COSC 2306
Data Programming

Linked Lists

Queue with Linked list

- Implement a queue with the Linked list class
  - tips: insert and remove elements on 2 sides
  - enqueue: append (add to the tail)
  - dequeue: pop (remove the head node)
  - size: check the count

Queue with Linked list

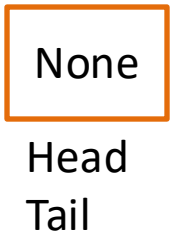
- Implement a queue with the Linked list class
  - tips: insert and remove elements on 2 sides
  - O(1) time cost for both enqueue() and dequeue()?
  - Solution: add a tail member for enqueue

Linked list with tail

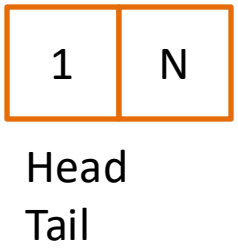
```
class LinkedList_2:

    def __init__(self):
        self.head = None
        self.tail = None


    def insert_head(self, value):
        newnode = Node(value)
        newnode.next = self.head
        self.head = newnode
        if self.tail == None:
            self.tail = self.head


    def pop(self):
        if self.isEmpty():
            return
        self.head = self.head.next
        if self.head == None:
            self.tail = None
```

Linked list with tail

- A = LinkedList_2()

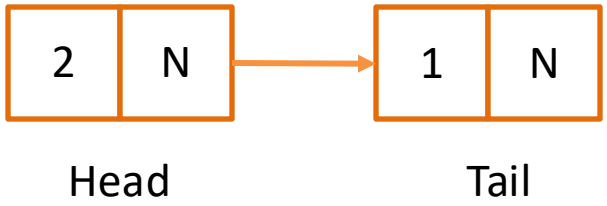


- A.insert_head(1)

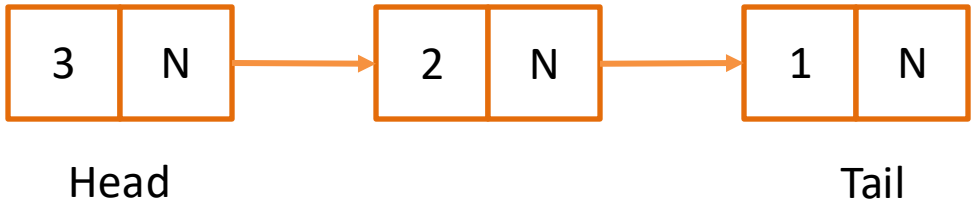


Queue with Linked list

- A.insert_head(2)

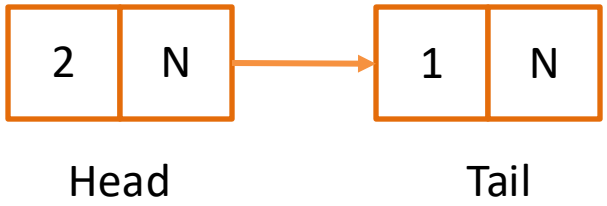


- A.insert_head(3)



Linked list with tail

- A.pop()



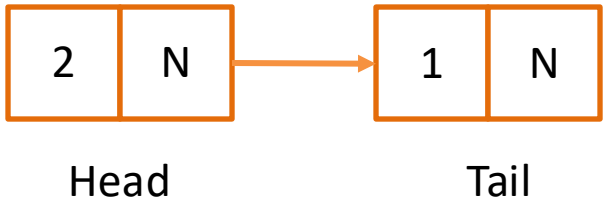
- A.pop()
- A.pop()

Linked list with tail

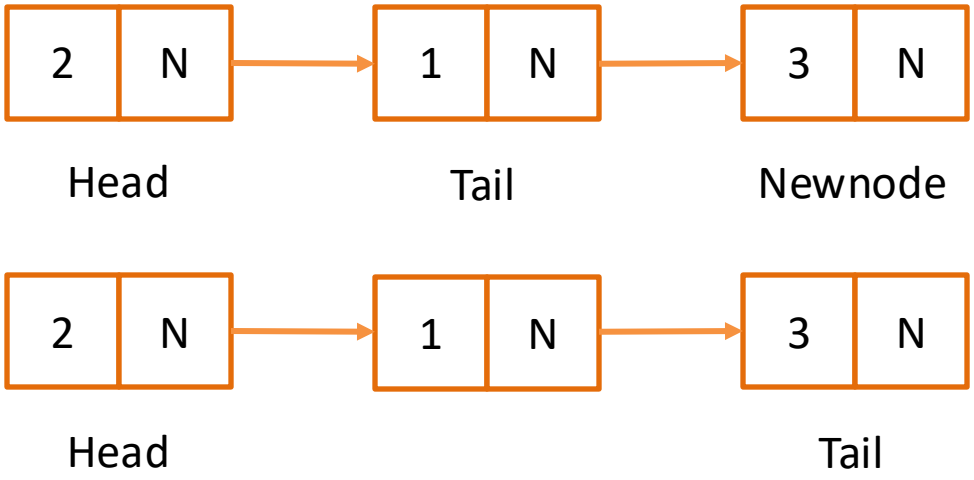
```
class LinkedList_2:


    def insert_tail(self, value):
        newnode = Node(value)
        if self.tail == None:
            self.tail = newnode
            self.head = newnode
        else:
            self.tail.next = newnode
            self.tail = newnode
```

Linked list with tail

- A



- A.insert_tail(3)



Linked list with tail

```
class LinkedList_2:
    def __init__(self):
        self.head = None
        self.tail = None

    def insert_head(self, value):
        newnode = Node(value)
        newnode.next = self.head
        self.head = newnode
        if self.tail == None:
            self.tail = self.head

    def insert_tail(self, value):
        newnode = Node(value)
        if self.tail == None:
            self.tail = newnode
            self.head = newnode
        else:
            self.tail.next = newnode
            self.tail = newnode
```

```
def pop(self):
    if self.isEmpty():
        return None
    headdata = self.head.data
    self.head = self.head.next
    return headdata
```

Queue with tailed linked list

- Tail as end (enqueue)
  - enquque() <-> insert_tail()
- Head as front (dequeue)
  - dequeue <-> pop()

Queue with tailed linked list

```
class QueuewithLinkedList:

    def __init__(self):
        self.items = LinkedList_2()

    def isEmpty(self):
        return self.items.isEmpty()
    def size(self):
        return self.items.size()


    def enqueue(self, item):
        self.items.insert_tail(item)
    def dequeue(self):
        return self.items.pop()
```