รายการโยง (Linked Lists)

NC252 โครงสร้างข้อมูลและขั้นตอนวิธี (Data Structures & Algorithms)

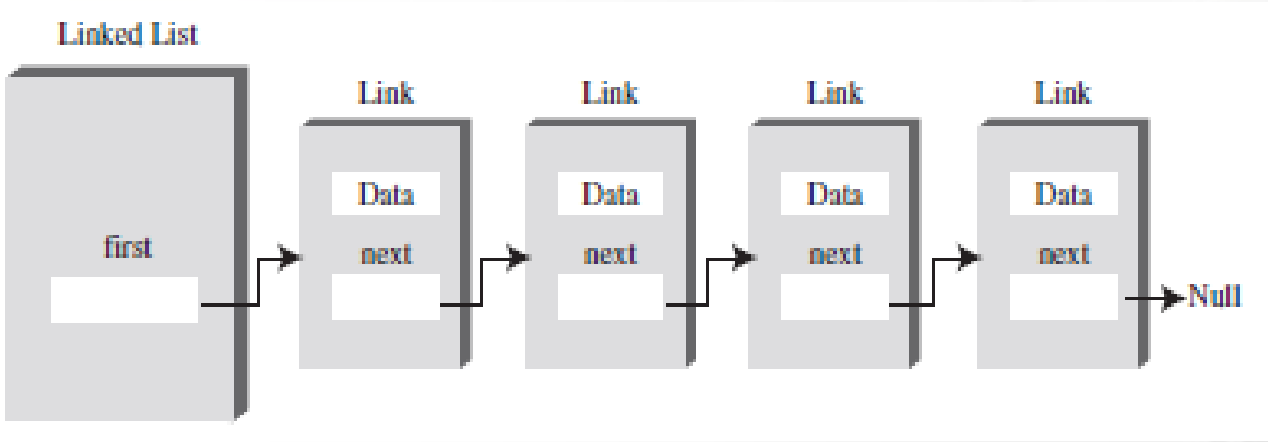
ผศ.ดร.ศุภฤกษ์ มานิตพรสุทธ์

มหาวิทยาลัยหอการค้าไทย
University of the Thai Chamber of Commerce.

Topics

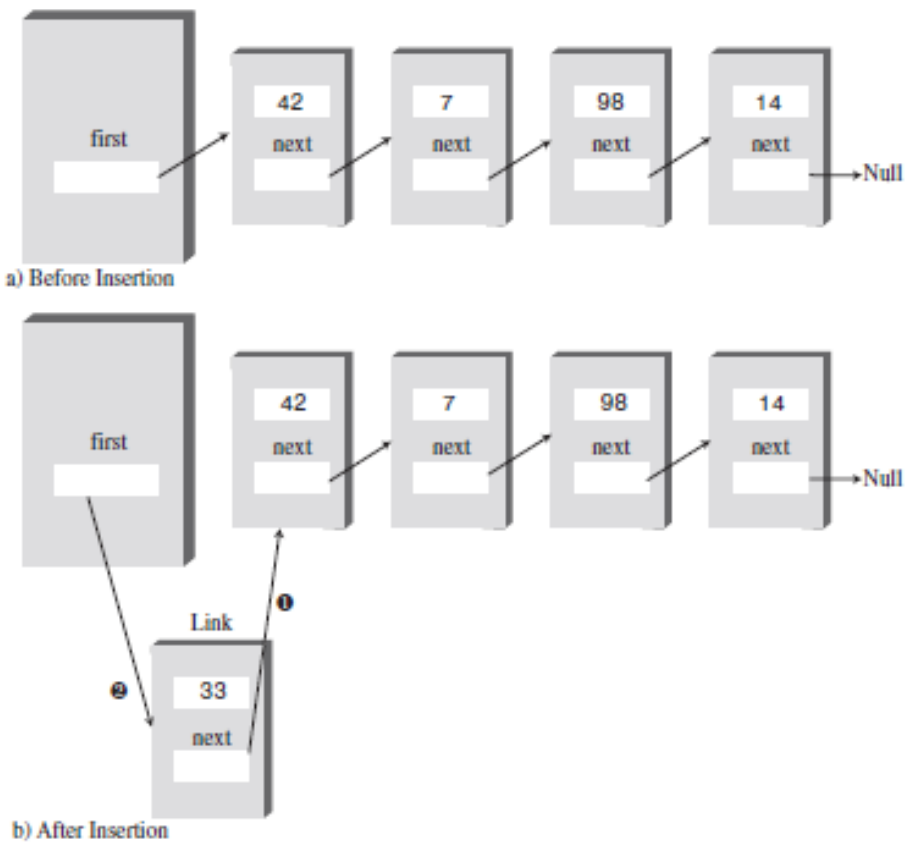
- Simple Linked List
- Double-Ended Lists
- Linked List Efficiency
- Sorted List
- Doubly Linked Lists

Simple Linked List



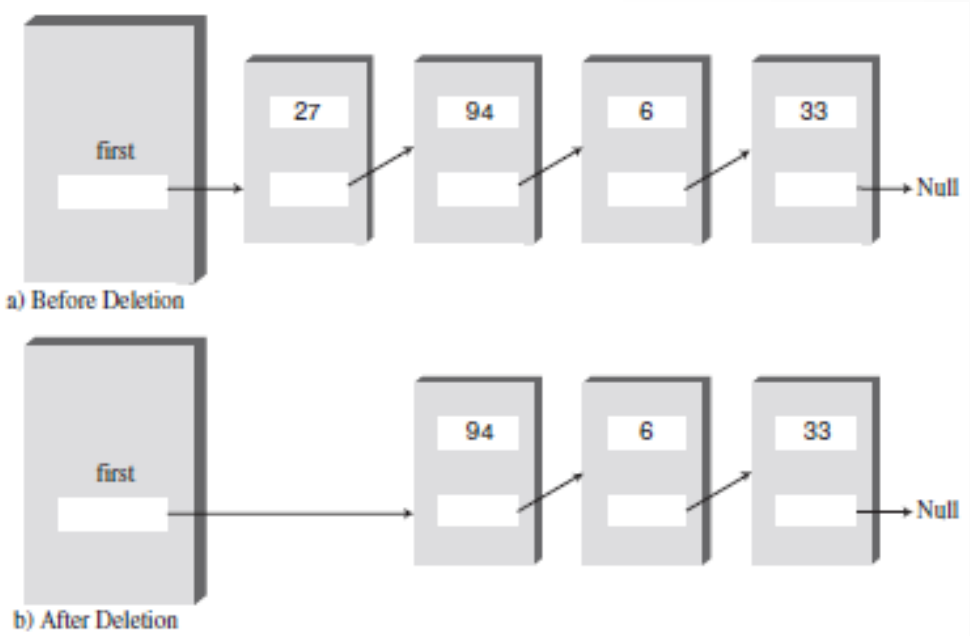
- First: เป็นจุดเริ่มต้นของรายการโยง
- ข้อมูลมีตัวชี้ เพื่อชี้ไปยังข้อมูลตัวถัดไป
- ข้อมูลตัวสุดท้ายในรายการ ตัวชี้เป็น Null

Linked List: insertFirst



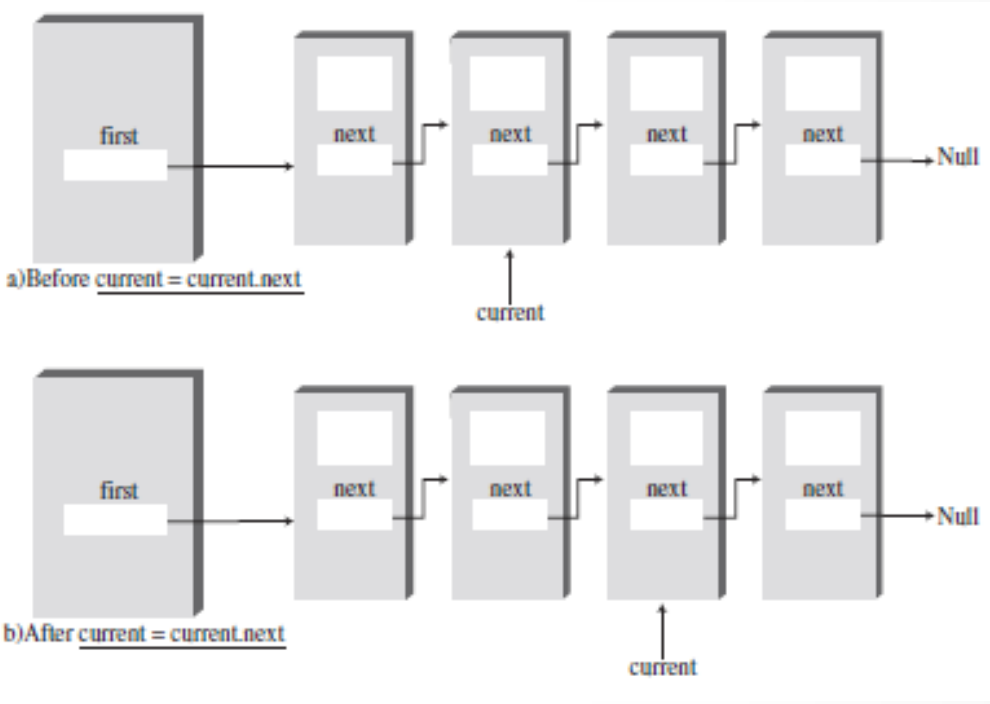
- การแทรกข้อมูลในรายการโยง
  - สร้างข้อมูลใหม่ให้มีโครงสร้างเช่นเดียวกับรายการโยง
  - กำหนดให้ next ของข้อมูลใหม่เท่ากับ next ของ Front
  - กำหนด next ของ Front ให้ชี้ไปที่ข้อมูลใหม่

Linked List: deleteFirst



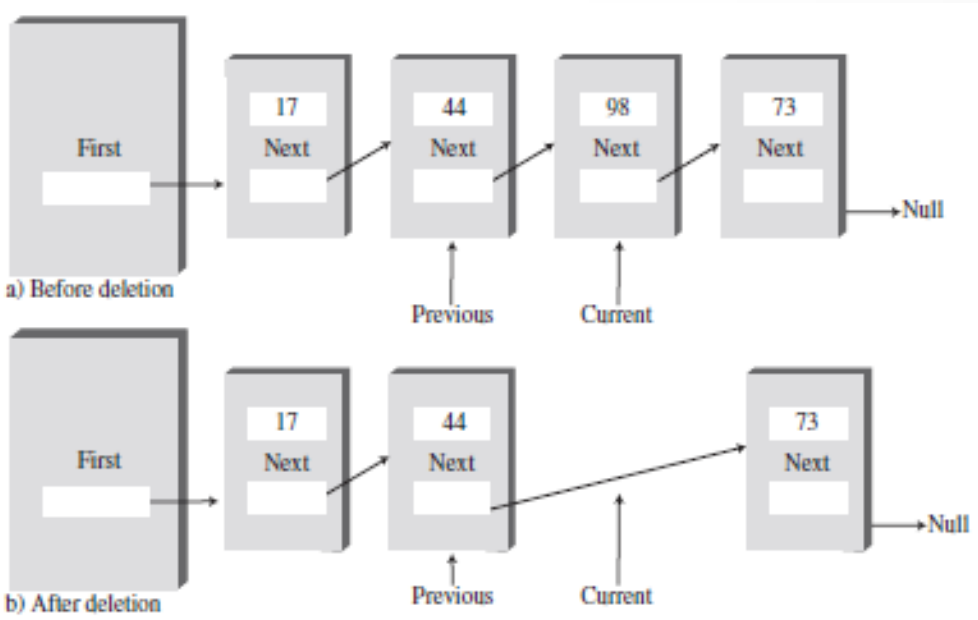
- การลบข้อมูลในรายการโยง
  - กำหนด next ของ Front ให้เท่ากับ next ของข้อมูลแรก
  - ลบข้อมูลแรก

Linked List: Search



- การค้นหา
  - ใช้ current เป็นตัวชี้
  - การเลื่อนไปตำแหน่งถัดไปทำได้โดย current = current.next

Linked List: Delete



- การลบ
  - ค้นหาข้อมูลที่ต้องการลบ (อัพเดท current และ previous)
  - กำหนดให้ previous.next เท่ากับ current.next
  - ลบข้อมูลที่ current ชี้อยู่

Linked List: Experiment

```
class Node:
    def __init__(self,initdata):
        self.data = initdata
        self.next = None

    def getData(self):
        return self.data

    def getNext(self):
        return self.next

    def setData(self,newdata):
        self.data = newdata

    def setNext(self,newnext):
        self.next = newnext
```

- คลาส Node
  - data ใช้สำหรับเก็บข้อมูล
  - next ใช้สำหรับชี้ไปยัง Node อันถัดไป

Linked List: Experiment

```
class UnorderedList:

    def __init__(self):
        self.head = None

    def isEmpty(self):
        return self.head == None

    def add(self,item):
        temp = Node(item)
        temp.setNext(self.head)
        self.head = temp

    def size(self):
        current = self.head
        count = 0
        while current != None:
            count = count + 1
            current = current.getNext()

        return count
```

```
def search(self,item):
    current = self.head
    found = False
    while current != None and not found:
        if current.getData() == item:
            found = True
        else:
            current = current.getNext()

    return found
```

```
def remove(self,item):
    current = self.head
    previous = None
    found = False
    while not found:
        if current.getData() == item:
            found = True
        else:
            previous = current
            current = current.getNext()

    if previous == None:
        self.head = current.getNext()
    else:
        previous.setNext(current.getNext())
```

Linked List: Efficiency

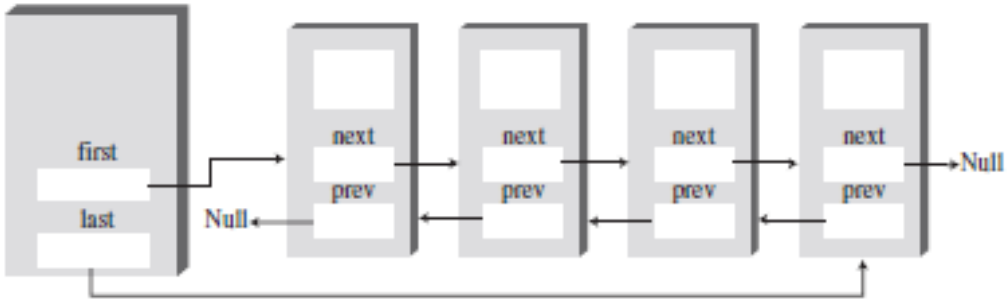
- Insert:
  - At the end/front: O(1)
  - At any specific location: O(N)
- Delete: O(N)
- Search: O(N)

Linked List vs. Array

- อาร์เรย์:
  - ข้อมูลสามารถเข้าถึงได้โดยตรงโดยใช้หมายเลขดัชนี (Index)
  - ขนาดของข้อมูลถูกกำหนดไว้ล่วงหน้า การเพิ่ม/ลดจำนวนข้อมูลทำให้เสียเวลามาก
- รายการโยง:
  - การเข้าถึงข้อมูลในรายการโยง จะต้องค้นหาผ่านตลอดรายการโยง
  - ขนาดของข้อมูลปรับเปลี่ยนได้ง่าย

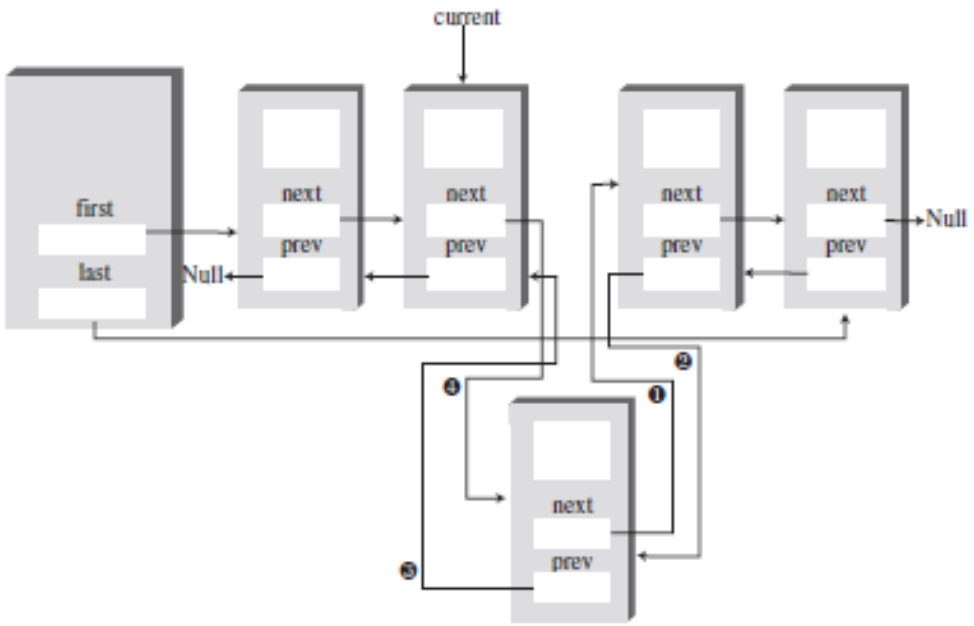
Doubly Linked List

- ปัญหาสำคัญของ Simple Linked List คือ ไม่สามารถวนกลับทางได้
- วิธีแก้คือ ใช้ Doubly Linked List



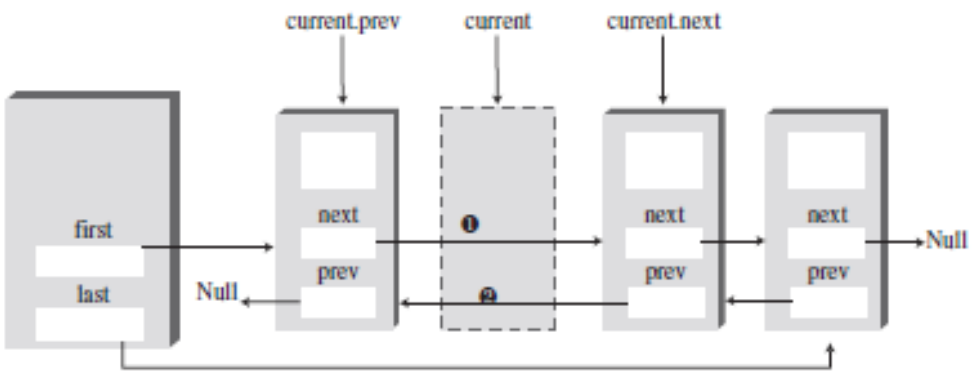

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
        self.prev = None
```

Doubly Linked List: insertAfter




```
def insert_after(self, ref_node, new_node):
    new_node.prev = ref_node
    if ref_node.next is None:
        self.last = new_node
    else:
        new_node.next = ref_node.next
        new_node.next.prev = new_node
    ref_node.next = new_node
```

Doubly Linked List: delete / display

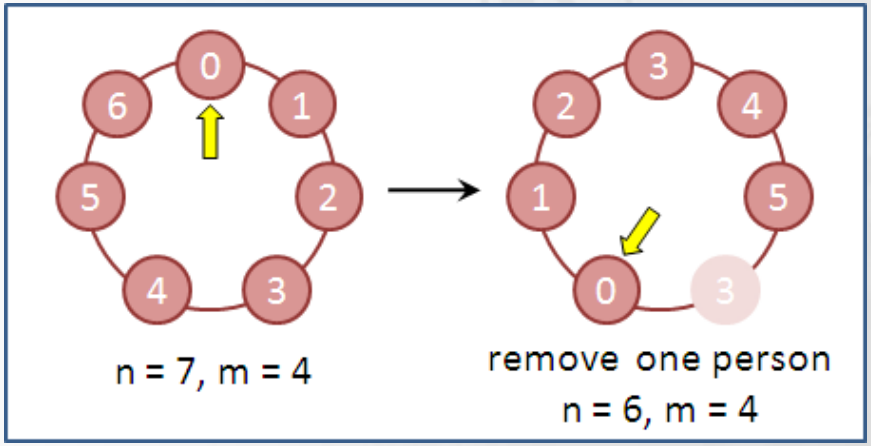



```
def remove(self, node):
    if node.prev is None:
        self.first = node.next
    else:
        node.prev.next = node.next

    if node.next is None:
        self.last = node.prev
    else:
        node.next.prev = node.prev
def display(self):
    current = self.first
    while current:
        print(current.data, end = ' ')
        current = current.next
```

Programming Assignment

- จงเขียนโปรแกรมเพื่อแก้ปัญหาต่อไปนี้
  - ให้โปรแกรมรับอินพุตเป็นจำนวน n และ m โดย n คือจำนวนโหนดทั้งหมด และ m คือจำนวนเต็มใดๆ ที่ m < n เช่น n = 7, m = 4 เป็นต้น
  - เมื่อนับจากจุดปัจจุบันไป m ตำแหน่ง ลบตำแหน่งนั้นไป แล้ววนซ้ำขั้นตอนนี้จนเหลือโหนดสุดท้าย
  - คำถามคือ โหนดที่เหลือโหนดสุดท้ายคือโหลดหมายเลขใด?



Hint: Circular Linked List