

자료구조 2주차 실습

Array

감성인공지능연구실
방윤석 임희수

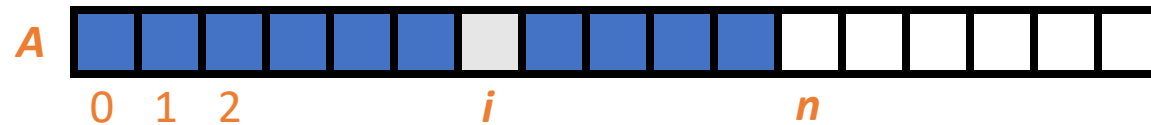


인하대학교
INHA UNIVERSITY

What is an Array?

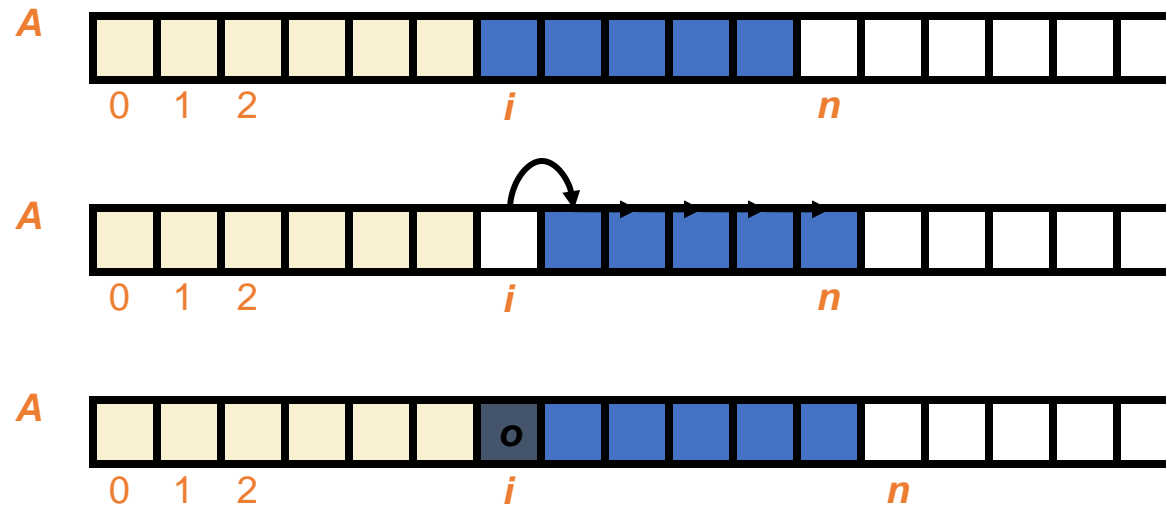


- A collection of data elements
 - All types of data elements are **homogeneous** (same type)
 - Elements (or their references) are stored at consecutive memory locations
 - can be addressed using consecutive indices, which, in Python, start with index 0

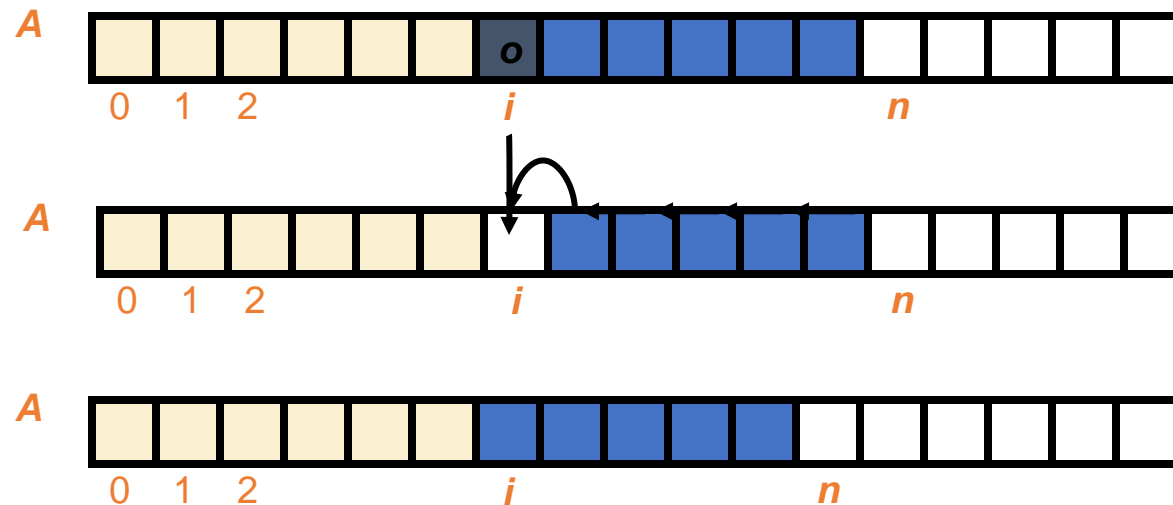


- Array is a static data structure
 - Cannot **grow** or **shrink** during program execution

- In an operation **add**(*i*, *o*), we need to make room for the new element by shifting forward the $n - i$ elements $A[i], \dots, A[n - 1]$
- In the worst case ($i = 0$), this takes $O(n)$ time



- In an operation **remove**(i), we need to fill the hole left by the removed element by shifting backward the $n - i - 1$ elements $A[i + 1], \dots, A[n - 1]$
- In the worst case ($i = 0$), this takes $O(n)$ time



Array : Methods

__init__ : 정수 n을 입력받아 n개의 메모리 공간을 갖는 배열 생성

Add : 현재 크기가 수용량보다 작다면 원소를 추가

Remove : array에 있는 i번째 원소를 제거.

print: 배열에 저장된 모든 데이터 출력.

find: 배열에 item이 존재하는지 탐색하여 해당 인덱스 출력.
여러 개 존재한다면 가장 작은 인덱스를 출력.
존재하지 않는다면 -1을 출력

replace: 배열의 인덱스 i에 저장된 데이터를 주어진 데이터로 대체.

full: 배열이 꽉 찼을 때 True를 반환.

```
import array
```

```
class ArrayList:
```

```
    def __init__(self,n):  
        self.capacity = n  
        self.array = array.array('h',[0]*self.capacity)  
        self.size = 0
```

```
    def add(self,idx,item):  
        if self.size < self.capacity:  
            self.size = self.size+1  
            for i in range(self.capacity-2,idx-1,-1):  
                self.array[i+1] = self.array[i]
```

```
        self.array[idx] = item  
        return self.array
```

```
    def remove(self,idx):  
        for i in range(idx+1, self.size):  
            self.array[i-1] = self.array[i]  
  
        self.size = self.size - 1  
        return self.array
```

```
    def print(self):  
        for i in range(0,self.capacity):  
            print(self.array[i],end=" ")
```

```
    def replace(self,idx,item):  
        self.array[idx] = item
```

```
    def find(self,item):  
        for i in range(self.size):  
            if self.array[i] == item:  
                return i  
        return -1
```

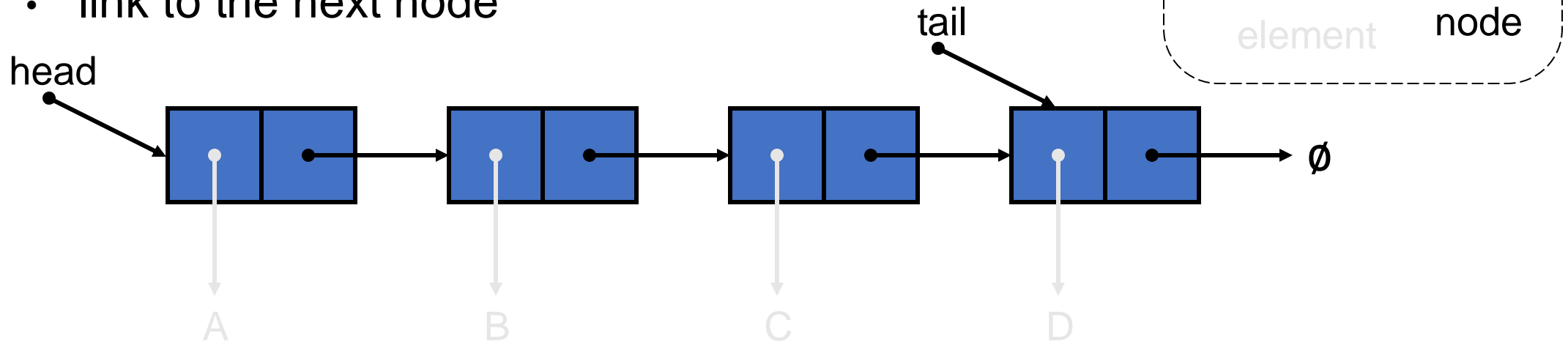
```
    def full(self):  
        return self.size == self.capacity
```

**배열이 꼭 찾을때 값을 중간에 값을 넣으면
무슨 일이 일어나는가?**

- A linear data structure where each elements is a separate object
 - Arrays are **static** structures, but cannot be easily extended or reduced to fit the data set.
 - Arrays are also **expensive** to maintain new insertions and deletions
- Types of Linked Lists
 - Singly linked list
 - Doubly linked list : a list which has two references, one to the next node and another to previous node.

Singly Linked List

- A singly linked list is a concrete data structure consisting of a sequence of nodes, starting from a head pointer
- Each node stores
 - element
 - link to the next node



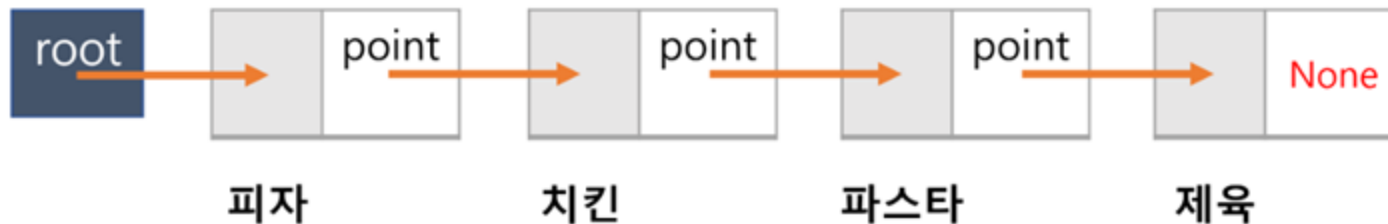
Singly Linked List



Linked List는 Node를 구현하고, 노드간 관계를 만들어주는 형식.

Linked List (연결 리스트)

- 노드로 연결된 자료구조
- 마지막 노드는 가리키는 곳이 없다.



- LinkedList는 각 객체가 다음에 위치한 객체의 **주소**를 가지고 있음.
 - head(root): LinkedList의 첫번째 객체의 주소를 가리킴.
 - tail: LinkedList의 마지막 객체의 주소를 가리킴.
- 오른쪽 그림은 위 LinkedList를 구현한 코드이다.
- 오른쪽 코드의 문제점은 무엇인가?

```
class Node():
    def __init__(self, item):
        self.item=item
        self.link=None

if __name__ == '__main__':
    node_pizza = Node('피자')
    node_chicken = Node('치킨')
    node_pasta = Node('파스타')
    node_jeyuk = Node('제육')

    node_pizza.link= node_chicken
    node_chicken.link = node_pasta
    node_pasta.link = node_jeyuk

    print(node_pizza.item)
    print(node_pizza.link.item)
    print(node_pizza.link.link.item)
    print(node_pizza.link.link.link.item)
```

- ADT는 데이터와 그 데이터에 대해 수행할 수 있는 연산들을 함께 묶어 정의한 것
- ADT의 핵심은 데이터의 구체적인 구현 방법을 숨기고, 데이터 타입의 인터페이스만을 사용자에게 제공하는 것
- 즉, ADT는 '무엇을 할 수 있는가'에 초점을 맞추며, '어떻게 하는가'에 대한 구현 세부 사항은 숨긴다.

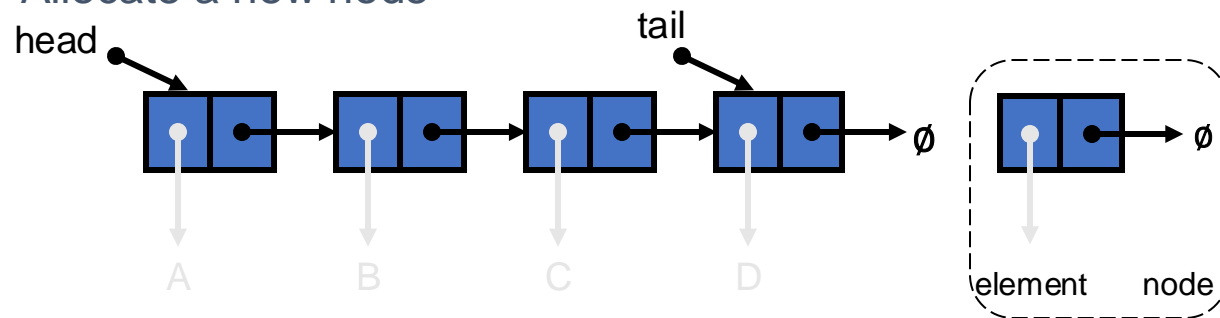
- 내가 리스트의 모든 아이템을 출력하려면 예제와 같은 방법을 사용하도 되지만 (예를 들어 2~3번째만 출력하고 싶은 경우)
- 사용자는 일일이 첫번째 줄과 4번째 줄을 삭제 해 줘야 한다.

```
print(node_pizza.item)
print(node_pizza.link.item)
print(node_pizza.link.link.item)
print(node_pizza.link.link.link.item)
```

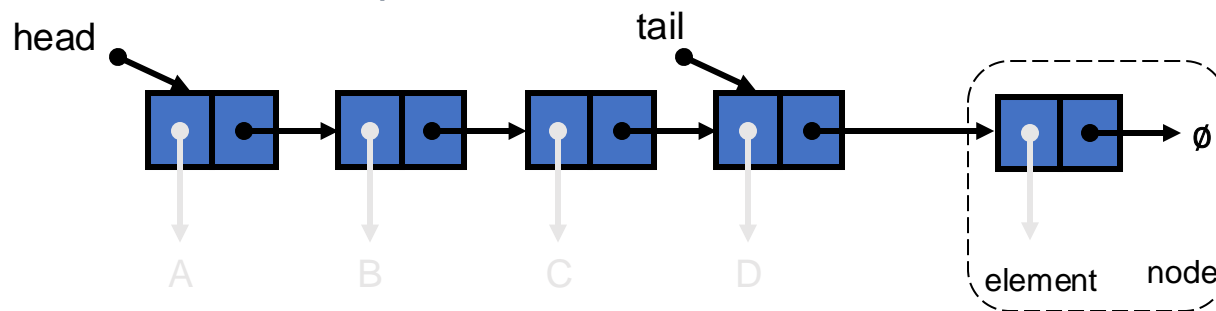
- 하지만 ADT를 활용하면 `show_list(:)`, `show_list(2:3)` 같은 형식으로 파라미터만 조정해도 출력을 조정할 수 있다.

Inserting at the Tail

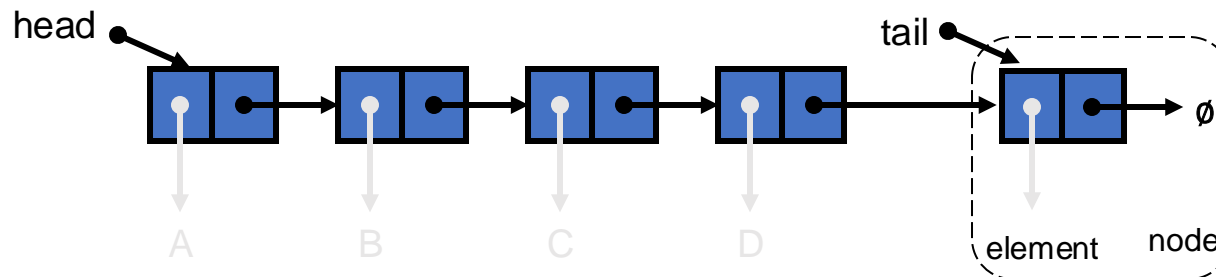
1. Allocate a new node



1. Have old last node point to new node



1. Update tail to point to new node



Singly Linked List

- Linked List는 노드를 통해 연결
- Head와 tail로 시작 과 끝을 표시
- Head가 가르키는게 없으면?
- Head랑 tail이랑 같은걸 가르키면?

```
class Node: # Node class
    def __init__(self, data):
        self.data = data
        self.link = None

# TIP : None is considered as False in Python

class LinkedList: # Linked list class
    def __init__(self): # Initialize
        self.head = None
        self.tail = None

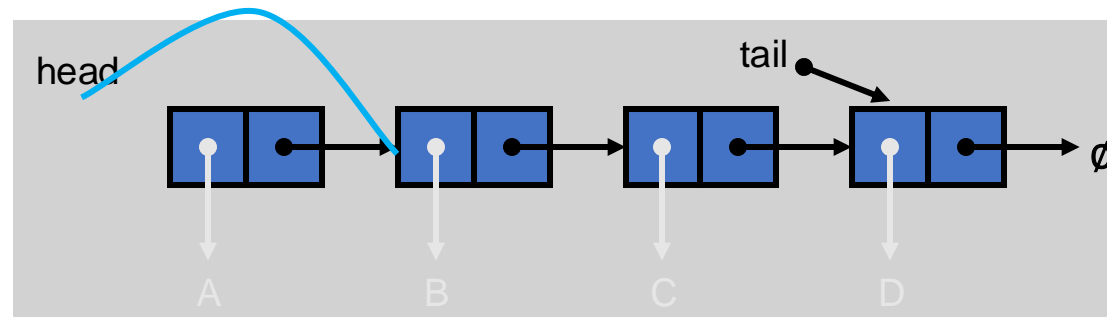
    def append(self, data): # Add new node at the end
        new_node = Node(data)

        # If there is no node in the linked list
        if not self.head:
            self.head = new_node
            self.tail = new_node
            return

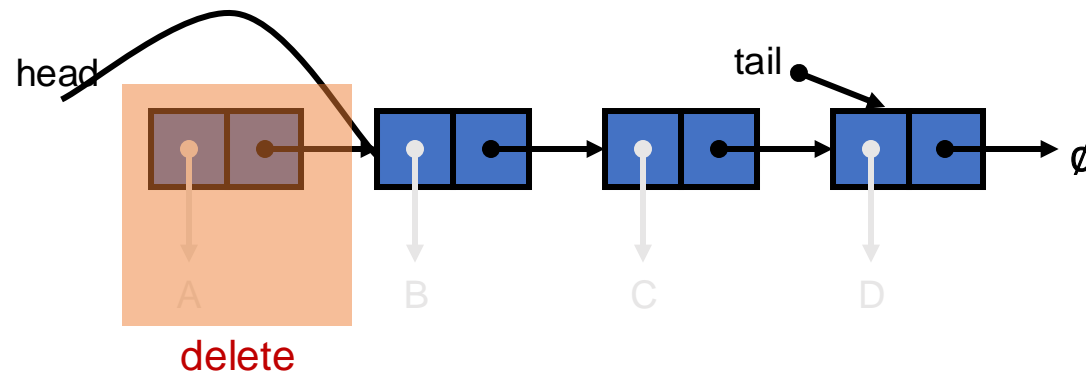
        # Add new node at the tail
        self.tail.link = new_node
        self.tail = new_node
```

Removing at the Head

1. Update head to point to next node in the list



1. Allow garbage collector to reclaim the former first node



Singly Linked List

- Delete : 연결된 노드를 끊어주고, 다시 연결 해 줘야 함

```
def delete(self, data): # Delete the node with specific value
    temp = self.head

    # Check if the head is the node to delete
    if temp and temp.data == data:
        self.head = temp.link
        if self.head is None:
            self.tail = None
        temp = None
        return

    # Check if the node is in the middle or the tail
    prev = None
    while temp and temp.data != data:
        prev = temp
        temp = temp.link

    if temp is None:
        return

    # Delete the node
    prev.link = temp.link
    if temp == self.tail:
        self.tail = prev
    temp = None
```

- Search : 리스트를 돌아다니며 값이 있으면 True를 리턴
- Display : 링크의 구조를 보여줌

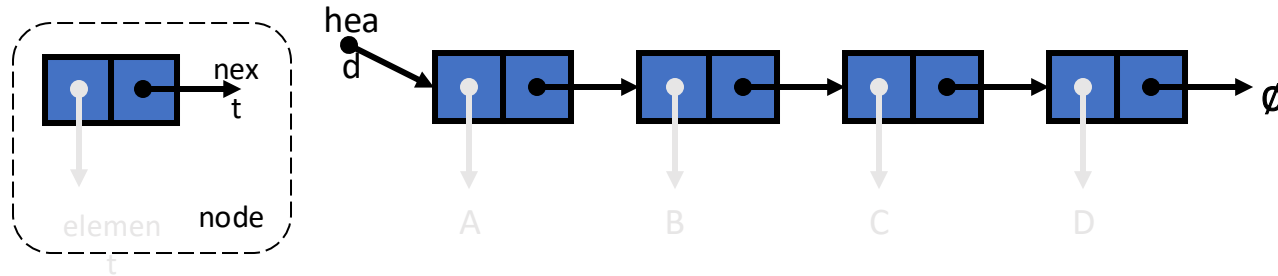
```
def search(self, data): # Search the node with specific value
    temp = self.head
    while temp:
        if temp.data == data:
            return True
        temp = temp.link
    return False

def display(self): # Display the linked list
    temp = self.head
    while temp:
        print(temp.data, end=' -> ')
        temp = temp.link
    print('None')
```

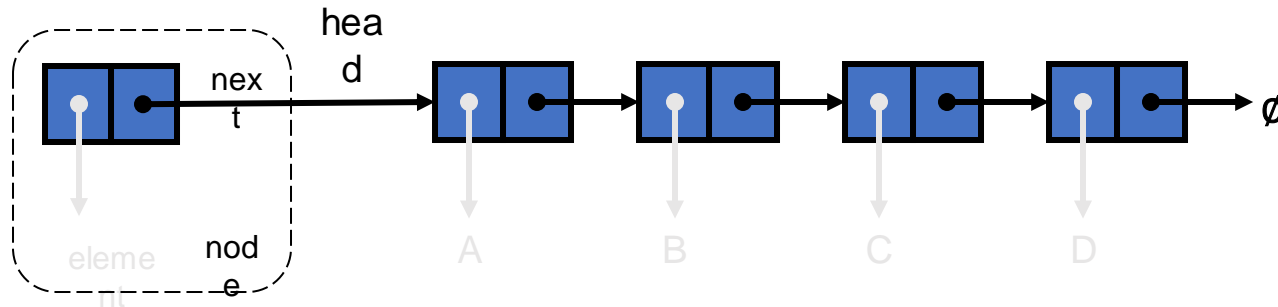
- 다음 슬라이드를 참고하여 Head 앞에 노드를 추가하는 prepend를 구현해보세요

Inserting at the Head

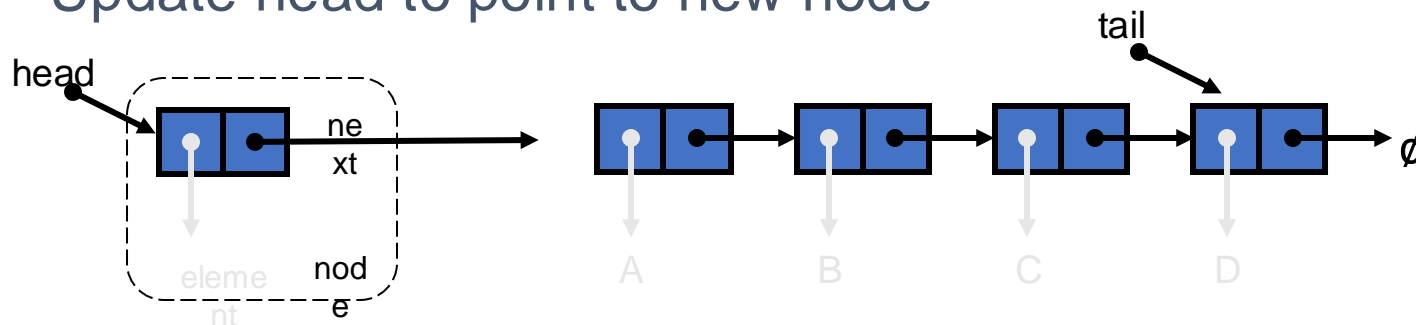
1. Allocate a new node



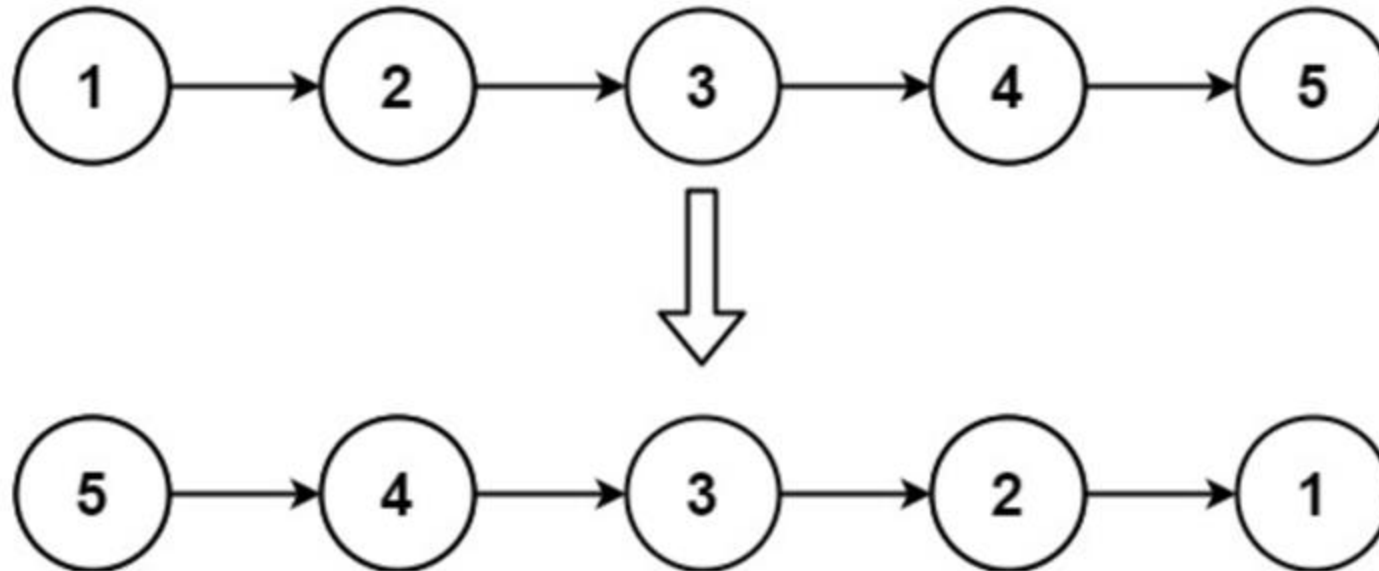
1. Have new node point to old head



1. Update head to point to new node



- Linked List에 Reverse하는 기능을 구현해보세요

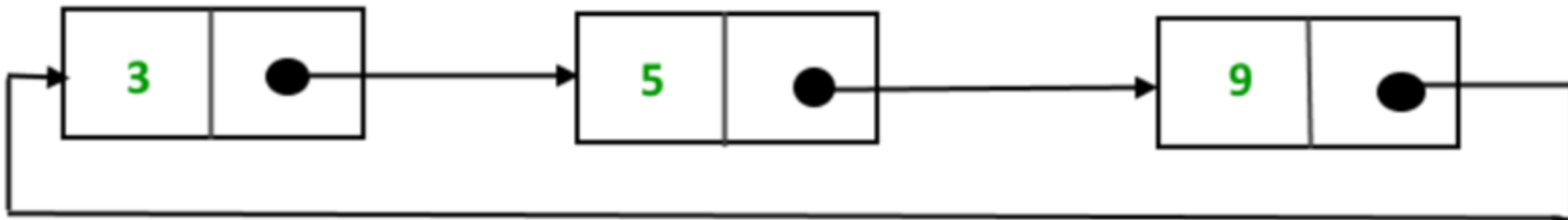


- 거꾸로 print하라는 뜻이 아님. 구조를 뒤집으라는 뜻임

Circularly Linked Lists



- Circularly Linked List는 커서가 필요
- 노드를 삭제할시, 커서 다음에 연결된 노드를 지울 것



문제

지혜는 동아리 회의를 위해 출석부에 친구들의 이름을 기록했다. 그런데 바쁘게 출석을 기록하다 보니 같은 친구의 이름이 여러 번 중복되어 적힌 걸 발견했다. 지혜는 회의록을 깔끔하게 정리하고 싶어 중복된 이름을 제거하기로 했다. 이를 위해 환형 연결 리스트 자료구조를 이용하여 다음의 기능을 수행하는 메소드를 구현하시오.

- 연결 리스트에 이름을 추가할 수 있다.
- 연결 리스트에서 중복된 이름을 제거할 수 있다.
- 현재 연결 리스트의 이름을 순서대로 출력할 수 있다.

메소드 설명

- `add(String name)` : 연결 리스트의 맨 끝에 이름을 추가한다.
- `removeDuplicates()` : 연결 리스트에 있는 중복된 이름을 제거한다. 제거 후 리스트에는 중복되지 않은 이름만 남아 있어야 하며, 최초 등장한 이름의 순서를 유지한다.
- `show()` : 연결 리스트의 이름을 순서대로 공백으로 구분하여 반환한다. 이름이 없으면 ``empty``를 반환한다.

Problem #1



인하대학교
INHA UNIVERSITY

입력 예시

```
name_list1 = NameList()
name_list1.insert("지혜")
name_list1.insert("민수")
name_list1.insert("유나")
name_list1.insert("민수")
name_list1.insert("태훈")
name_list1.insert("유나")
name_list1.display()
name_list1.removeDuplicates()
name_list1.display()
```

출력 예

```
지혜 민수 유나 민수 태훈 유나
지혜 민수 유나 태훈
```

문제

수민이는 정수형을 저장하는 dynamic array를 구현하고자 하였다. 컴퓨터에 남은 메모리를 확인한 결과, 여러개의 빈 공간을 확인할 수 있었다. 하지만, 모든 공간이 **정수형 3개와 포인터 하나만** 할당 될 수 있었다.

그래서 수민이는 정수형 3개씩 저장하는 linked list를 구현해서 dynamic array처럼 동작하도록 하기로 하였다.

- 각 노드는 정수형 3개를 저장할 수 있다.
- 마지막 노드를 제외한 모든 노드는 빈 배열을 가지고 있지 않는다.
- 연결 리스트에 정수 데이터를 추가할 수 있다.
- 연결 리스트에서 특정 index의 값을 제거할 수 있다.
- 현재 연결 리스트의 이름을 순서대로 출력할 수 있다


메소드 설명

- `add(int idx, int value)` : 리스트에 값을 추가한다.
- `remove(int idx)` : index가 idx인 값을 제거한다. idx 뒤의 값들은 모두 한칸씩 앞으로 이동시킨다.
- `show()` : 리스트에 들어있는 모든 값을 출력한다.

입력 예시

```
arr_list = DynamicArray()
arr_list.add(0, 1)
arr_list.add(1, 2)
arr_list.add(2, 3)
arr_list.add(3, 4)
arr_list.add(4, 5)
arr_list.print()
```

출력 예



1 2 3 4 5