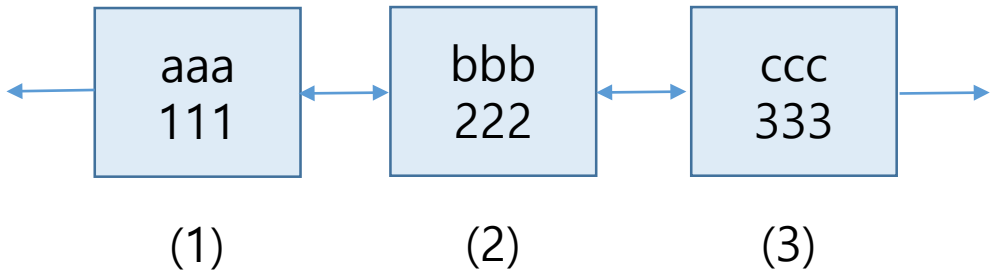
Doubly Linked List
(양방향 연결 리스트)

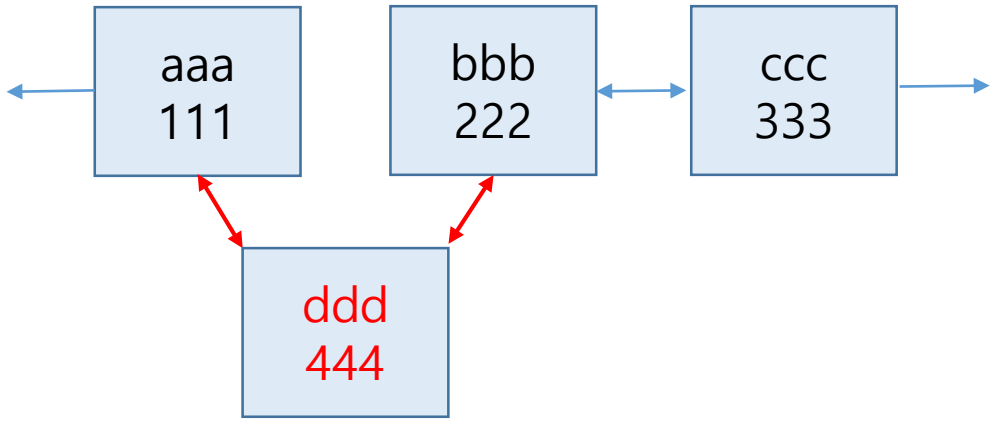
양방향 연결 리스트

- 검색



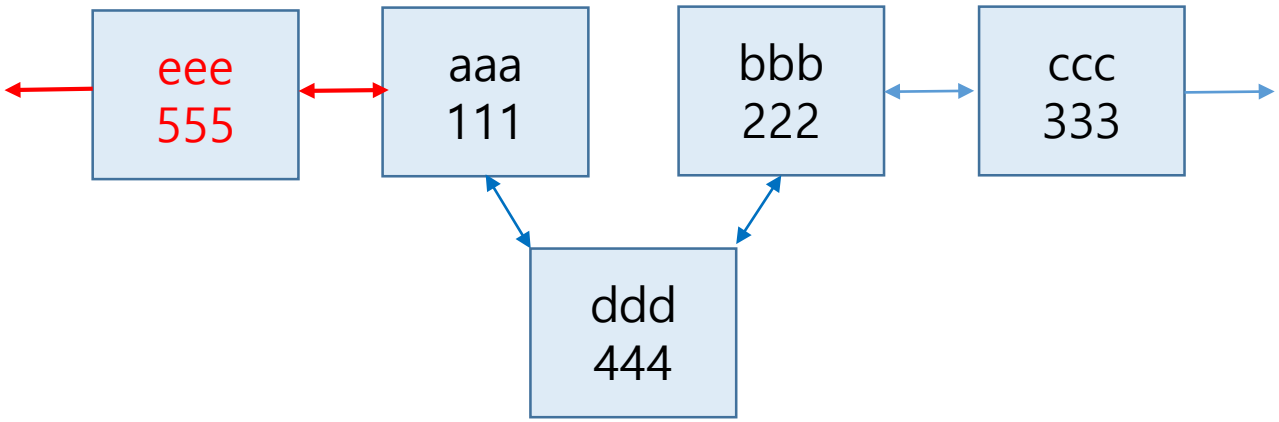
- 시간 복잡도 (time complexity)
  - O(N) => 연결 리스트의 노드 개수에 비례하여 선형 증가

- 삽입



- 시간 복잡도
  - O(1) if the previous or the next node is given

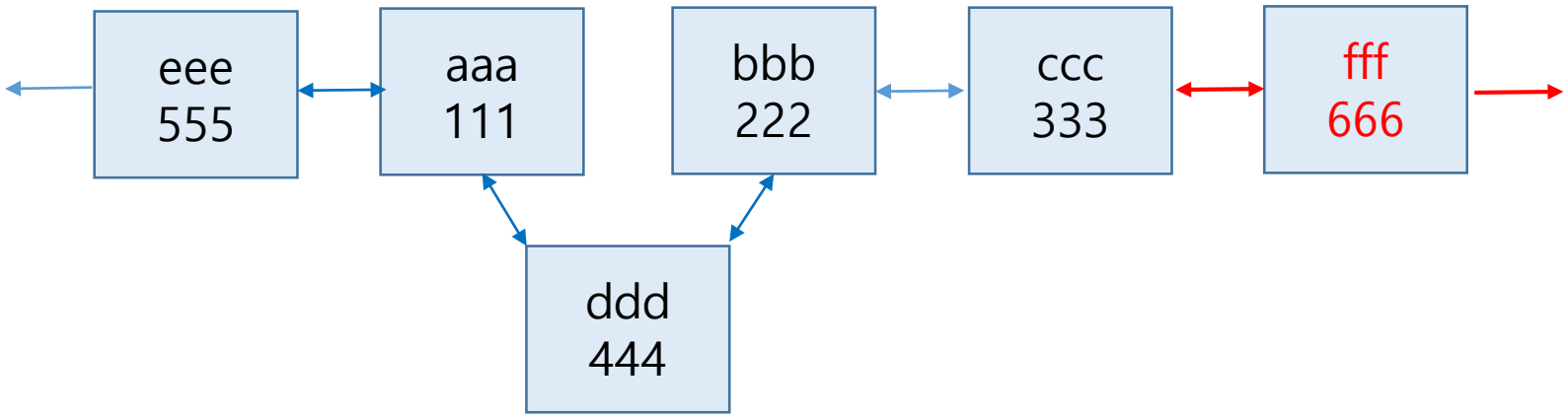
- 전면 삽입



- 시간 복잡도

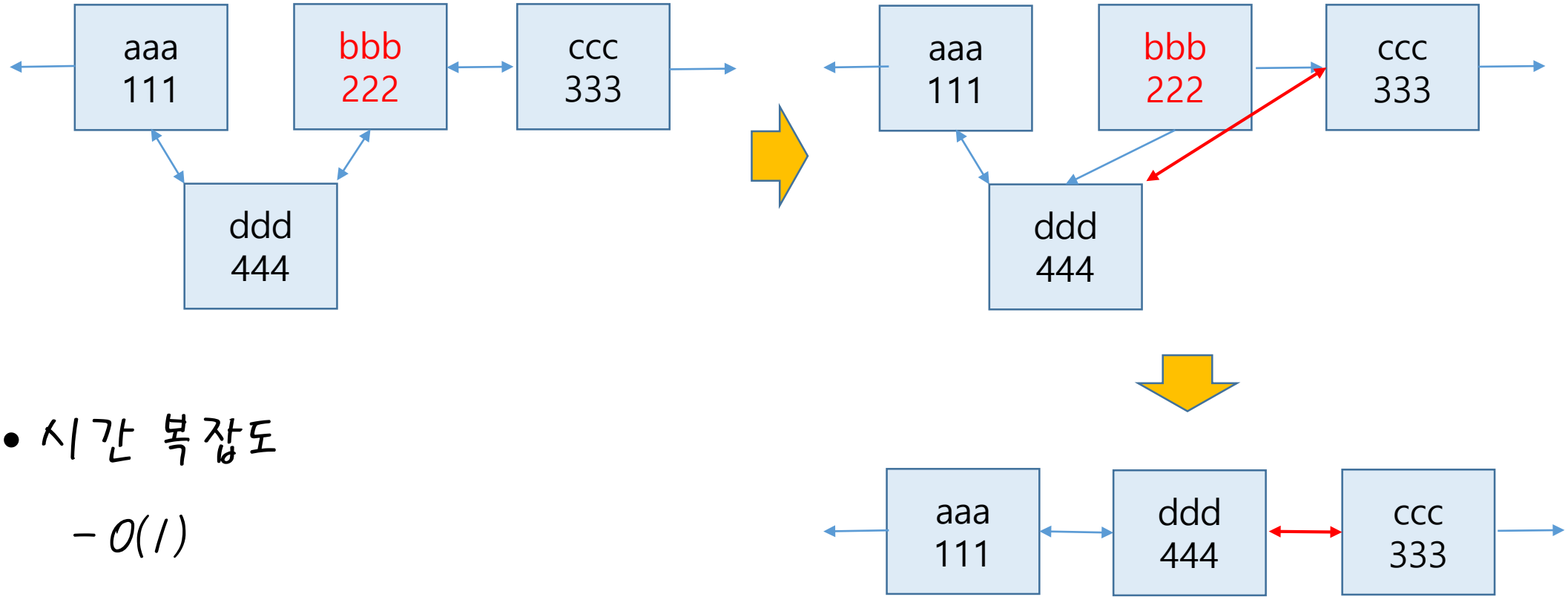
  - O(1)

- 후면 삽입



- 시간 복잡도
  - O(1) if the previous node is given

- 삭제


aaa
111
bbb
222
ccc
333
ddd
444
aaa
111
bbb
222
ccc
333
ddd
444
aaa
111
ddd
444
ccc
333

- 시간 복잡도

  - O(1)

양방향 연결 리스트 구현 – 1단계

- 클래스 DNode 정의
  - 리스트를 구성하는 하나의 노드에 해당
  - three member variables
    - item (노드의 데이터를 가리킴), prev (이전 노드를 가리킴), next (다음 노드를 가리킴)
  - 하나의 생성자 & no other methods


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

양방향 연결 리스트 구현 – 2단계

- 클래스 DList 정의
  - one member variables
    - head (리스트의 첫 노드를 가리킴)
  - 생성자, insert_front(), delete_front(), search(), print_list()


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

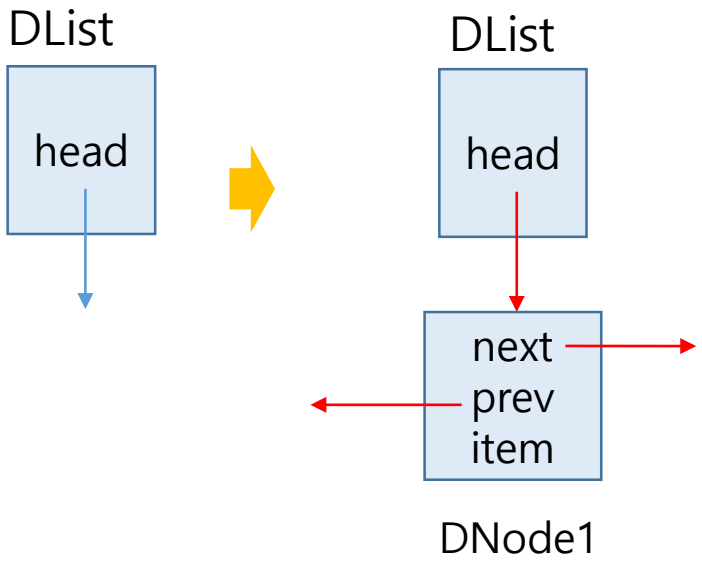
양방향 연결 리스트 구현 – 2단계

- 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
```

(1)



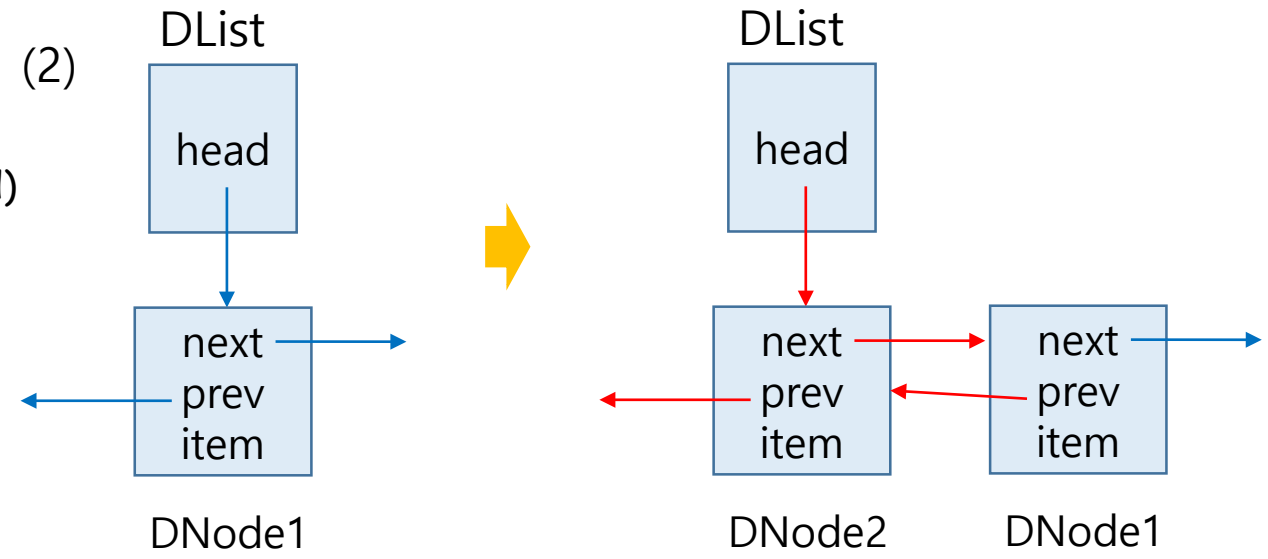
양방향 연결 리스트 구현 – 2단계

- 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
```

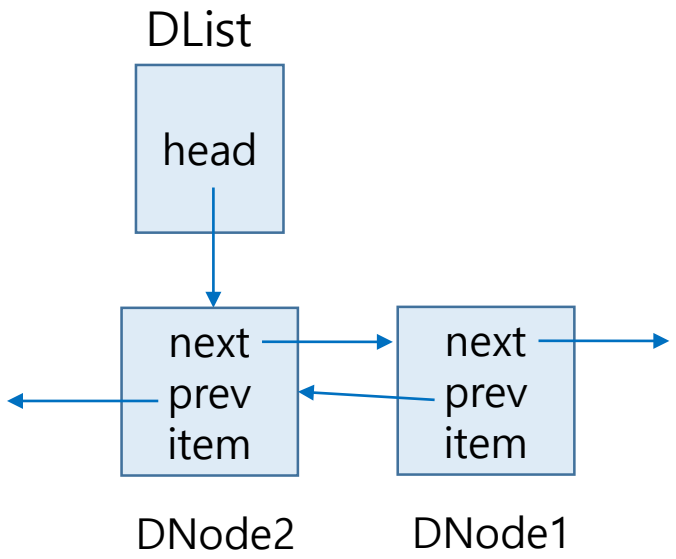
(2)



양방향 연결 리스트 구현 – 2단계

- 클래스 DList 정의
  - print_list()

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

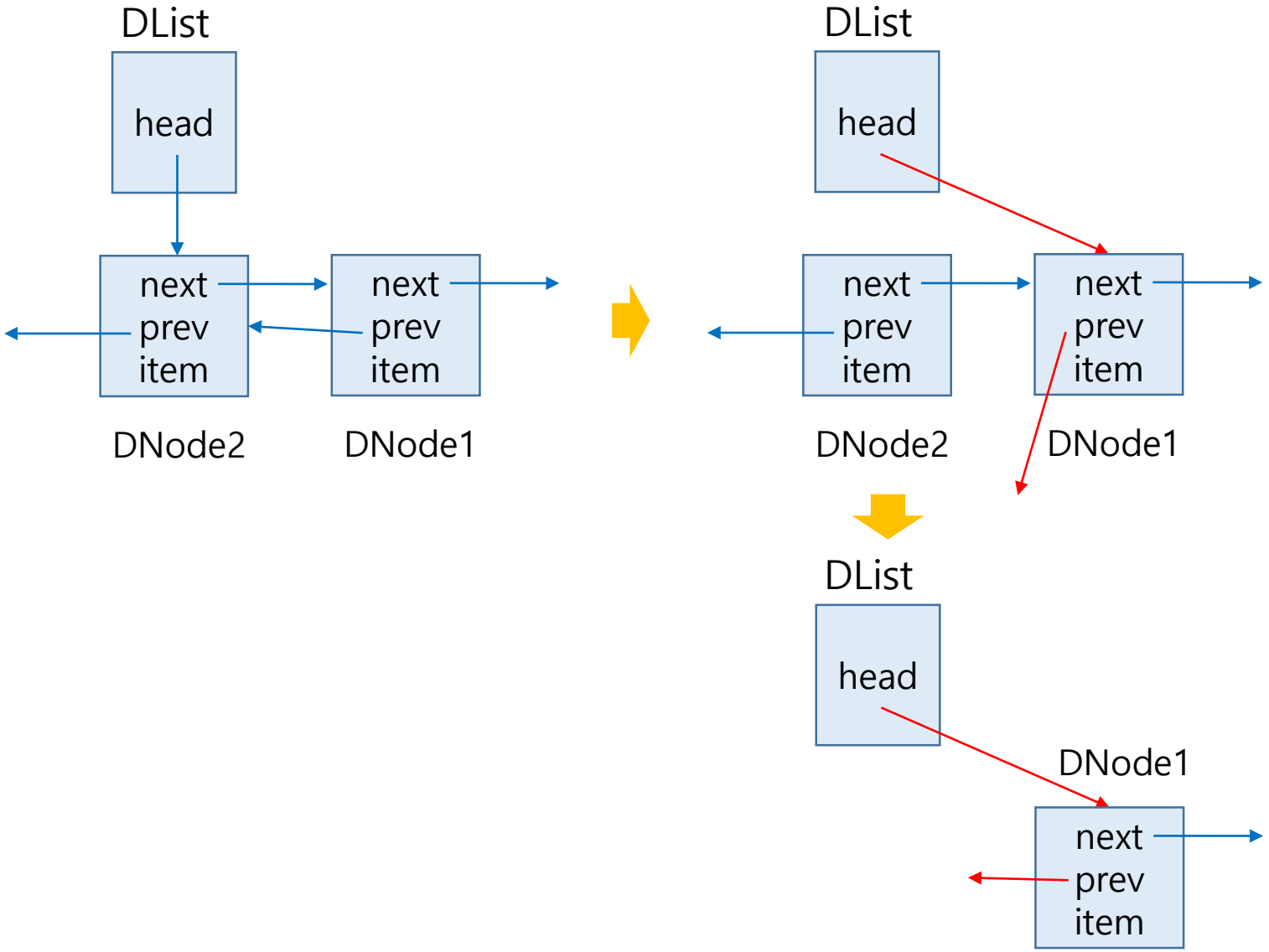


양방향 연결 리스트 테스트 프로그램/

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

양방향 연결 리스트 구현 – 3단계

- 클래스 DList 정의
  - delete_front()



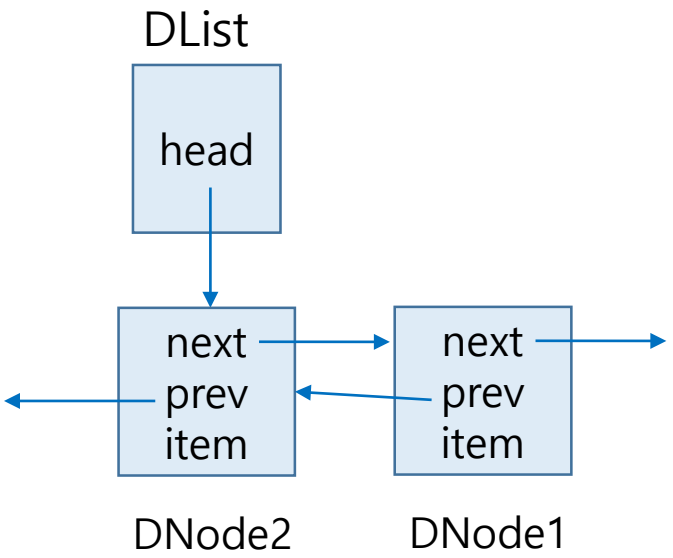
양방향 연결 리스트 활용 프로그램2

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

d.delete_front()
d.print_list()
```

양방향 연결 리스트 구현 – 4단계

- 클래스 DList 정의
  - search()



양방향 연결 리스트 활용 프로그램3

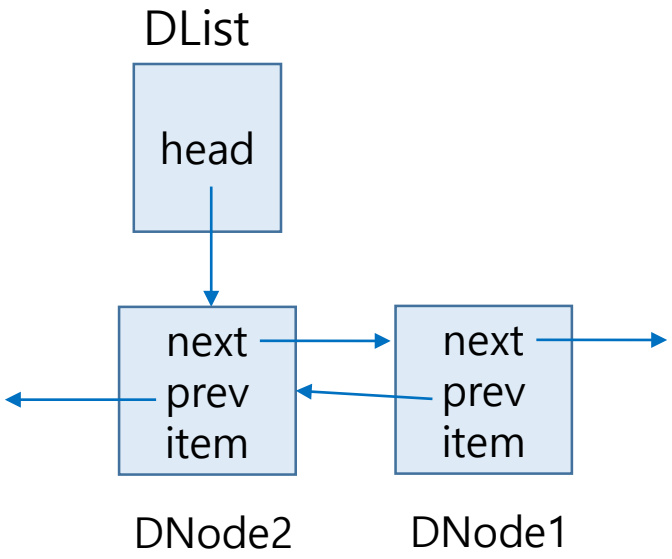
```
d = DList()
d.insert_front('mango')
d.insert_front('orange')
d.insert_front('apple')
d.print_list()

d.delete_front()
d.print_list()

dnode = d.search('mango')
print(dnode.item)
dnode = d.search('apple')
print(dnode)
```

양방향 연결 리스트 구현 – 5단계

```
def delete_target(self, item):
```



양방향 연결 리스트 활용 프로그램4

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

dnode = d.delete_target('mango')
print(dnode.item)
d.print_list()
```

Homework Candidate

- insert_back() 함수 구현하기
  - 리스트 뒤에 추가
- delete_back() 함수 구현하기
  - 리스트 마지막 노드 삭제

Homework Candidate

- DList의 메서드 수정

  - 원 코드

```
d = DList()
dn1 = DNode('mango')
d.insert_front(dn1)
```

  - 수정 코드 (Dnode 생성이 메서드 내에서 이루어지도록)

```
d = DList()
d.insert_front('mango')
```