Unit II
Data Structure

Linked List

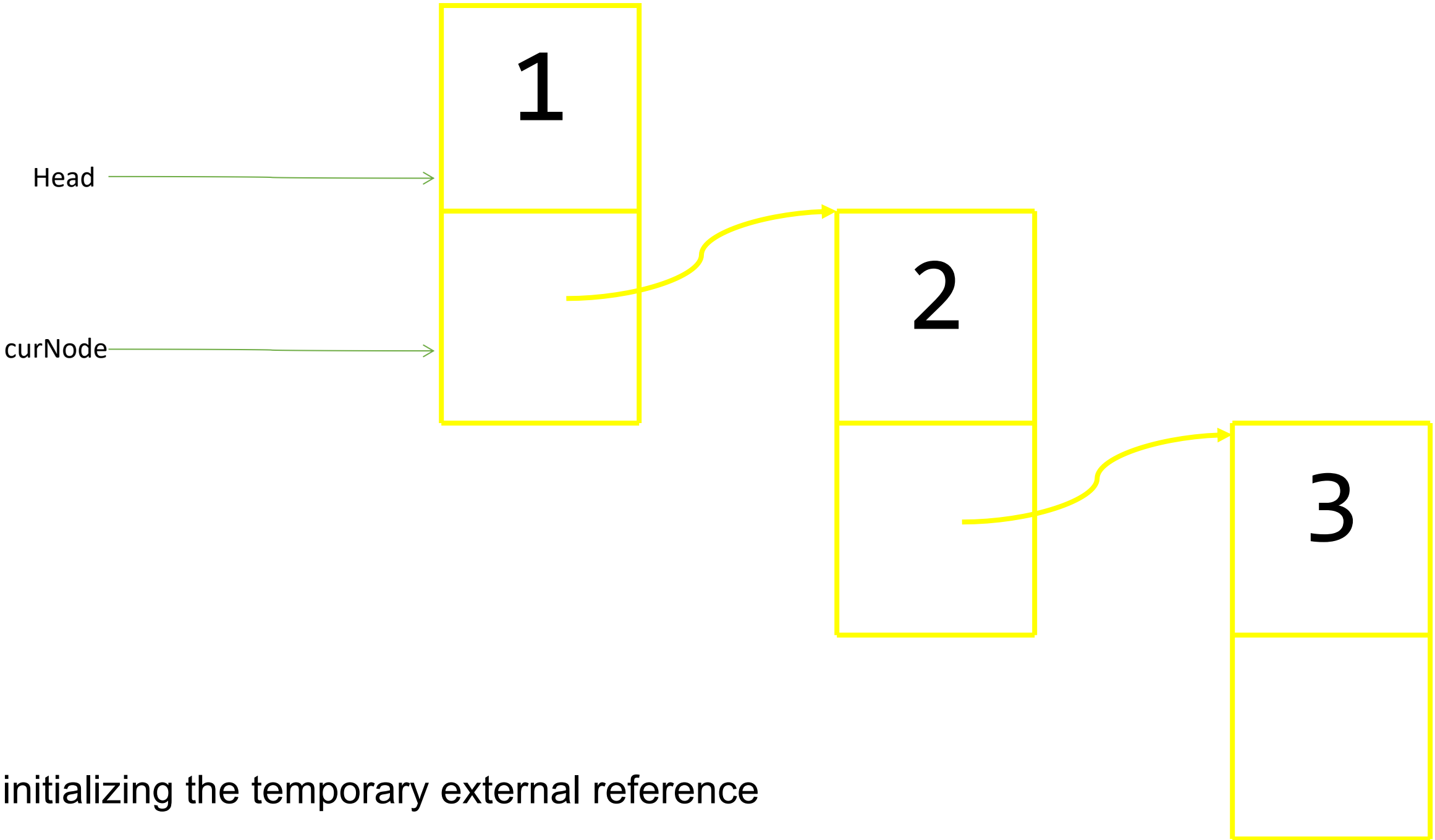
Operations on Linked List:

1. Traversing the Nodes
2. Searching for a Node
3. Prepending Nodes
4. Removing Nodes

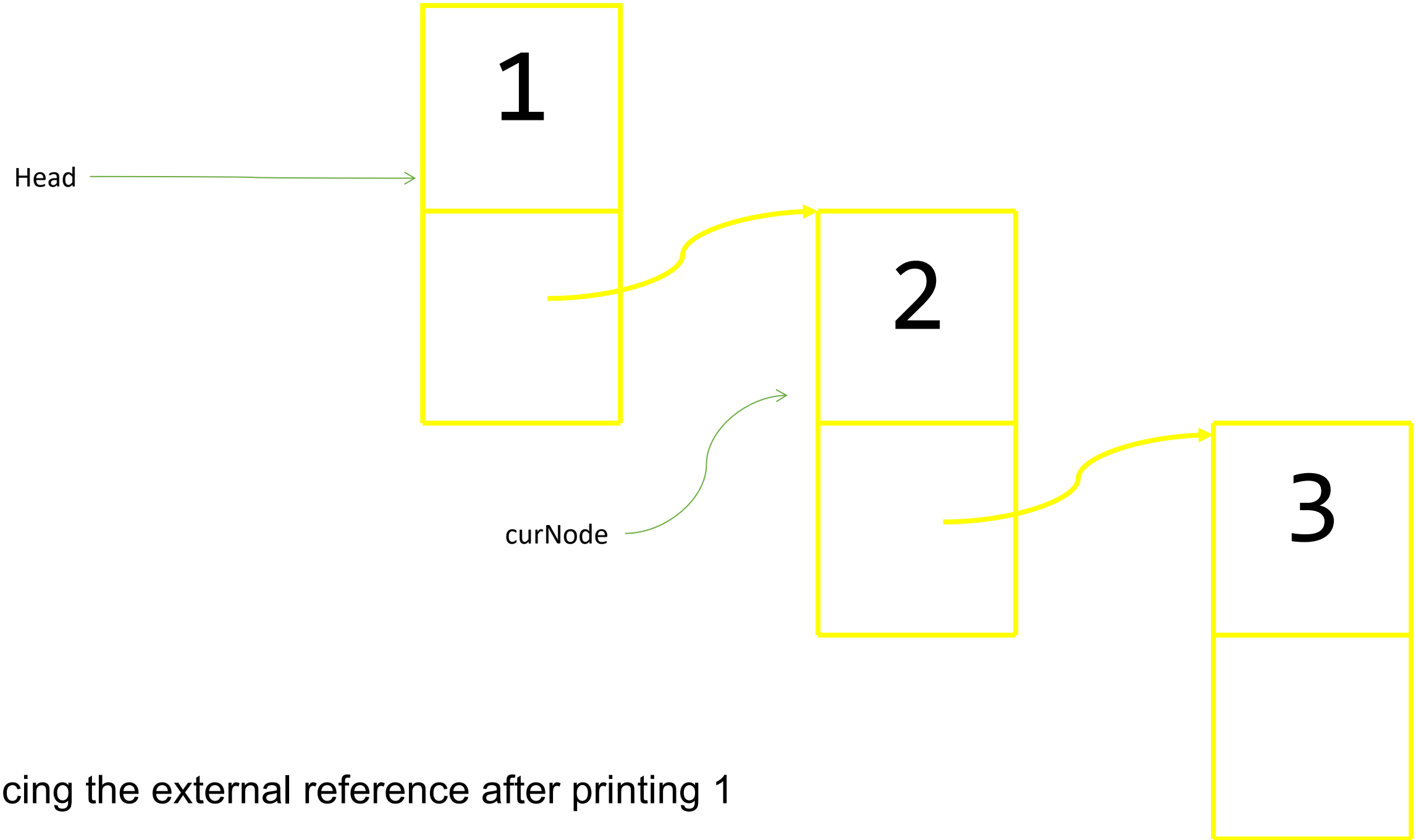
Operations on Linked List:

1. **Traversing the Nodes**

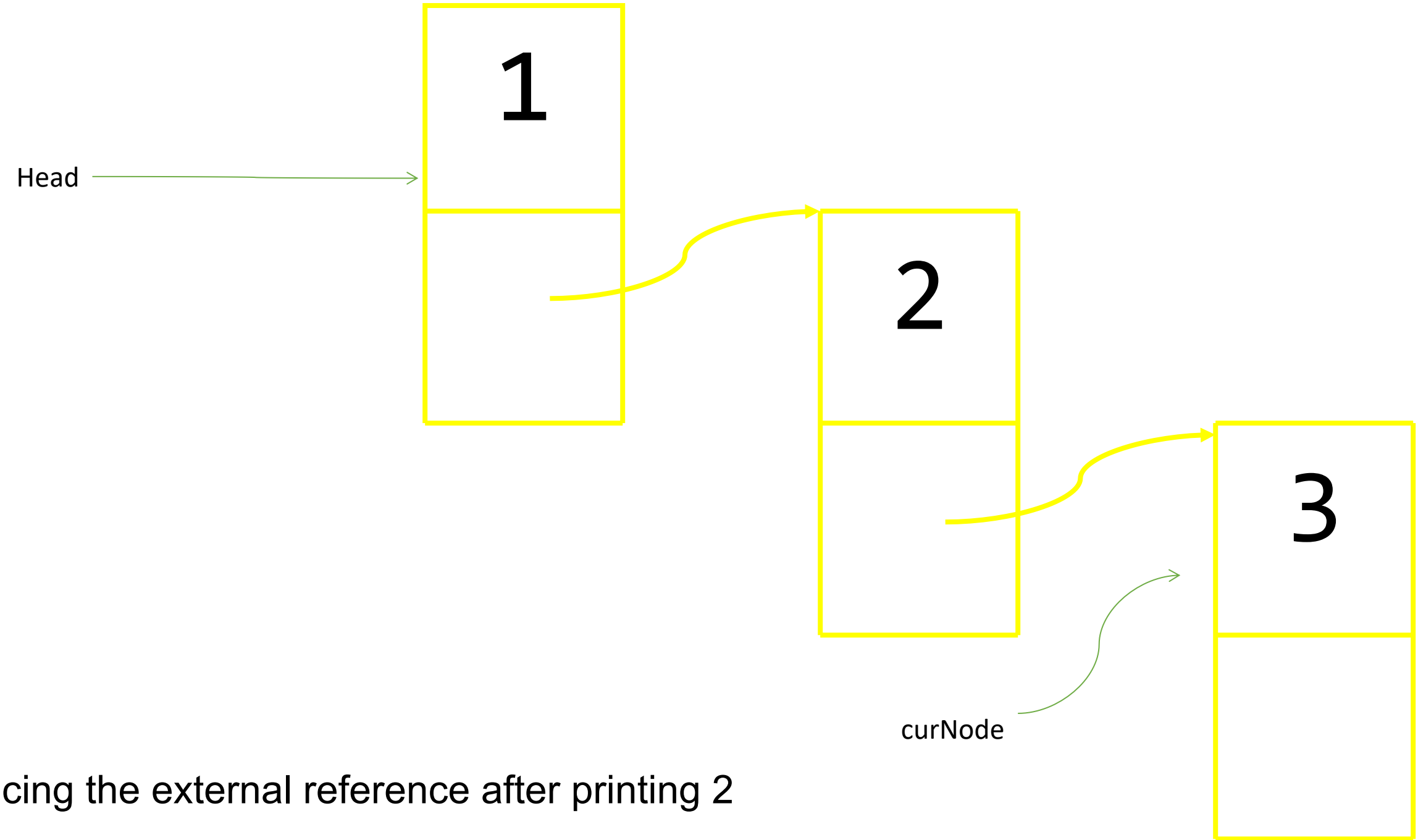
Traversing a linked list requires the initialization and adjustment of temporary external reference variable.



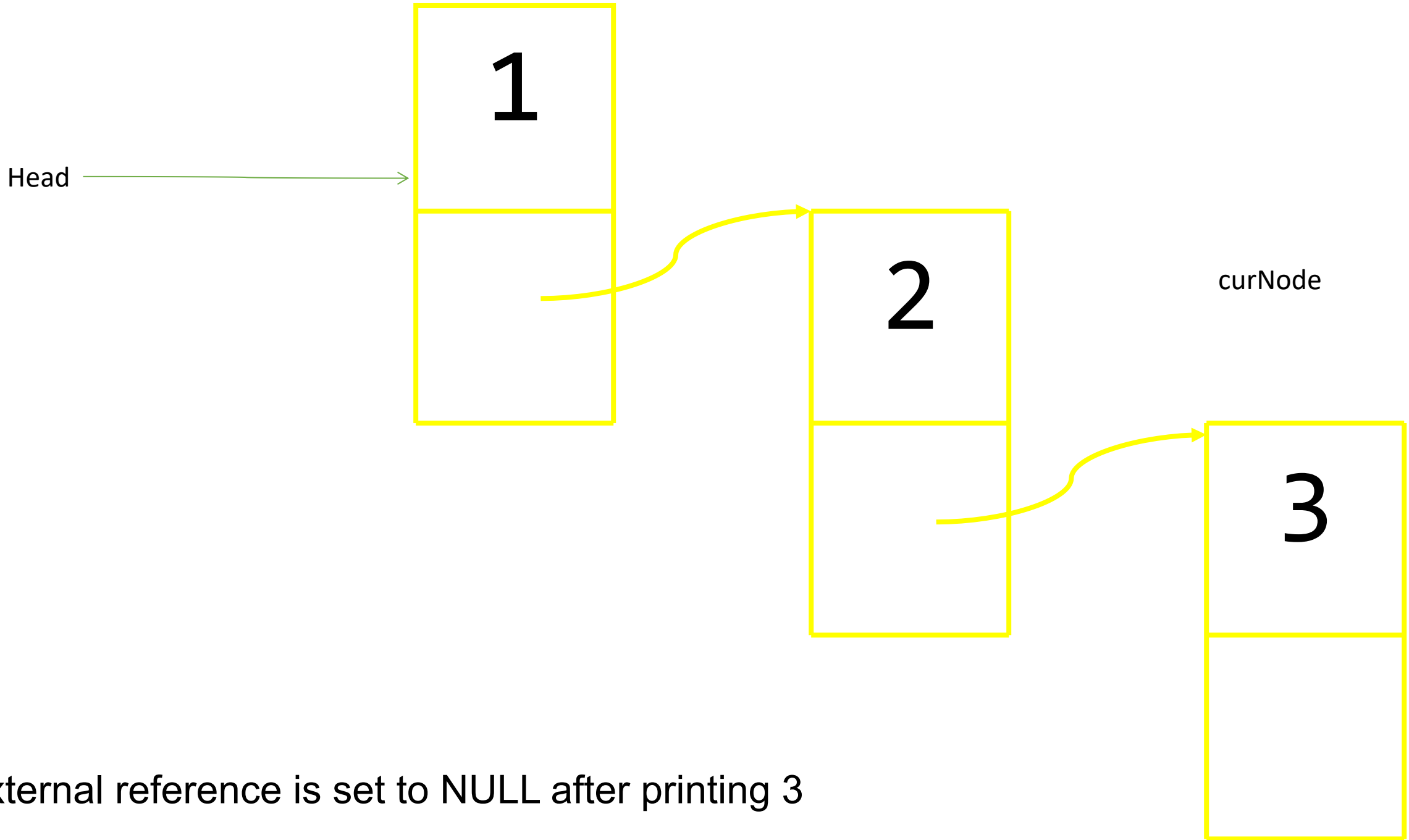
a. After initializing the temporary external reference



b. Advancing the external reference after printing 1

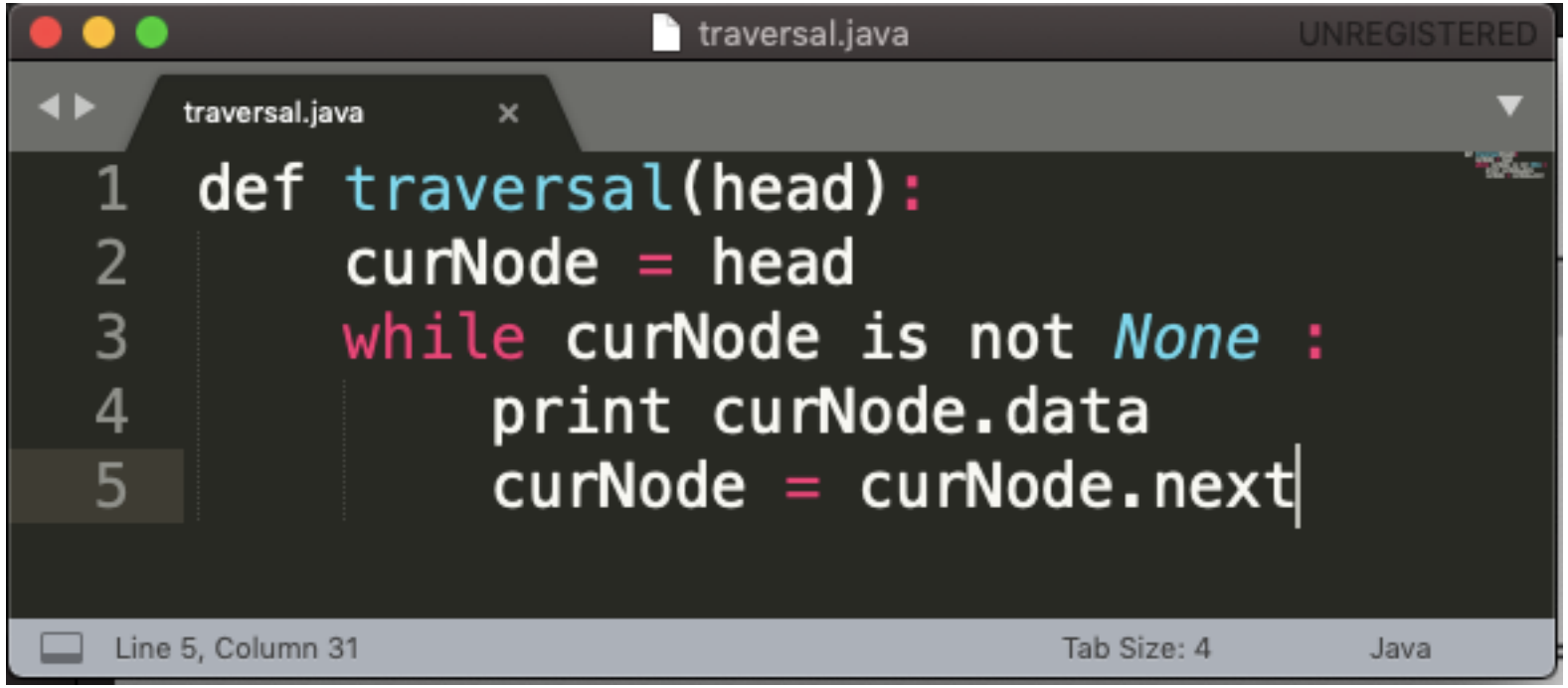


c. Advancing the external reference after printing 2



d. The external reference is set to NULL after printing 3

Algorithms ?



Time Complexity???

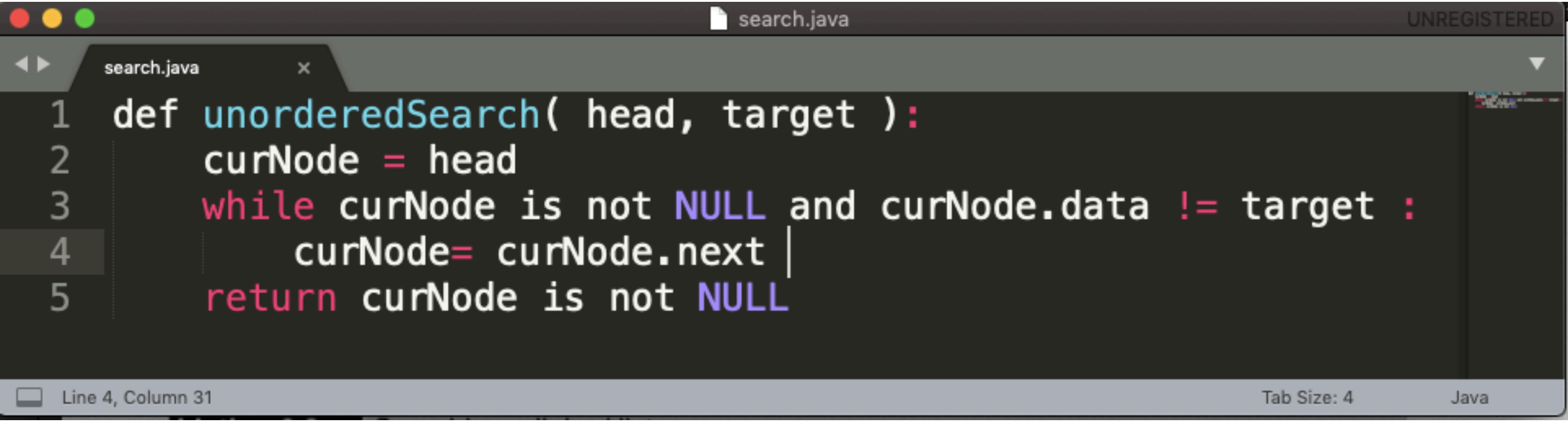
O(n)

Operations on Linked List:

2. Searching for the Nodes

A Linear search operation can be performed on a linked list. It is very similar to the traversal demonstrated earlier. The only difference is that the loop can be terminated early if we find the target value within the list.

Algorithms ?



Time Complexity???

O(n)

Operations on Linked List:

3. Perpending Nodes

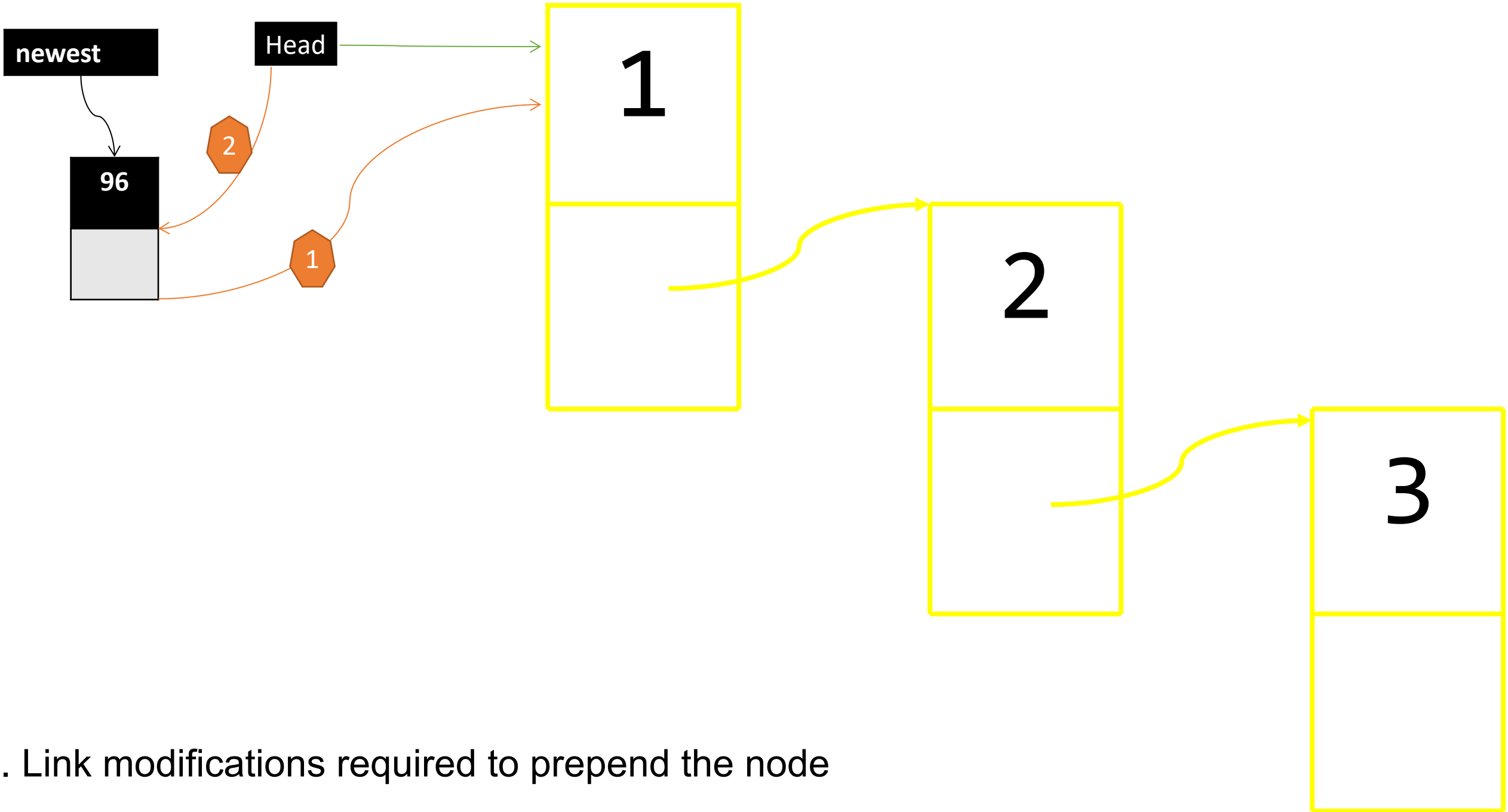
When working with an unordered list, new values can be inserted at any point within the list.

a. Inserting an element at the head of the singly Linked List
b. Inserting an element at the tail of the singly Linked List

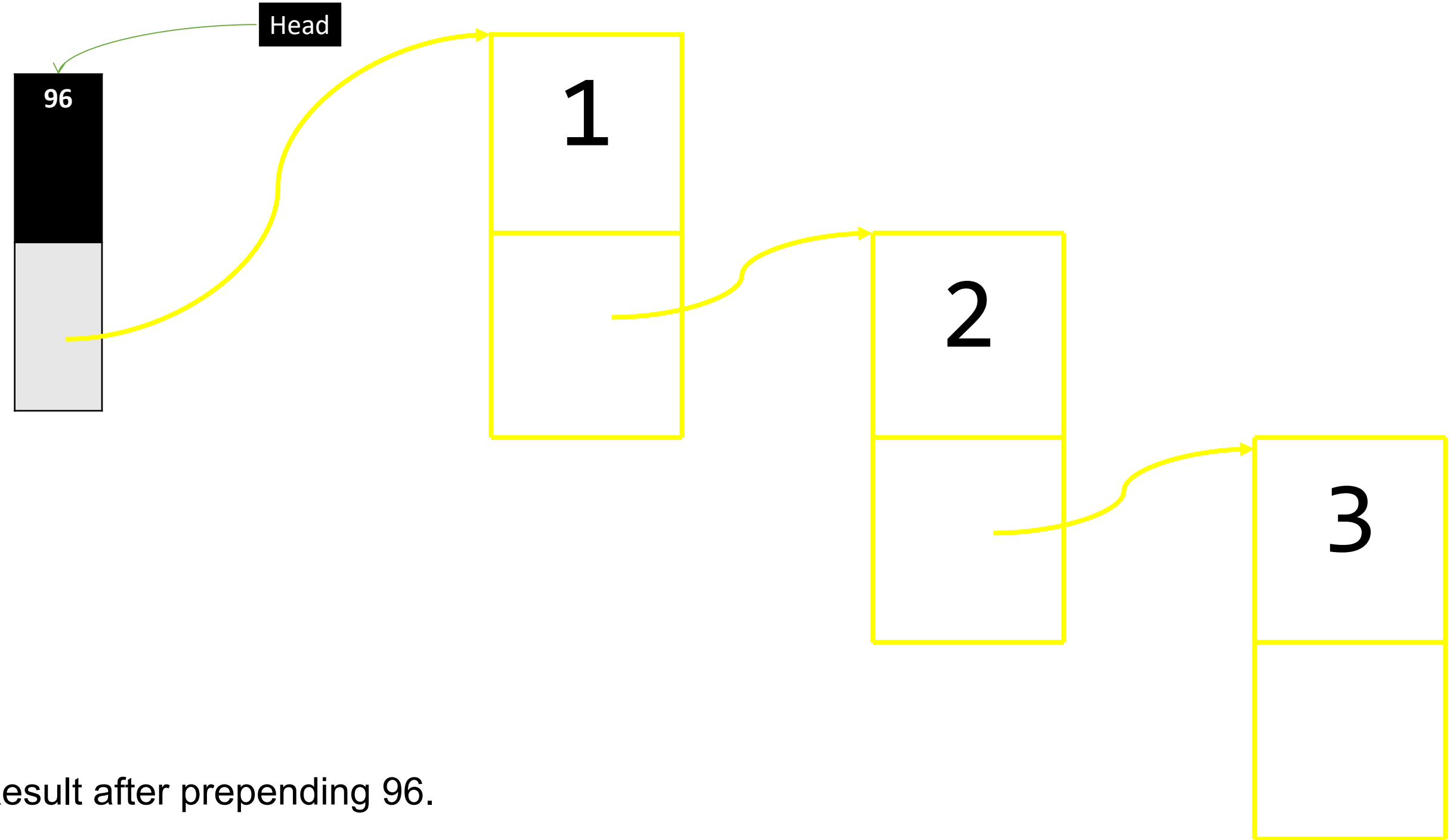
Operations on Linked List:

3. Perpending Nodes

a. Inserting an element at the head of the singly Linked List



i. Link modifications required to prepend the node



ii. Result after prepending 96.

Algorithms ?

a. Inserting an element at the head of the singly Linked List

addFirst.java UNREGISTERED

addFirst.java

```
1  Algorithm addFirst(e):
2      newest=Node(e) /*create new node instance
3                     storing reference to element e */
4      newest.next = head /*set new node's next
5                     to reference the old head node */
6      head = newest /*set variable head to reference the new node */
7      size = size + 1 /* increment the node count */
```

Line 4, Column 26 Tab Size: 4 Java

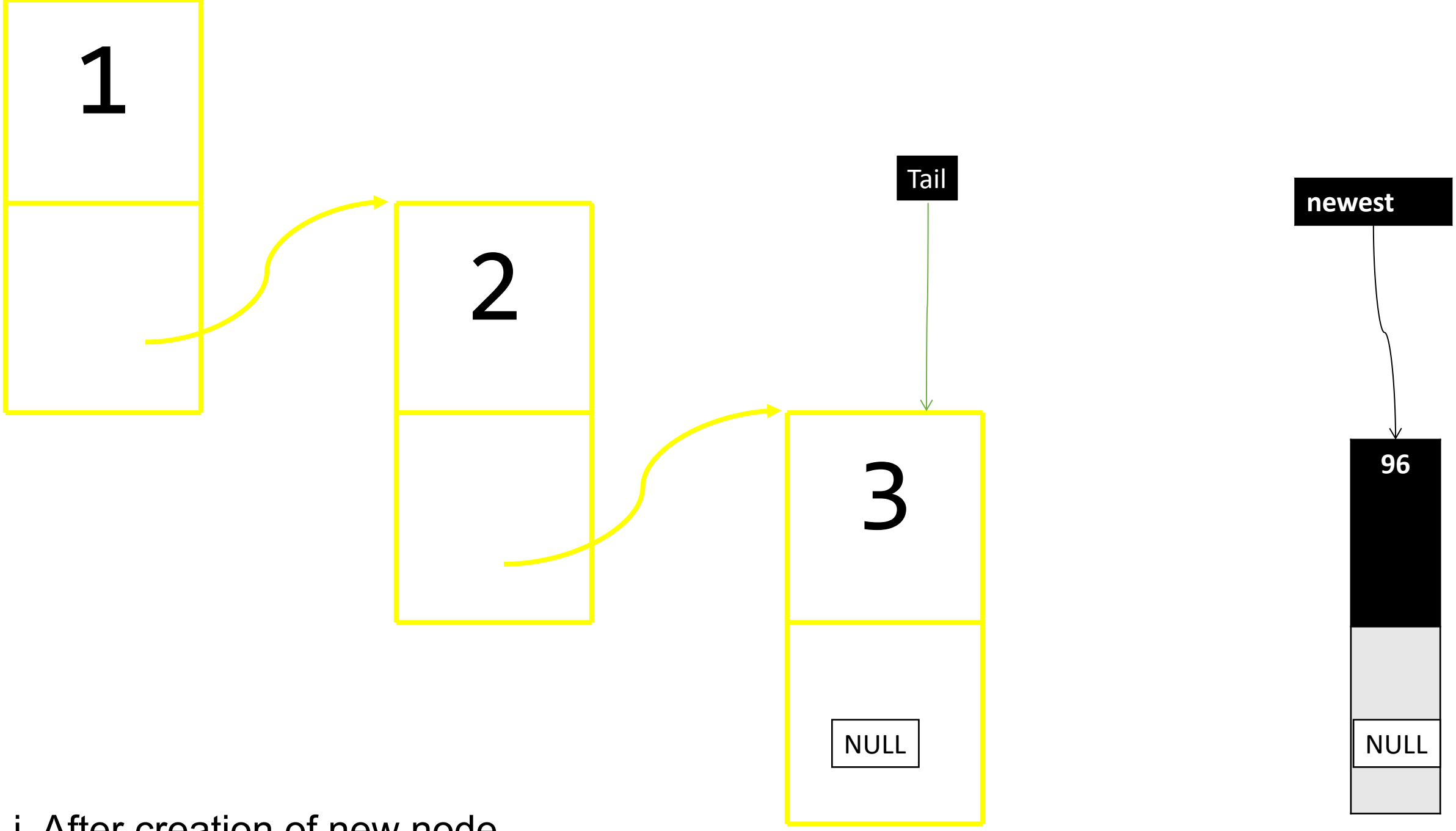
Time Complexity???

O(1)

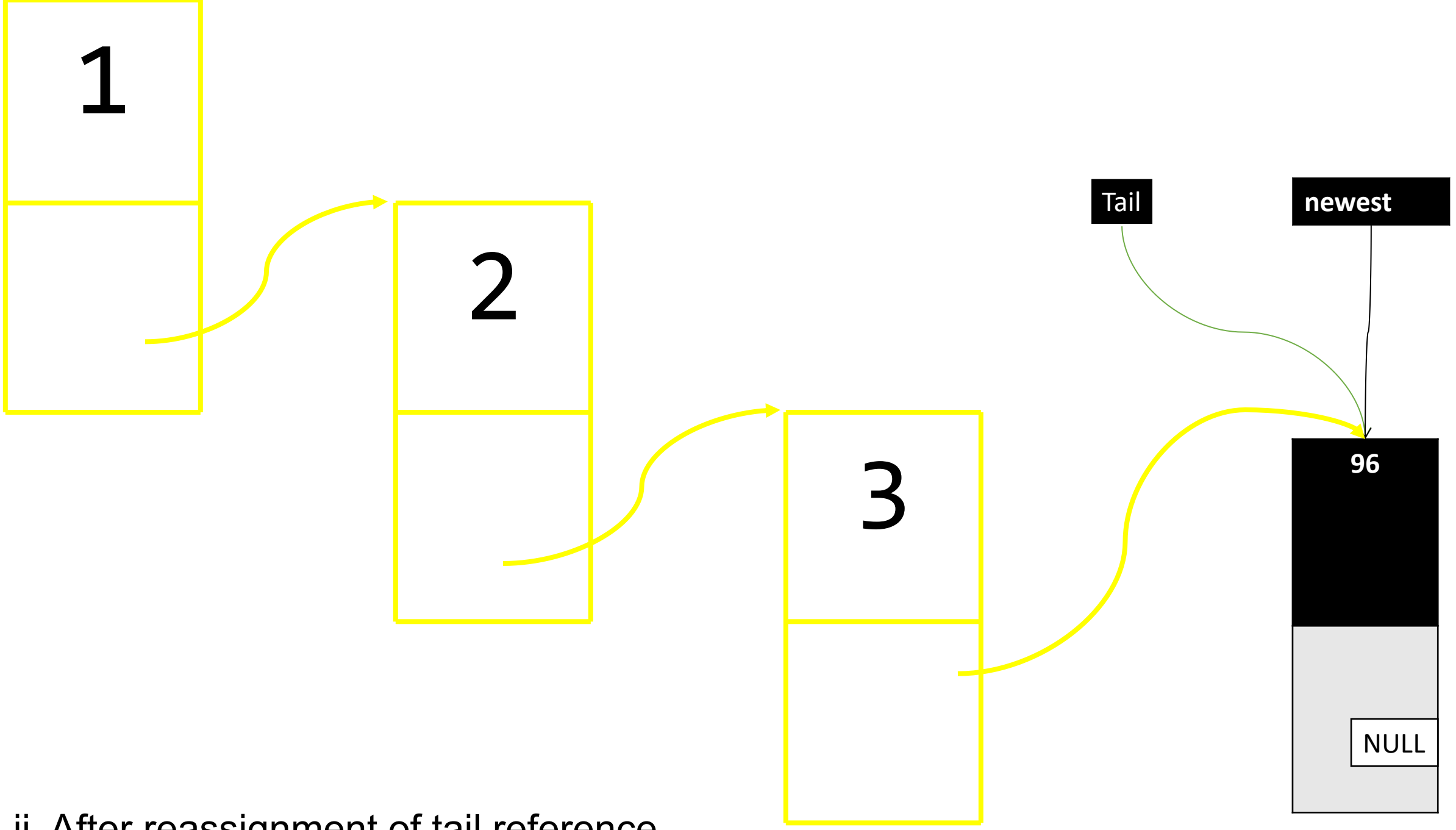
Operations on Linked List:

3. Perpending Nodes

b. Inserting an element at the tail of the singly Linked List



i. After creation of new node



ii. After reassignment of tail reference

Algorithms ?

b. Inserting an element at the tail of the singly Linked List

addLast.java

```
1  Algorithm addLast(e):
2      newest=Node(e) /*create new node instance
3                      storing reference to element e */
4      newest.next = null /* set new node's next
5                          to reference the null object */
6      tail.next = newest /* make old tail node
7                           point to new node */
8      tail = newest    /* set variable tail to
9                       reference the new node */
10     size = size + 1 // increment the node count
11
```

Line 10, Column 24 Tab Size: 4 Java

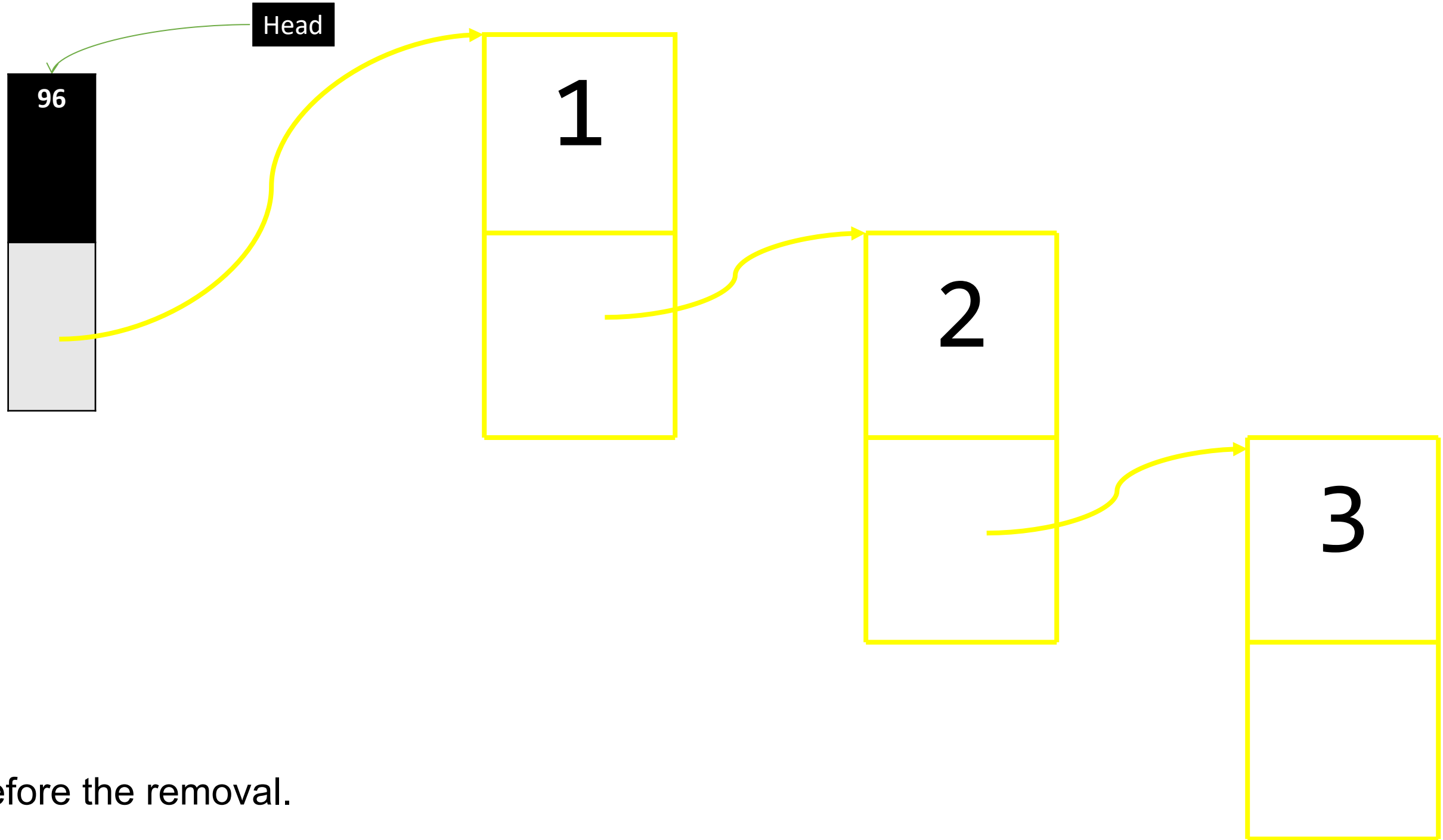
Time Complexity???

O(1)

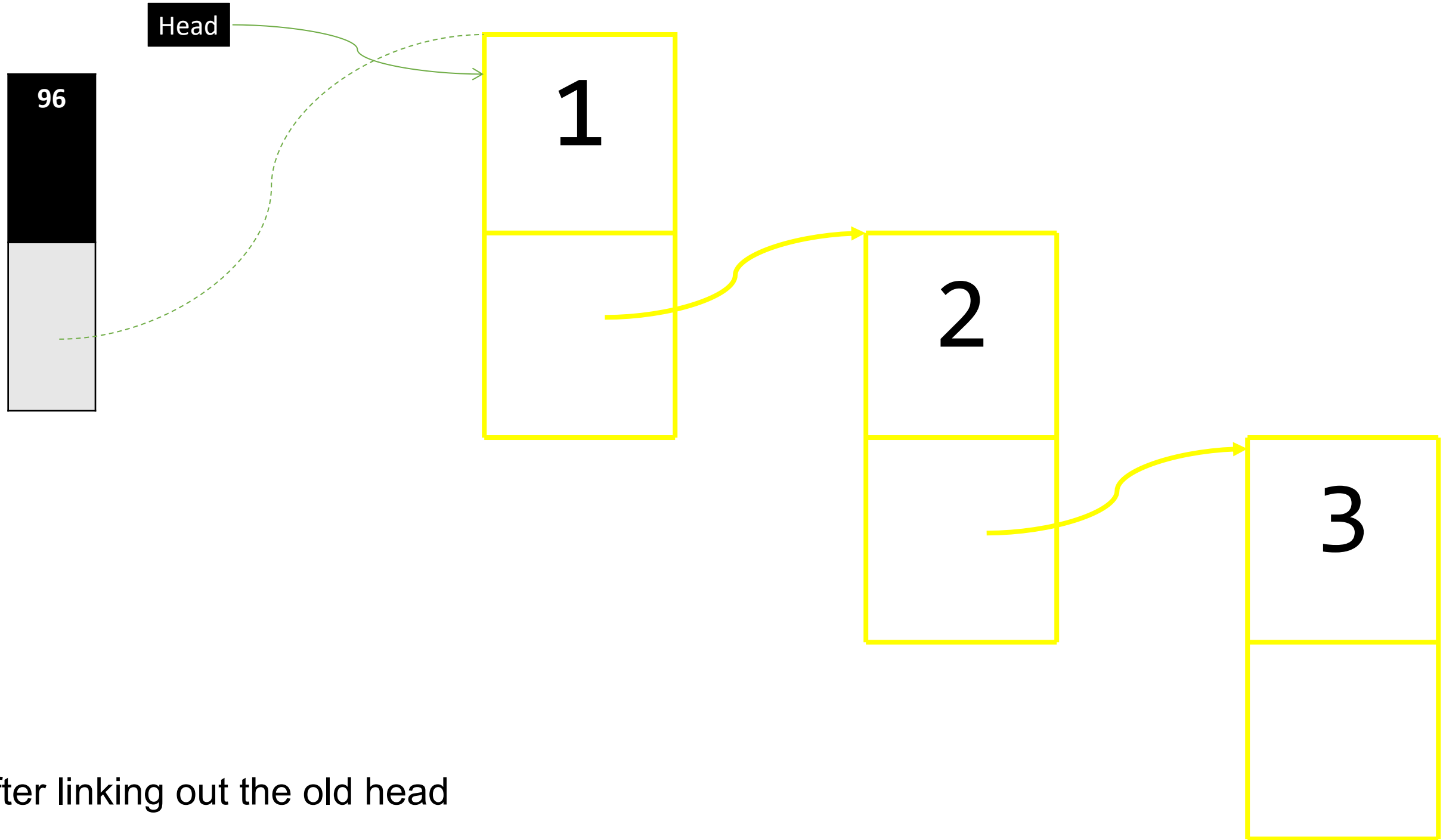
Operations on Linked List:

3. Removing Nodes

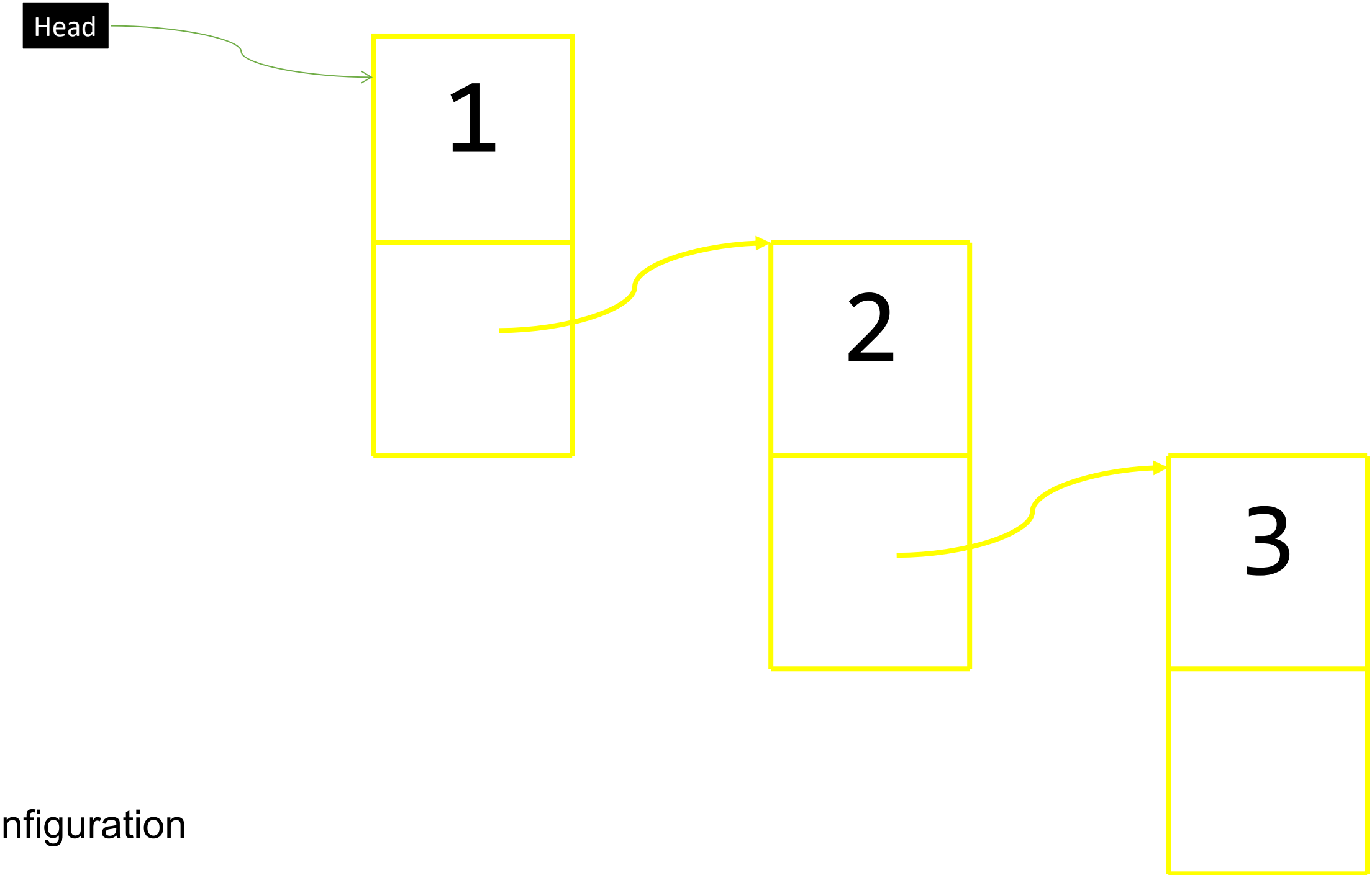
- An item can be removed from a linked list by removing or unlinking the node containing that item.
- Removing an element from the head of a singly linked list is essentially the reverse operation of inserting a new element at the head.



i. Before the removal.



ii. After linking out the old head



iii. Final COnfiguration

Algorithms ?

removeFirst.java

```
1  Algorithm removeFirst():
2      if head == null then
3          the list is empty.
4      head = head.next /*make head point to next node (or null) */
5      size = size - 1 /* decrement the node count */
```

Line 5, Column 24 Tab Size: 4 Java

Time Complexity???

O(1)

Operations on Linked List:

3. Removing Nodes

CAN WE REMOVE A NODE FROM THE TAIL OF THE LINKED LIST???

Implementing a Singly Linked List Class

size(): Returns the number of elements in the list.
isEmpty(): Returns true if the list is empty, and false otherwise.
first(): Returns(but does not remove) the first element in the list.
last(): Returns (but does not remove) the last element in the list.
addFirst(e): Adds a new element to the front of the list.
addLast(e): Adds a new element to the end of the list.
removeFirst(): Removes and returns the first element of the list.

Note: If first(), last(), or removeFirst() are called on a list that is empty, we will simply return a null reference and leave the list unchanged.