08강

알고리즘과 자료구조

자료구조 개요 및 연결리스트

서울과학기술대학교 신일훈 교수

한국방송통신대학교 프리임칼리지

학습목표

1. 자료구조의 개념과 종류를 이해한다.
2. 자료구조와 프로그램 성능의 관계를 이해한다.
3. 단방향 연결 리스트 개념 및 연산을 이해한다.
4. 단방향 연결 리스트를 구현할 수 있다.

01

자료구조 개념

1. 자료구조 개념

- 자료구조 (data structure)
  - 자료를 정리하고 조직화하는 다양한 구조
    - 검색, 추가, 삭제의 용이성 및 속도

1. 자료구조 개념

- 자료구조 (data structure)
  - 예> 도서관에서 도서들을 어떻게 정리, 보관할 것인가?
    - 주제별 정리, 저자별 정리, 연도별 정리, 제목 순 정리 등등
  - 예> 휴대폰은 내부적으로 연락처 정보를 어떻게 정리, 저장할까?
  - 예> 윈도우는 내부적으로 현재 실행중인 프로세스 목록을 어떻게 관리할까?
    - 작업 관리자 실행 시 프로세스 정보 출력
    - 다음 CPU 스케줄링의 대상 결정?

02

자료구조 종류

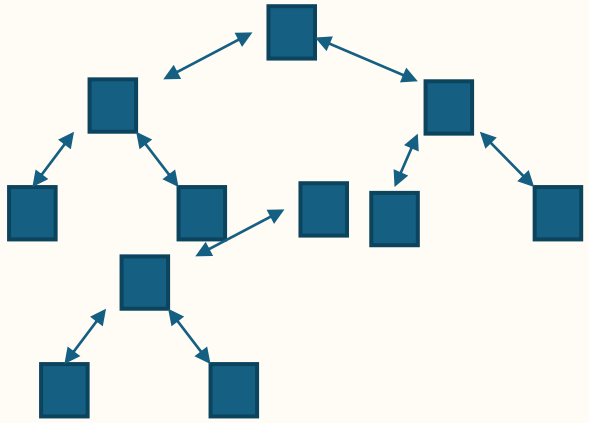
2. 자료구조 종류

선형 (linear) 자료구조

- 항목 접근 방법에 따라 다시 세분화
  - 배열, 연결리스트, 스택, 큐, …

2. 자료구조 종류

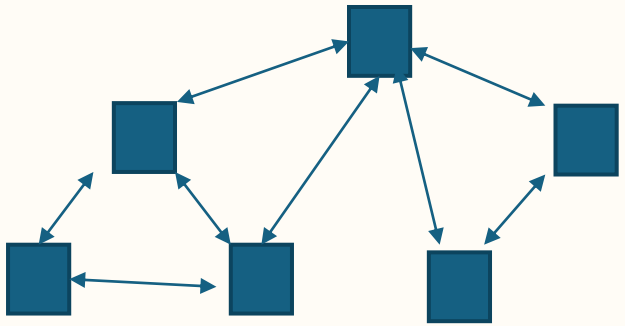
- 비선형 (non-linear) 자료구조
  - 트리 (tree) : 컴퓨터의 폴더 구조와 유사한 계층 구조



  - binary tree, AVL tree, RB tree, B tree, heap, …

2. 자료구조 종류

- 비선형 (non-linear) 자료구조
  - 그래프 (Graph)
    - 정점(노드)과 간선(에지)으로 구성됨
    - 무방향 그래프
    - 방향 그래프
      - 지하철 노선도
      - SNS follow 관계도
      - …



03

자료구조와 프로그램 성능

3. 자료구조와 프로그램 성능

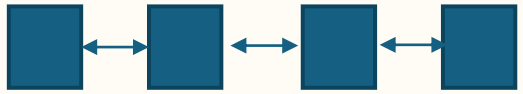
연락처 프로그램

- 개별 엔트리: 이름, 핸드폰, 이메일
- 총 백만 엔트리로 구성
- 어떻게 엔트리들을 조직화할 것인가?

3. 자료구조와 프로그램 성능

❑ 연락처 프로그램

- 1번 자료구조 후보
  - 모든 엔트리를 순서 없이 선형으로 연결



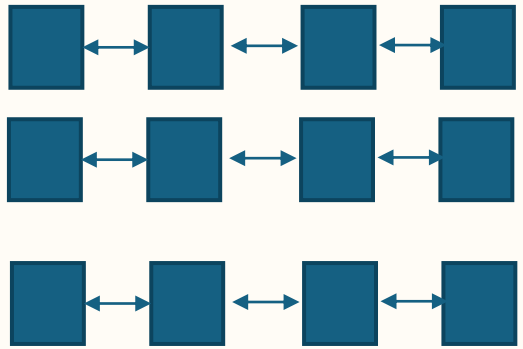
  - 특정 연락처 검색?
    - 첫번째 엔트리부터 마지막 엔트리까지 하나씩 검사해야 함.
    - time consuming.

      최악의 경우, 백만 개 모두 검사해야 함 (O(N)).

3. 자료구조와 프로그램 성능

❑ 연락처 프로그램

- 2번 자료구조 후보
  - 모든 엔트리를 알파벳 순서로 다수의 리스트로 연결

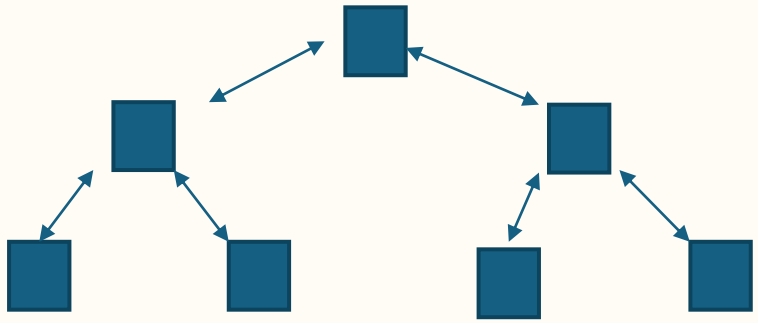


  - 특정 연락처 검색?
    - 검색 시간이 크게 감소함. (평균적으로 1/26)
    - 그러나 최악의 시간복잡도는 여전히 (O(N))

3. 자료구조와 프로그램 성능

❑ 연락처 프로그램

- 3번 자료구조 후보
  - 모든 엔트리를 알파벳 순서로 이진트리(binary tree) 구조로 연결



  - 특정 연락처 검색?
    - 검색 시간이 크게 감소함. 최악의 시간복잡도: $O(\log_2 N)$
    - 구현 난이도 ⬆

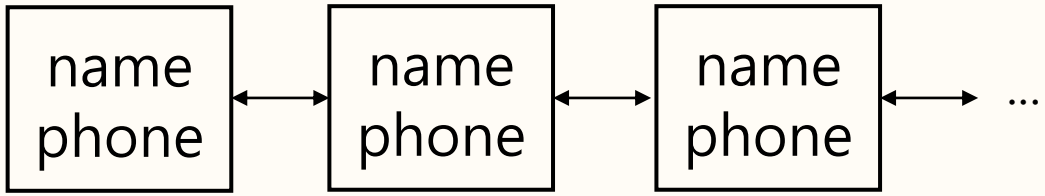
04

연결 리스트

4. 연결 리스트 (Linked List)

개념

- 데이터를 저장하는 노드(node)와 노드를 연결하는 링크로 구성됨
- 모든 노드들이 선형으로 연결됨
- 일반적으로 임의의 위치에서 삭제와 삽입이 가능함



4. 연결 리스트 (Linked List)

종류

- 단방향 연결 리스트 (singly linked list)
- 양방향 연결 리스트 (doubly linked list)

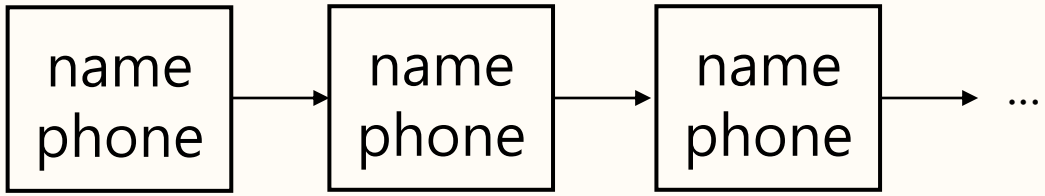
05

단방향 연결 리스트 개념

5. 단방향 연결 리스트 개념

❑ 개념

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



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

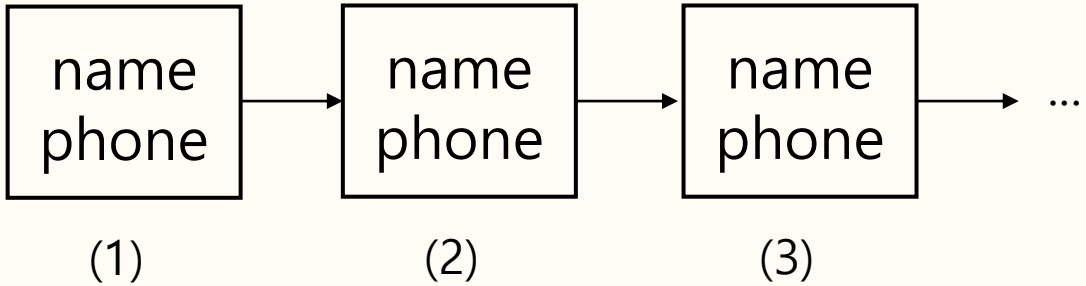
06

단방향 연결 리스트 연산

6. 단방향 연결 리스트 연산

❑ 검색

- 가장 앞 노드를 접근할 수 있어야 함 (위치를 알아야 함).
- 타겟 노드를 발견할 때까지, 가장 앞 노드부터 마지막 노드까지 차례로 검사

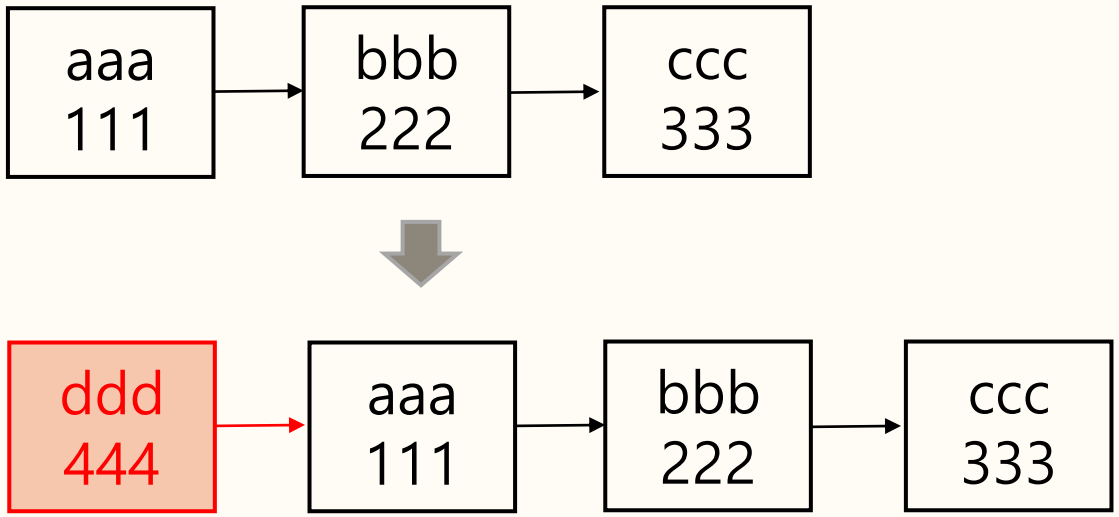


- 최악 시간복잡도: O(N)

6. 단방향 연결 리스트 연산

❑ 전면 삽입

- 가장 앞 노드는 한번에 접근할 수 있다는 전제

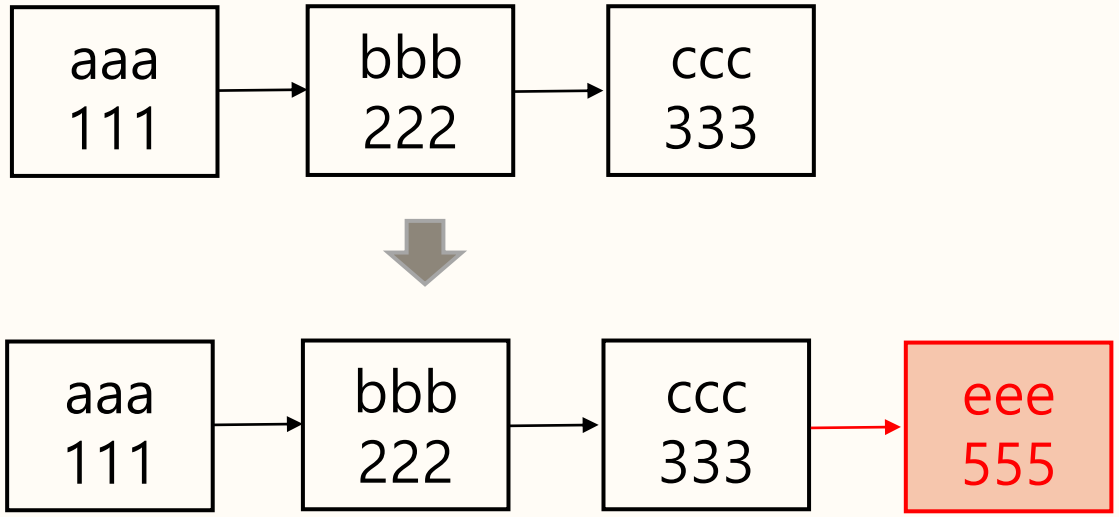


- 최악 시간복잡도: O(1)

6. 단방향 연결 리스트 연산

❑ 후면 삽입

- 가장 앞 노드는 한번에 접근할 수 있다는 전제.

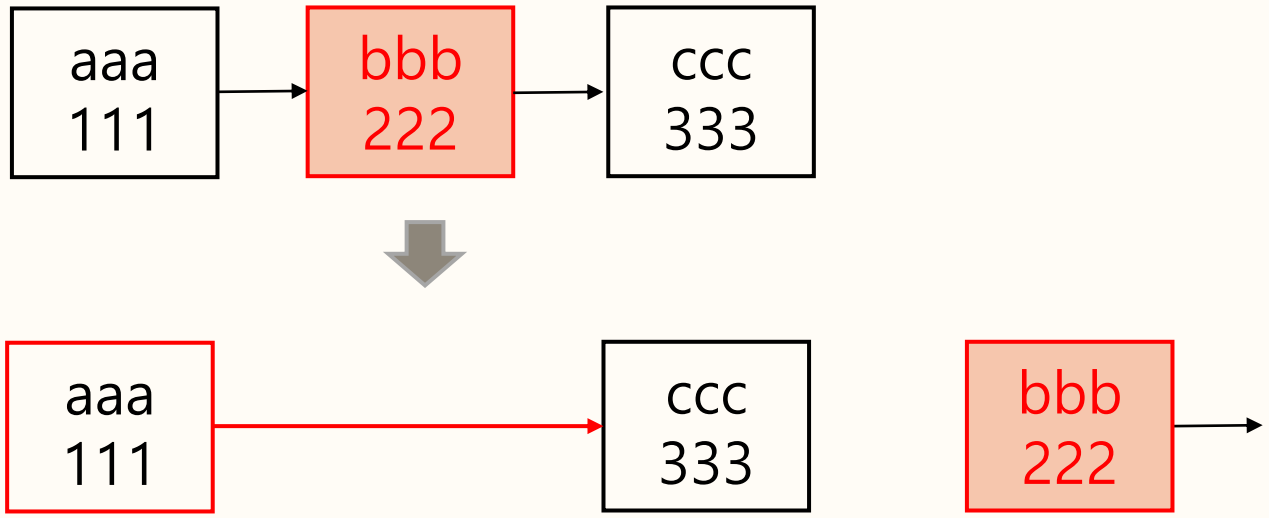


- 최악 시간복잡도: O(N) if the tail node is not directly accessed.

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

6. 단방향 연결 리스트 연산

❑ 삭제



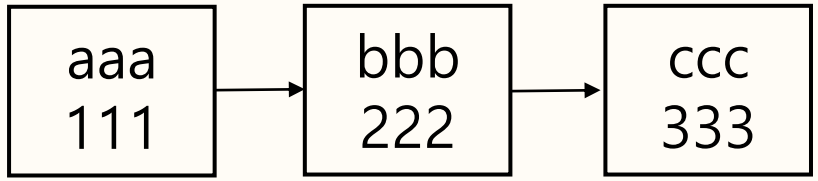
- 타겟 노드의 바로 앞(previous) 노드의 링크를 수정해야 함.
- 최악 시간복잡도: O(N)

07

단방향 연결 리스트 구현

7. 단방향 연결 리스트 구현

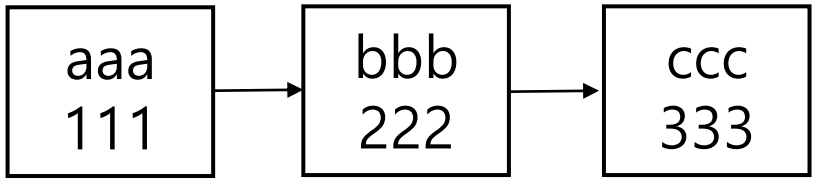
- 클래스 SNode 정의
  - 리스트를 구성하는 하나의 노드에 해당
  - two (or more) 멤버 변수
    - item (노드의 데이터를 저장)
    - next (다음 노드를 가리키는 링크)
  - 하나의 생성자



7. 단방향 연결 리스트 구현

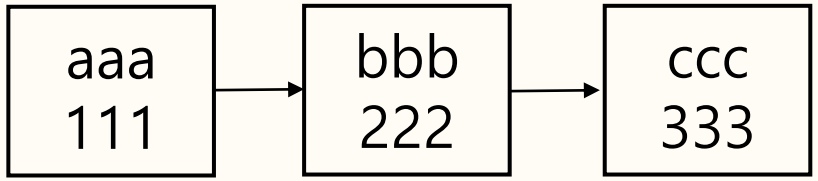
클래스 SNode 정의

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



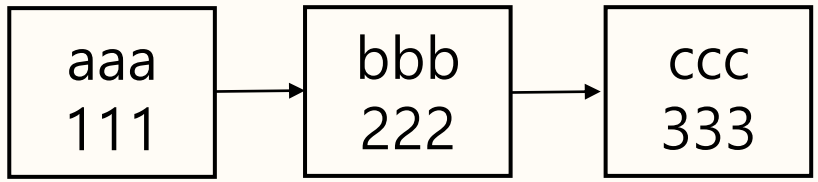
7. 단방향 연결 리스트 구현

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



7. 단방향 연결 리스트 구현

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



7. 단방향 연결 리스트 구현

클래스 SList 정의 (생성자)

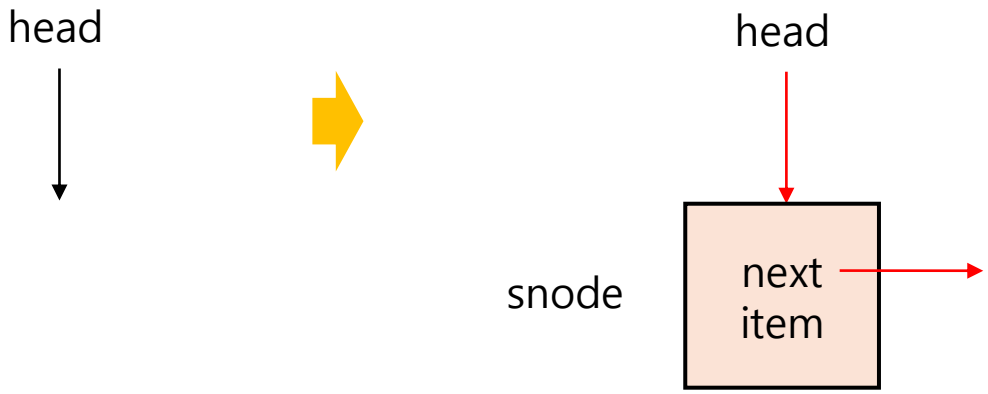
```
class SList:
    def __init__(self):
        self.head = None
```

7. 단방향 연결 리스트 구현

클래스 SList 정의 (insert_front())

```
class SList:
    def insert_front(self, item) :
        snode = SNode(item, self.head)
        self.head = snode
        return
```

(1)

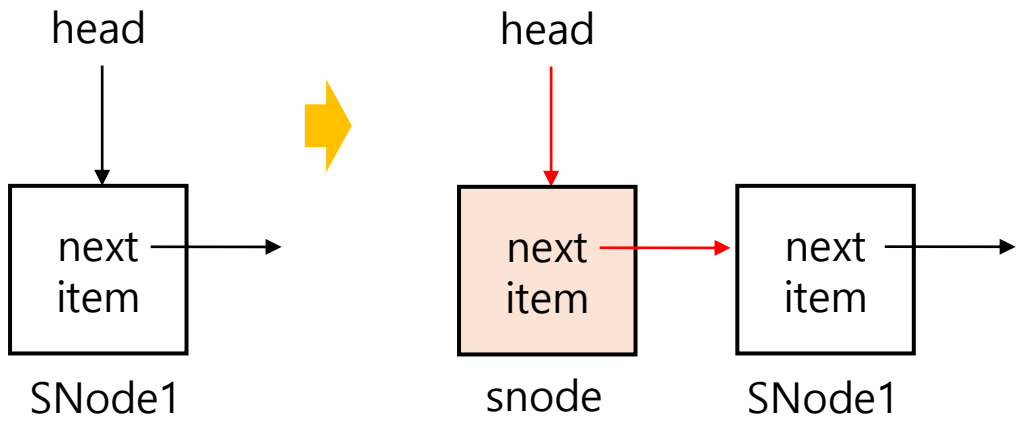


7. 단방향 연결 리스트 구현

클래스 SList 정의 (insert_front())

```
class SList:
    def insert_front(self, item) :
        snode = SNode(item, self.head)
        self.head = snode
        return
```

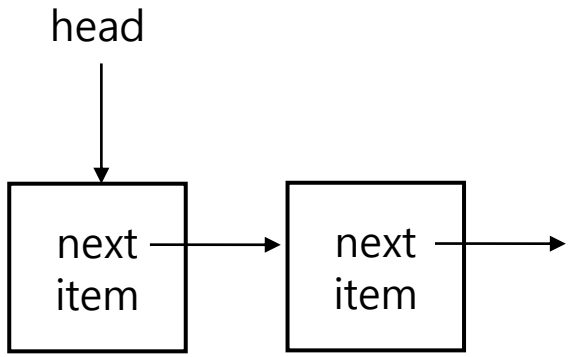
(2)



7. 단방향 연결 리스트 구현

클래스 SList 정의 (print_list())

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



7. 단방향 연결 리스트 구현

SList 1차 테스트

```
s = SList()                    # 빈 연결 리스트 생성
s.insert_front("mango")
s.insert_front("tomato")
s.insert_front("orange")
s.insert_front("apple")
s.print_list()
```

7. 단방향 연결 리스트 구현

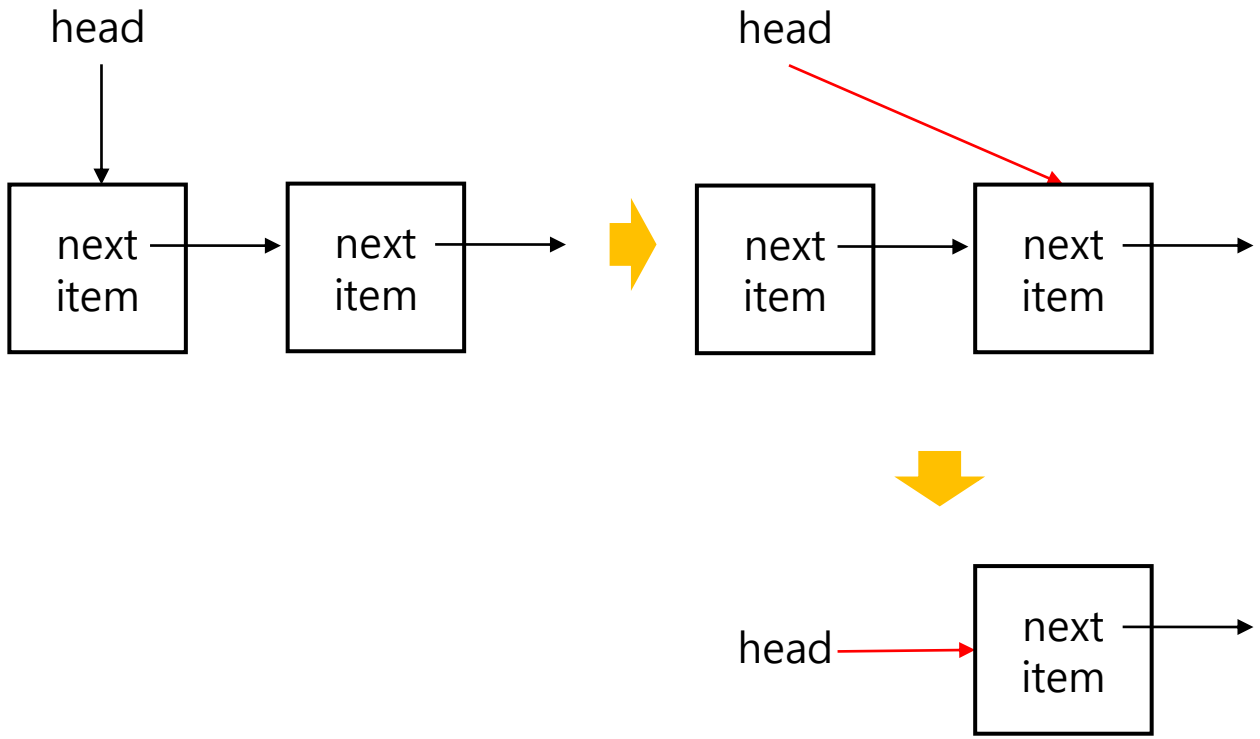
SList 1차 테스트 실행

```
In [4]: runfile('D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘/
untitled1.py', wdir='D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘')
apple  -> orange  -> tomato  -> mango
```

7. 단방향 연결 리스트 구현

클래스 SList 정의 (delete_front())

```
class SList:
    def delete_front (self) :
        first = self.head
        if first != None:
            self.head = first.next
            del(first)
```



7. 단방향 연결 리스트 구현

SList 2차 테스트

```
s = SList()                    # 빈 연결 리스트 생성
s.insert_front("mango")
s.insert_front("tomato")
s.insert_front("orange")
s.insert_front("apple")
s.print_list()                 # 출력
s.delete_front()               # 삭제
s.print_list()
```

7. 단방향 연결 리스트 구현

SList 2차 테스트 실행

```
In [5]: runfile('D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘/
untitled1.py', wdir='D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘')
apple  -> orange  -> tomato  -> mango
orange  -> tomato  -> mango
```

7. 단방향 연결 리스트 구현

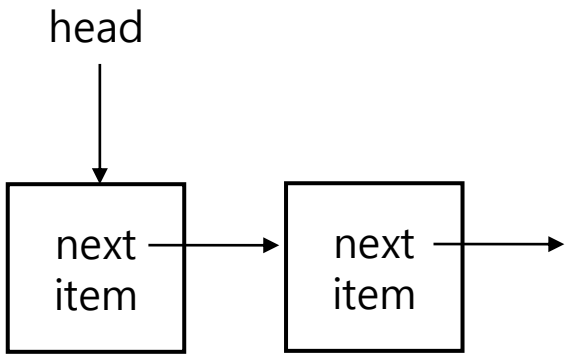
클래스 SList 정의 (search())

```
class SList:
    def search(self, item):
        if (self.head == None):
            return False
```

7. 단방향 연결 리스트 구현

클래스 SList 정의 (search())

```
class SList:
    def search(self, item):
        ...
        cur = self.head
        while cur != None:
            if item == cur.item:
                return True
            cur = cur.next
        return False
```



7. 단방향 연결 리스트 구현

SList 3차 테스트

```
s = SList()                    # 빈 연결 리스트 생성
s.insert_front("mango")
s.insert_front("tomato")
s.insert_front("orange")
s.insert_front("apple")
print(s.search("apple"))
s.delete_front()
print(s.search("apple"))
```

7. 단방향 연결 리스트 구현

SList 3차 테스트 실행

```
In [6]: runfile('D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘/
untitled1.py', wdir='D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘')
True
False
```

7. 단방향 연결 리스트 구현

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

7. 단방향 연결 리스트 구현

❑ 단방향 연결 리스트 구현2 (head, tail)

클래스 SList 정의 (생성자)

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

7. 단방향 연결 리스트 구현

❑ 단방향 연결 리스트 구현2 (head, tail)

클래스 SList 정의 (insert_front())

```
class SList:
    def insert_front(self, item) :
        snode = SNode(item, self.head)
        self.head = snode
        if (self.tail == None) :
            self.tail = snode
        return
```

정리하기

- 자료구조의 개념 및 종류
- 자료구조와 프로그램 성능
- 단방향 연결 리스트 개념 및 연산
- 단방향 연결 리스트 구현

09강

다음 시간 안내 ▶▶▶

양방향 연결리스트