SNUDM 2021학년도 봄학기 서울대학교 〈산업컴퓨팅개론〉 강좌

# Python 실습 V
# 자료구조의 구현과 응용

TA 유용재
yooy@dm.snu.ac.kr

# O V E R V I E W

1. 리스트와 자료구조
2. 그래프와 알고리즘
3. 이진 트리의 구현

# 1 리스트와 자료구조

## 원소들을 차례로 하나하나 쌓는다, Stack

# Stack

- ✓ 책을 새로 쌓든, 있는 책을 빼든… 반드시 최상단에서 작업이 이루어짐
- ✓ **Last In First Out** 마지막에 들어간 원소가 가장 먼저 빠져나오는 자료구조
- ✓ 책의 최상단 = 원소가 삽입되는 위치 = 원소가 제거되는 위치 = top

## 새로운 원소를 추가하는 push, 기존의 원소를 제거하는 pop

```
push(7)
push(2)
pop()
push(3)
push(9)
push(4)
pop()
push(1)
pop()
pop()
pop()
```

- 데이터의 삽입은 push, 데이터의 삭제는 pop
- **데이터 삭제 시, 가장 최근에 삽입되었던 데이터가 삭제됨**

**Q** 비어 있는 Stack에 대하여 좌측 코드를 따를 때, 가장 마지막으로 삭제되는 원소는?

## 리스트를 이용해 Python에서 Stack 구현하기

### Stack for Python

```
stack = []

stack.append(3)
stack.append(1)
stack.append(8)
stack.append(9)

stack.pop()
stack.pop()

print(stack)
```

◀ 비어 있는 리스트를 생성함으로써 빈 Stack에서 출발

◀ append를 이용하여 리스트의 맨 끝에 원소를 삽입

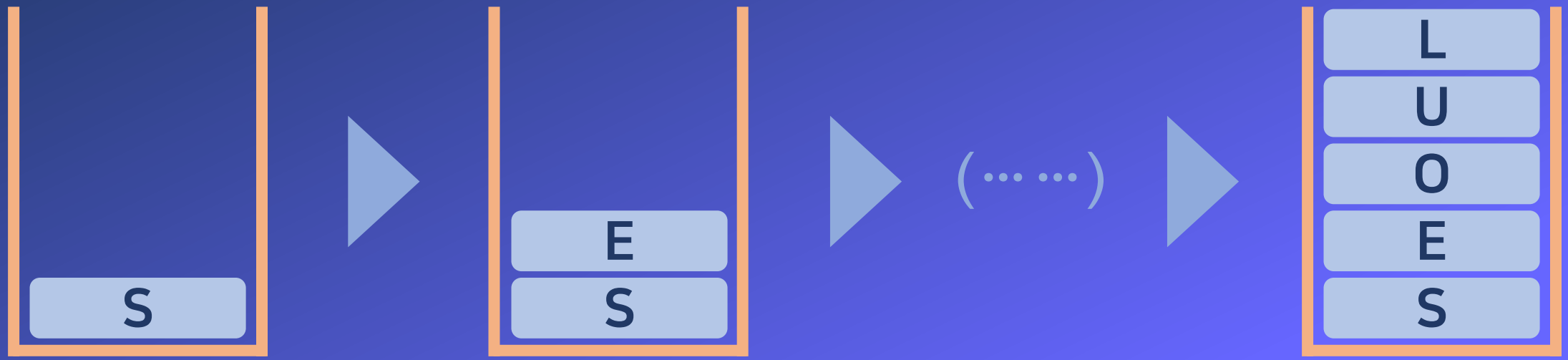
◀ pop을 이용하여 가장 마지막에 삽입된 원소를 제거

## Stack, 어떤 상황에서 어떻게 활용할 수 있을까

알파벳으로 이루어진 단어 한 개를 사용자로부터 입력받는다.

**입력받은 단어를 철자 역순으로 출력하려면?**

(S E O U L → L U O E S)



Stack, 어떤 상황에서 어떻게 활용할 수 있을까

# Queue


- ✓ 먼저 줄을 서고 기다린 손님부터 먼저 대기열을 빠져나감
- ✓ First In First Out 처음에 들어간 원소가 먼저 빠져나오는 자료구조
- ✓ 대기열의 맨 앞 = 원소가 제거되는 위치 = head

  대기열의 맨 뒤 = 원소가 삽입되는 위치 = tail

## Queue에서 원소의 삽입과 삭제

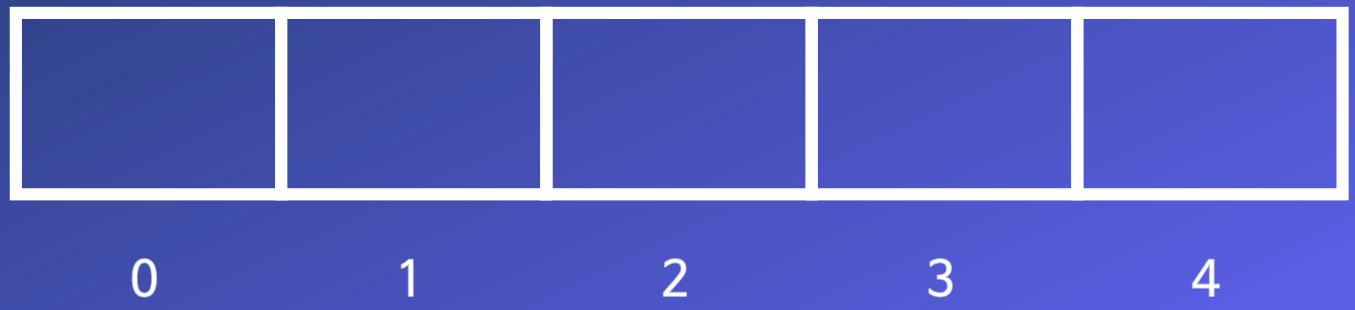
```
enqueue(7)
enqueue(2)
dequeue()
enqueue(3)
enqueue(9)
enqueue(4)
dequeue()
enqueue(1)
dequeue()
dequeue()
dequeue()
```

- 데이터의 삽입은 enqueue, 데이터의 삭제는 dequeue
- **데이터 삭제 시, 가장 먼저 삽입되었던 데이터가 삭제됨**

**Q** 비어 있는 Queue에 대하여 좌측 코드를 따를 때, 가장 마지막으로 삭제되는 원소는?

## 리스트 형태 Queue가 갖는 문제, 대안적 자료구조의 필요성

```
queue = []

queue.append(4)
queue.append(7)
queue.append(2)

queue.pop(0)
```




**Q** 위와 같이 리스트를 이용하여 Queue를 구현하였을 때, dequeue가 갖는 시간복잡도를 계산하면?

## 시간복잡도 문제를 해소한 Python에서의 Queue

### Queue for Python

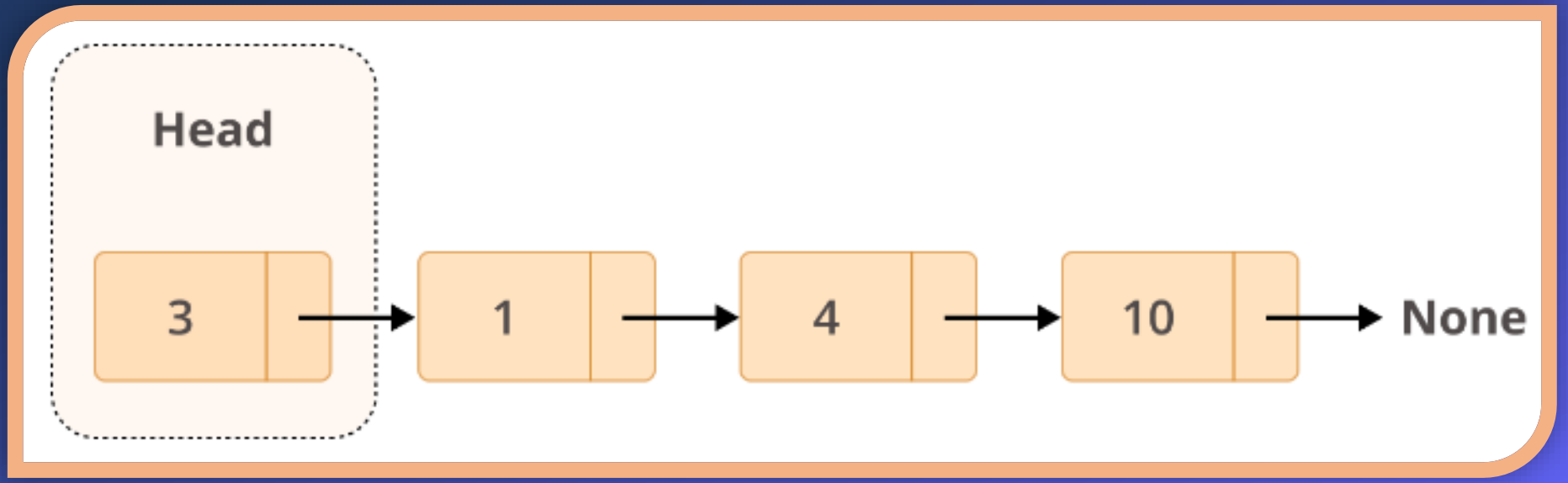
```
from queue import Queue

num_queue = Queue()                    ◀ 비어 있는 새로운 Queue를 생성

num_queue.put(5)                       ◀ 새로운 원소를 삽입하기 위한 put
num_queue.put(4)
num_queue.put(7)

num_queue.get()                        ◀ 가장 먼저 삽입된 원소를 삭제

print(num_queue.queue)                 ◀ Queue에 존재하는 원소들을 출력
```

나 다음에는 너, 너 다음에는 누구? **Linked List**

# Linked List


- ✓ Linked List 각 원소는 **자신의 값**과 **다음 원소 포인터**를 가짐
- ✓ 먼저 헤드 포인터가 Linked List의 첫 번째 원소를 가리킴
- ✓ 첫 번째 원소의 '다음 원소 포인터'는 두 번째 원소를 가리킴
- ✓ 원소들이 꼬리에 꼬리를 무는 형태로 연이어 존재하는 자료구조

## 간단한 다이어그램으로 이해하는 Linked List




- ✓ 새로운 원소를 Head에 삽입하려면?
- ✓ Python에서 Linked List를 구현하려 할 때, 올바른 접근 방법은?

# 클래스를 이용하여 Python에서 Linked List 구현하기

## Node

```
class Node :

  def __init__(self, v, n = None) :
    self.value = v
    self.next = n
```

◀ Node 클래스를 새로 선언

◀ 새로운 Node를 생성하는 경우
◀ Node의 첫째 구성 요소는 값
◀ Node의 둘째 구성 요소는 포인터

# 클래스를 이용하여 Python에서 Linked List 구현하기

## 빈 Linked List를 생성하고 새로운 Node를 삽입하기

```
class LinkedList :                          ◀ LinkedList 클래스를 새로 선언

  def __init__(self) :                      ◀ 새로이 LinkedList를 생성하면 빈 상태
      self.head = None

  def insertNode(self, v) :
    if self.head is None :                  ◀ 빈 상태에서 삽입되는 새 Node는 곧 head
      self.head = Node(v)
    else :
      self.head = Node(v, self.head)        ◀ head 위치에 새로 삽입된다고 가정
```

## 클래스를 이용하여 Python에서 Linked List 구현하기

### head가 가리키는 원소를 제거하기

```
class LinkedList :

  def deleteNode(self) :
    if self.head is None :                    ◀ 빈 LinkedList에서는 Node를 제거할 수 없음
      print("Linked List is empty!")
      return
    else :
      self.head = self.head.next              ◀ 현재의 head가 가리키는 원소를 제거
```

# 클래스를 이용하여 Python에서 Linked List 구현하기

## Linked List에 존재하는 모든 원소를 순서대로 출력하기

```
class LinkedList :

  def printList(self) :
    if self.head is None :                    ◀ LinkedList가 비어 있는 경우
      print('Linked List is empty!')
      return
    else :
      print('<Linked List>')
      link = self.head                        ◀ head 위치에서 출력을 시작
      while link :
        print(link.value, '->', end = ' ')
        link = link.next                      ◀ 다음 원소로 이동하여 출력을 계속
```

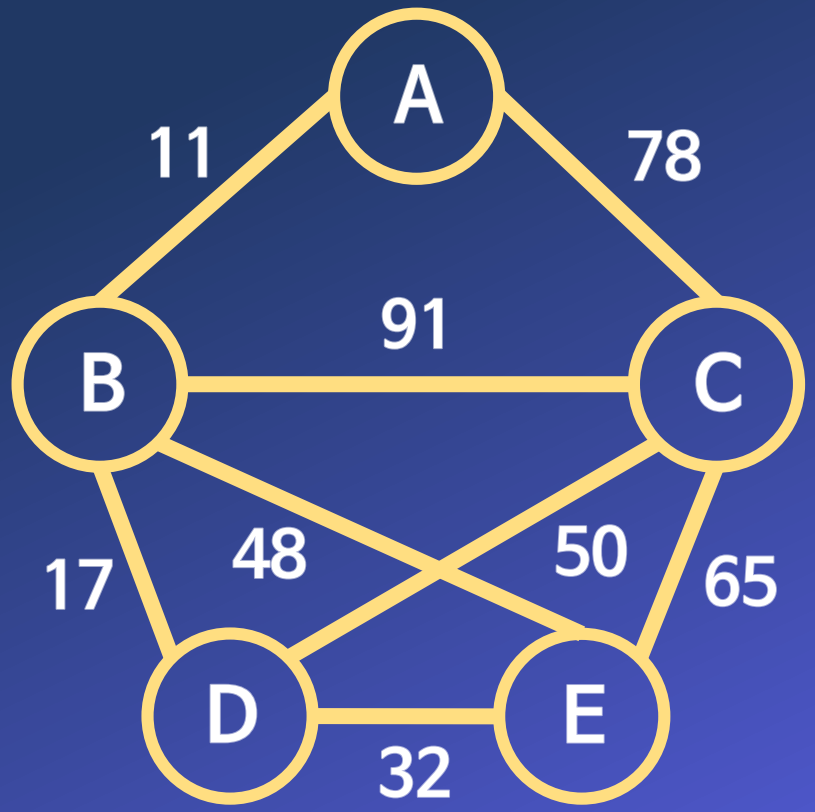
# 2 그래프와 알고리즘

## 여러 개의 점과 점을 잇는 선, 그래프 자료구조

# Graph

- ✓ G = (V, E) 값 여러 개의 꼭짓점과 그들을 잇는 모서리로 정의됨
- ✓ 유향 그래프와 무향 그래프, 가중 그래프와 비가중 그래프

## 간선 가중치가 존재하는 그래프에서의 최소 신장 트리 문제




✓ **Minimum Spanning Tree**

모든 꼭짓점을 포함하면서, 가중치 합이 최소가 되는 트리

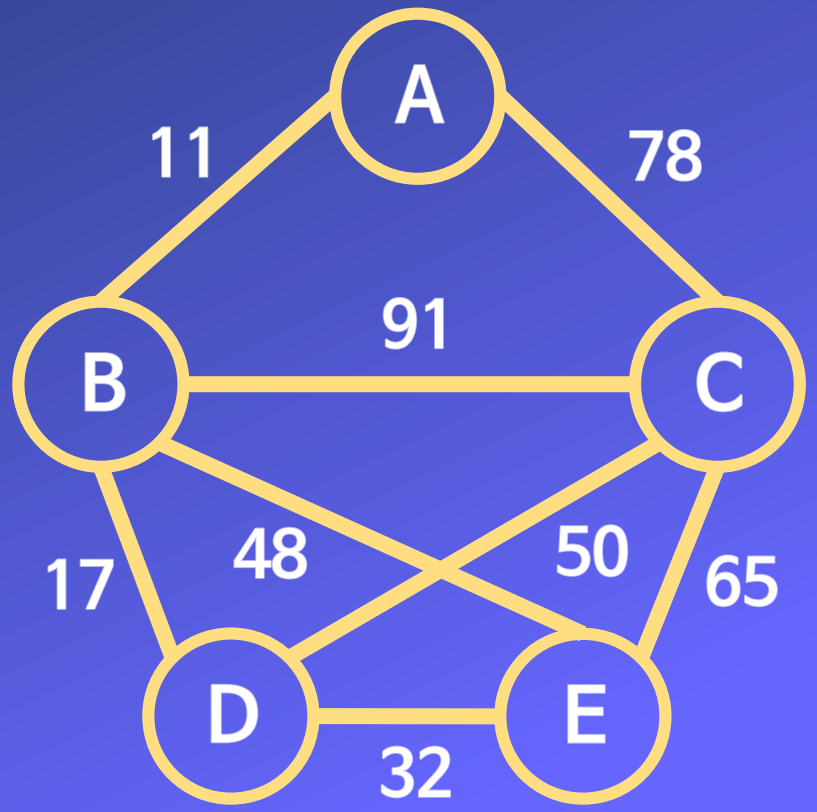
✓ 좌측과 같은 무향 그래프에서, 최소 신장 트리는?

## 가중치 합을 최소로 만드는 트리 탐색, Prim 알고리즘

### Prim's Algorithm

✓ 임의의 꼭짓점을 선택하여, 해당 꼭짓점과 이어진 간선 중 가중치가 최소인 것을 선택하여 두 점을 연결

✓ 앞서 연결된 두 점과 이어진 간선 중 가중치 최소인 것을 선택 세 점을 연결, 이 과정을 반복하여 최소 신장 트리를 도출



## 그래프가 주어졌을 때, Python에서 Prim 알고리즘 적용하기

### 꼭짓점과 간선 가중치를 이용하여 Graph를 정의

```
V = 5
G = [[0, 11, 78, 0, 0],
     [11, 0, 91, 17, 48],
     [78, 91, 0, 50, 65],
     [0, 17, 50, 0, 32],
     [0, 48, 65, 32, 0]]
selected = [0, 0, 0, 0, 0]
edge_count = 0
selected[0] = 1
```

## 그래프가 주어졌을 때, Python에서 Prim 알고리즘 적용하기

### 꼭짓점과 간선 가중치를 이용하여 Graph를 정의

```
while (edge_count < V - 1) :
  min = float('inf')
  x = 0
  y = 0
  for i in range(V) :
    if (selected[i] == 1) :
      for j in range(V) :
        if (selected[j] == 0) and (G[i][j] != 0) :
          if min > G[i][j] :
            min = G[i][j]
            x = i
            y = j
  print(str(x) + " - " + str(y) + " : " + str(G[x][y]))
  selected[y] = 1
  edge_count = edge_count + 1
```

# 3 이진 트리의 구현

나무 모양에 데이터를 담는 자료구조, Tree


## Tree

- ✓ **최상위 Root Node를 필두로,** 계층적으로 자료를 저장하는 구조
- ✓ 그래프의 한 종류로 볼 수 있으나, cycle을 갖지 않는다는 점에 유의
- ✓ 한 Node에서 세 갈래 이상으로 뻗어나가지 않는 경우 Binary

## Python을 이용해 Tree와 그 Node를 구현하기

### Binary Tree 구현을 위한 클래스 선언

```
class Node :

  def __init__(self, item) :
    self.data = item
    self.left = None
    self.right = None

class BinaryTree :

  def __init__(self, r) :
    self.root = r
```

## 주어진 Tree의 Size를 구하기 위해 생각해야 할 것은?

- ✓ **(Tree Size) = (왼쪽 Subtree의 Size) + (오른쪽 Subtree의 Size) + 1**
- ✓ 같은 과정을 재귀적으로 반복한다면?
- ✓ Size를 계산하는 구문은 어느 클래스에 포함되어야 할까?

## Tree의 Size 계산을 위한 Python에서의 클래스 활용

### Node

```
class Node :

  def size(self) :
    if self.left is not None :
      l = self.left.size()
    else :
      l = 0
    if self.right is not None :
      r = self.right.size()
    else :
      r = 0
    return (l + r + 1)
```

## Tree의 Size 계산을 위한 Python에서의 클래스 활용

### BinaryTree

```
class BinaryTree :

  def size(self) :
    if self.root is not None :
      return self.root.size()
    else :
      return 0
```

# Assignment #5

실시간 Zoom 수업을 통해 출제 예정

# Any Questions?

TA

TA TODAY 유용재 yooy@dm.snu.ac.kr

안영훈 younghoon.ahn@dm.snu.ac.kr