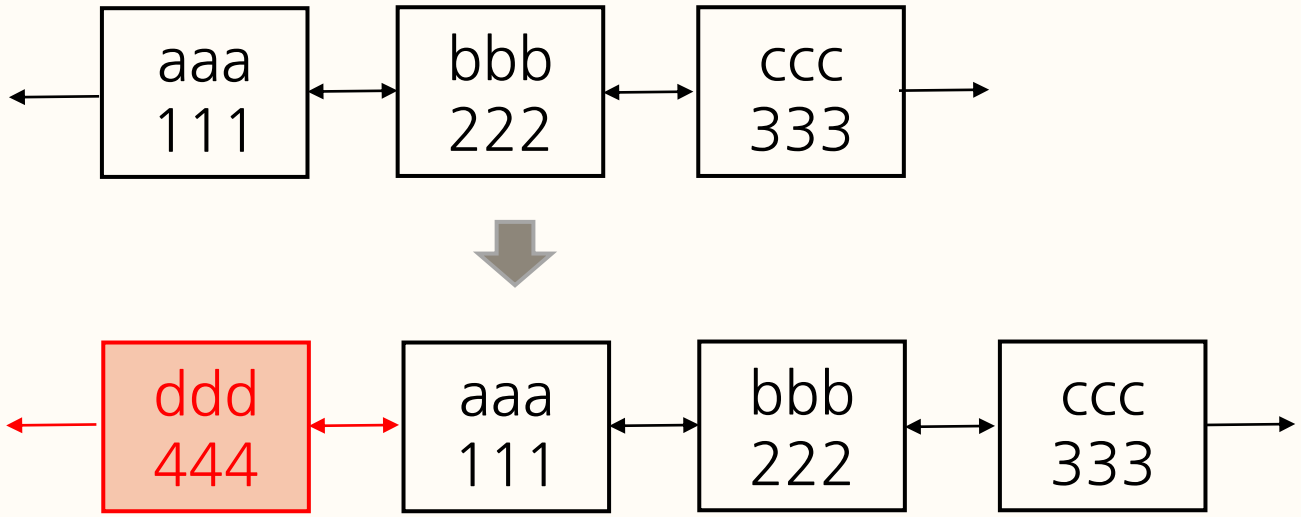


- 최악 시간복잡도: O(N)

2. 양방향 연결 리스트 연산

❑ 전면 삽입

- 가장 앞 노드는 한번에 접근할 수 있다는 전제.

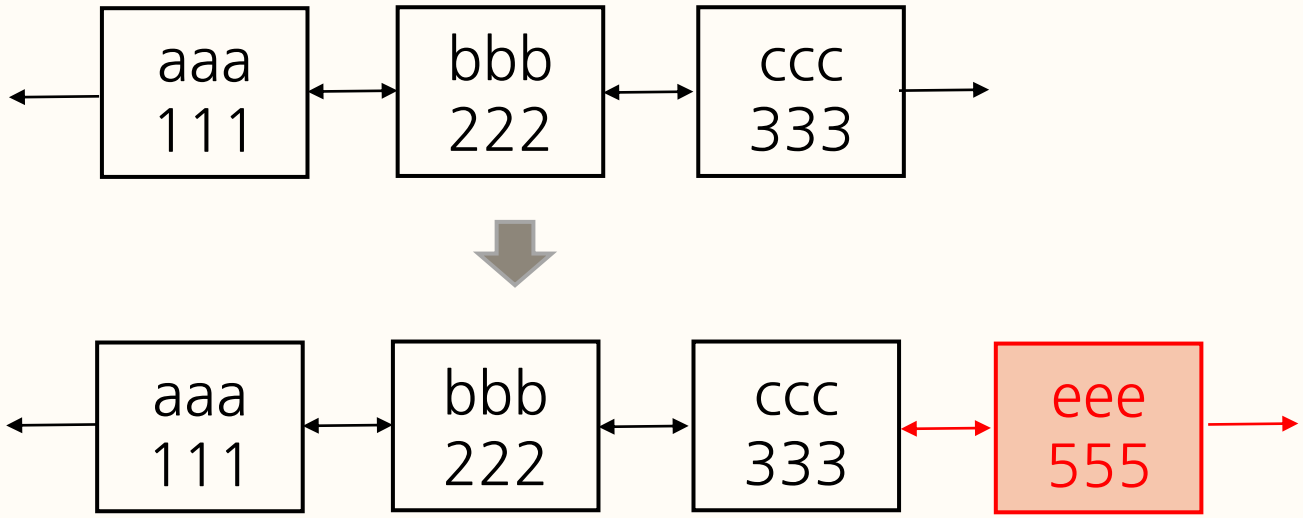


- 최악 시간복잡도: O(1)

2. 양방향 연결 리스트 연산

❑ 후면 삽입

- 가장 앞 노드는 한번에 접근할 수 있다는 전제.



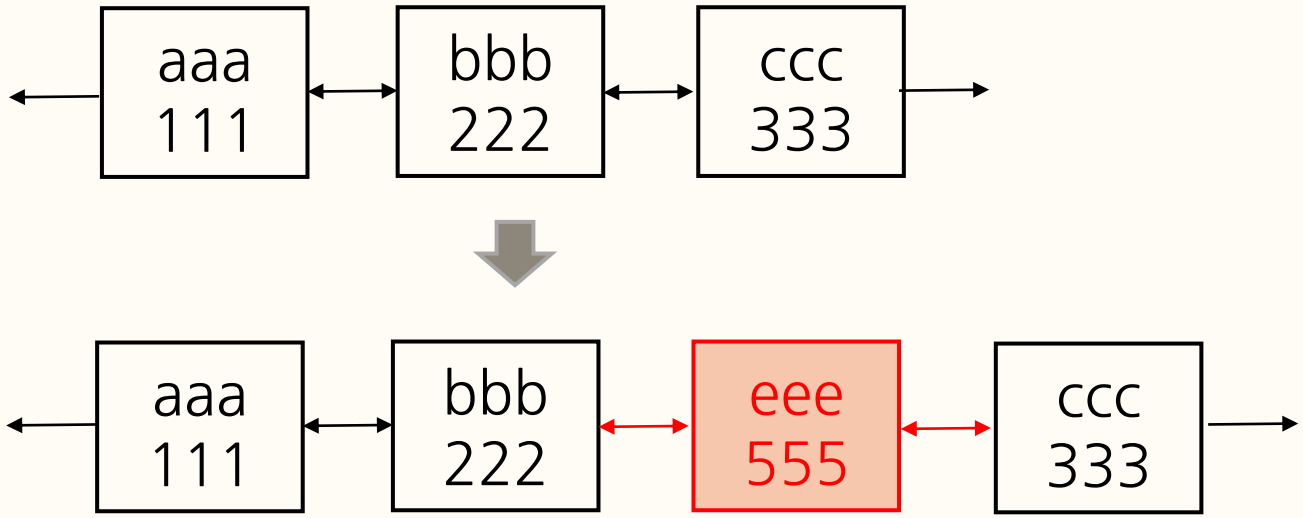
- 최악 시간복잡도: O(N) if the tail node is not directly accessed.

  O(1) if the tail node is directly accessed.

2. 양방향 연결 리스트 연산

❑ 삽입

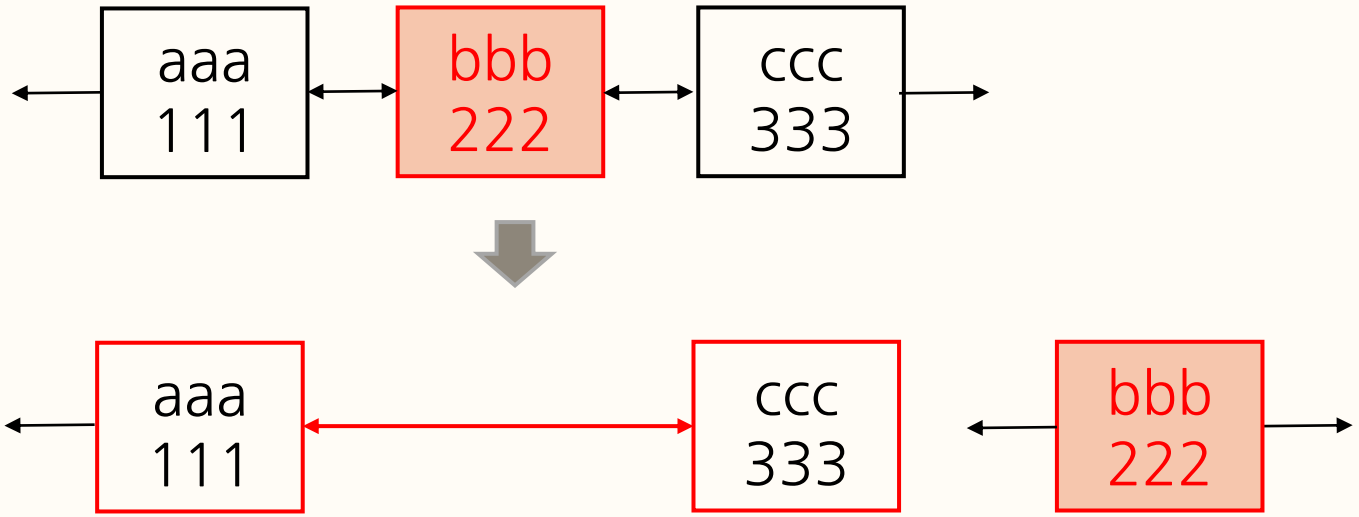
- 가장 앞 노드는 한번에 접근할 수 있다는 전제.



- 최악 시간복잡도: O(1) if the previous or the next node is given

2. 양방향 연결 리스트 연산

❑ 삭제



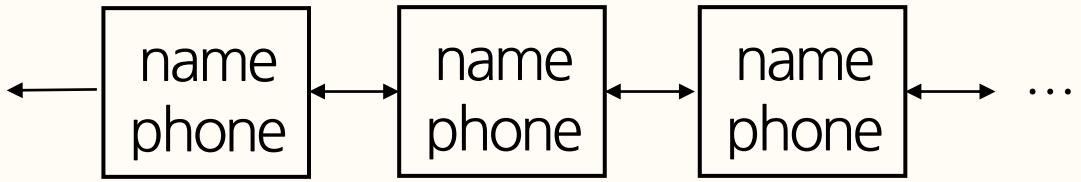
- 타겟 노드의 바로 앞(previous) 노드의 next 링크,
  타겟 노드의 바로 뒤(next) 노드의 prev 링크를 수정해야 함.
- 최악 시간복잡도: O(N)

03

양방향 연결 리스트 구현

3. 양방향 연결 리스트 구현

- 클래스 DNode 정의
  - 리스트를 구성하는 하나의 노드에 해당



  - three (or more) 멤버 변수
    - item (노드의 데이터를 저장)
    - next (다음 노드를 가리키는 링크)
    - prev (이전 노드를 가리키는 링크)
  - 하나의 생성자

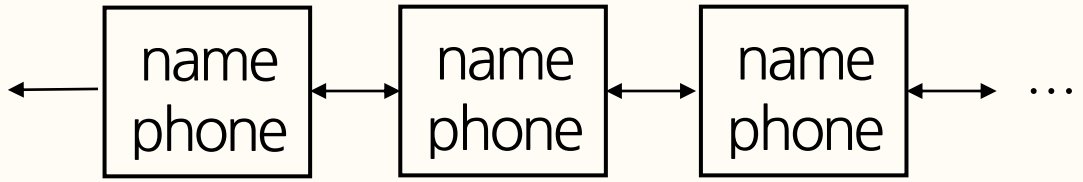
3. 양방향 연결 리스트 구현

클래스 DNode 정의

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

3. 양방향 연결 리스트 구현

- 클래스 DList 정의
  - 양방향 리스트를 나타남
  - one (or more) 멤버 변수
    - head (리스트의 첫 노드를 가리킴)



3. 양방향 연결 리스트 구현

- 클래스 DList 정의
  - 메서드
    - 생성자
    - insert_front()
    - delete_front()
    - search()
    - print_list()
    - ...

3. 양방향 연결 리스트 구현

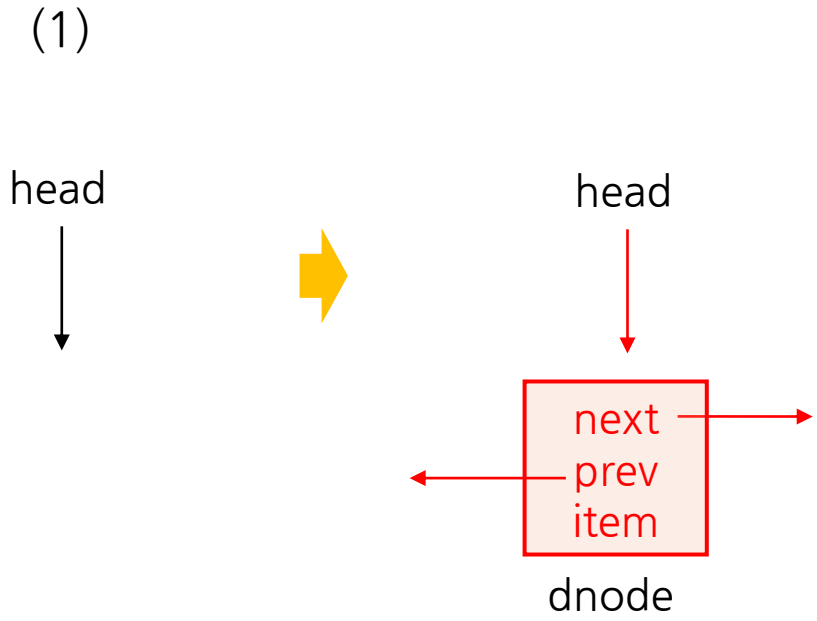
클래스 DList 정의 (생성자)

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

3. 양방향 연결 리스트 구현

클래스 DList 정의 (insert_front())

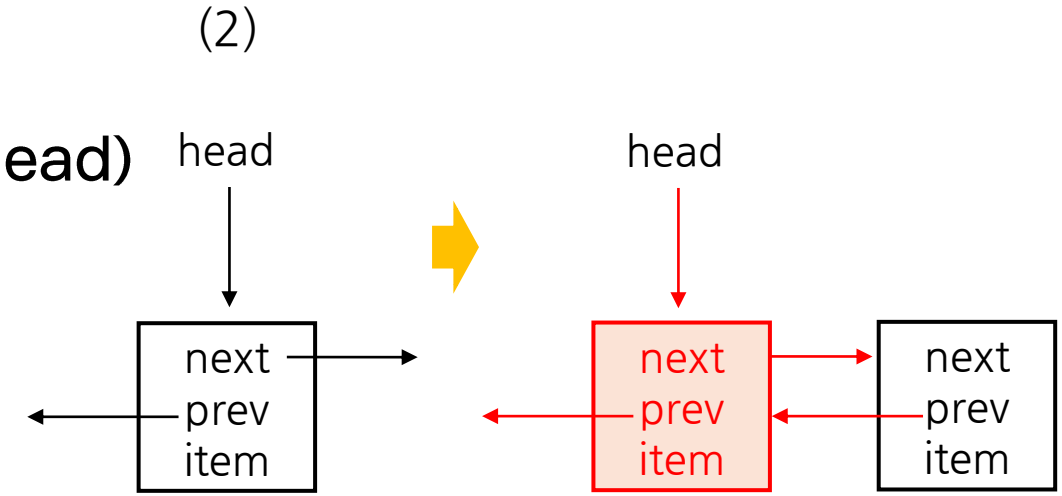
```
class DList :
    def insert_front(self, item):
        dnode = DNode(item, None, self.head)
        if (self.head != None):
            self.head.prev = dnode
        self.head = dnode
```



3. 양방향 연결 리스트 구현

클래스 DList 정의 (insert_front())

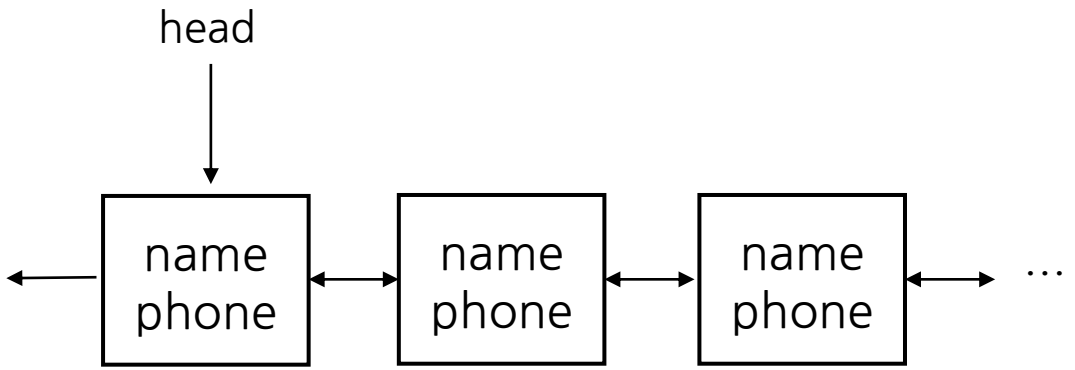
```
class DList :
    def insert_front(self, item):
        dnode = DNode(item, None, self.head)
        if (self.head != None):
            self.head.prev = dnode
        self.head = dnode
```



3. 양방향 연결 리스트 구현

클래스 DList 정의 (print_list())

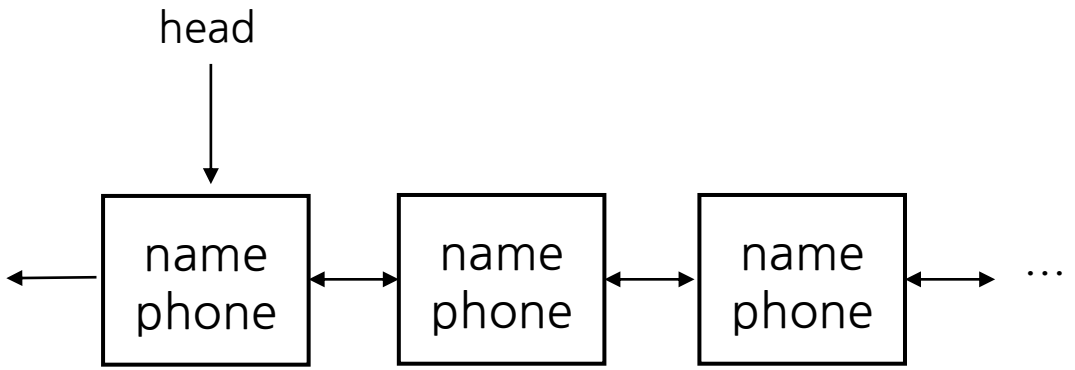
```
class DList:
    def print_list(self):
        if self.head == None:
            print('empty')
            return
```



3. 양방향 연결 리스트 구현

클래스 DList 정의 (print_list())

```
class DList:
    def print_list(self):
        ...
        node = self.head
        while node:
            if (node.next != None):
                print(node.item, ' <=> ', end= ' ')
            else:
                print(node.item)
            node = node.next
```



3. 양방향 연결 리스트 구현

DList 1차 테스트

```
d = DList()
d.insert_front('mango')
d.insert_front('orange')
d.insert_front('apple')
d.print_list()
```

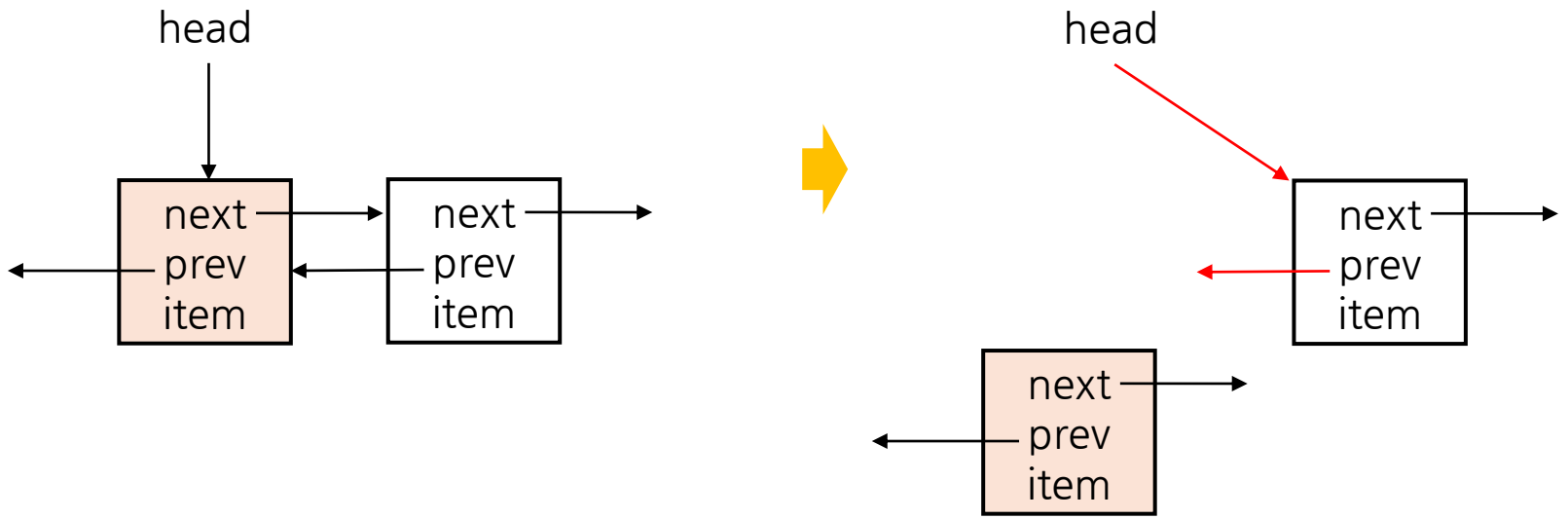
3. 양방향 연결 리스트 구현

DList 1차 테스트 실행

```
In [7]: runfile('D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘/
untitled2.py', wdir='D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘')
apple  <=>  orange  <=>  mango
```

3. 양방향 연결 리스트 구현

클래스 DList 정의 (delete_front())



3. 양방향 연결 리스트 구현

클래스 DList 정의 (delete_front())

```
class DList :
    def delete_front(self):
        target = self.head;     second = None
        if (target != None):
            second = target.next
            self.head = second
            del(target)
        if (second) :
            second.prev = None
```

3. 양방향 연결 리스트 구현

DList 2차 테스트

```
d = DList()
d.insert_front('mango')
d.insert_front('orange')
d.insert_front('apple')
d.print_list()
d.delete_front()
d.print_list()
```

3. 양방향 연결 리스트 구현

DList 2차 테스트 실행

```
In [8]: runfile('D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘/
untitled2.py', wdir='D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘')
apple  <=>  orange  <=>  mango
orange  <=>  mango
```

3. 양방향 연결 리스트 구현

클래스 DList 정의 (search())

```
class DList :
    def search(self, target):
        node = self.head
        while node:
            if target == node.item:
                return True
            node = node.next
        return False
```

3. 양방향 연결 리스트 구현

DList 3차 테스트

```
d = DList()
d.insert_front('mango')
d.insert_front('orange')
d.insert_front('apple')
print(d.search('mango'))
print(d.search('mongo'))
d.delete_front()
print(d.search('mango'))
```

3. 양방향 연결 리스트 구현

DList 3차 테스트 실행

```
In [10]: runfile('D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘/
untitled2.py', wdir='D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘')
True
False
True
```

3. 양방향 연결 리스트 구현

- 그 밖의 메서드 구현
  - 메서드
    - insert_back()
    - delete_back()
    - delete_target()

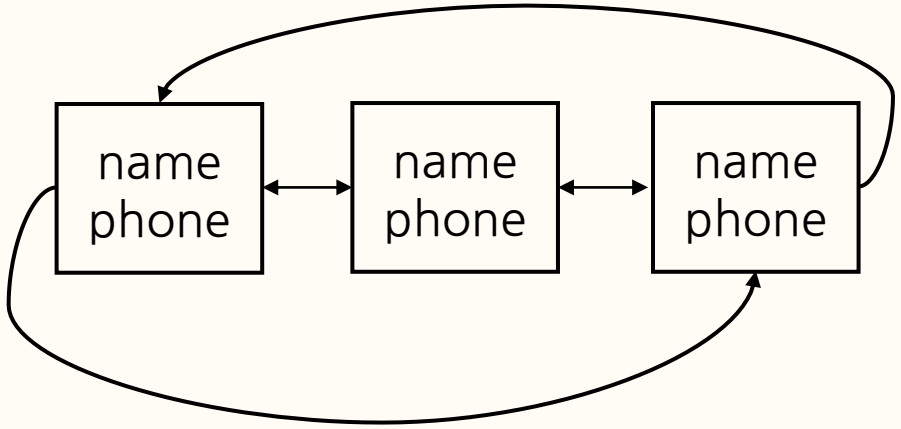
04

원형 양방향 연결 리스트 개념

4. 원형 양방향 연결 리스트 개념

개념

- 데이터를 저장하는 노드(node)와 노드를 연결하는 링크로 구성됨
- 모든 노드들이 양쪽 방향 링크를 통해 선형으로 연결됨



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

05

원형 양방향 연결 리스트 구현

5. 원형 양방향 연결 리스트 구현

- 클래스 CNode 정의
  - 리스트를 구성하는 하나의 노드에 해당
  - three (or more) 멤버 변수
    - item (노드의 데이터를 저장)
    - next (다음 노드를 가리키는 링크)
    - prev (이전 노드를 가리키는 링크)
  - 하나의 생성자

5. 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
```

5. 원형 양방향 연결 리스트 구현

- 클래스 CList 정의
  - 원형 양방향 리스트를 나타남
  - one (or more) 멤버 변수
    - head (리스트의 첫 노드를 가리킴)

5. 원형 양방향 연결 리스트 구현

- 클래스 CList 정의
  - 메서드
    - 생성자
    - insert_front()
    - delete_front()
    - search()
    - print_list()
    - ...

5. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (생성자)

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

5. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (insert_front())

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

5. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (insert_front())

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

5. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (print_list())

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

5. 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
```

5. 원형 양방향 연결 리스트 구현

CList 테스트

```
c = CList()
c.insert_front('mango')
c.insert_front('orange')
c.insert_front('apple')
c.print_list()
```

5. 원형 양방향 연결 리스트 구현

Clist 테스트 실행

```
In [14]: runfile('D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘/
untitled1.py', wdir='D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘')
apple   <=>   orange   <=>   mango
```

5. 원형 양방향 연결 리스트 구현

- 그 밖의 메서드 구현
  - 메서드
    - insert_back()
    - delete_front()
    - delete_back()
    - delete_target()
    - search()

정리하기

- 양방향 연결 리스트 개념 및 연산
- 양방향 연결 리스트 구현
- 원형 양방향 연결 리스트 개념
- 원형 양방향 연결 리스트 구현

10강

다음시간안내 ▶▶▶

큐와 스택 1