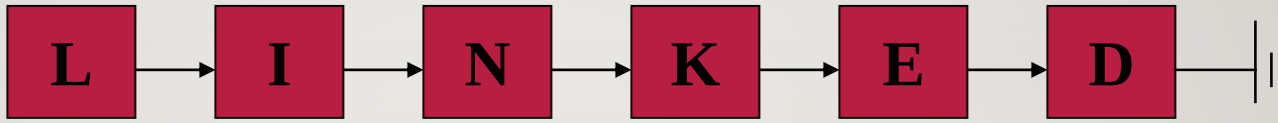


LINKED LIST VARIATIONS

NATIONAL UNIVERSITY OF TECHNOLOGY (NUTECH)

DR. SAMAN RIAZ

LECTURE # 9

ROADMAP

- List as an ADT
- An Array-Based Implementation of Lists
- Introduction to Linked Lists
- A Pointer-Based Implementation in C++
- Variations of Linked-lists

LINKED LISTS - ADVANTAGES

- Access any item as long as external link to first item maintained
- Insert new item without shifting
- Delete existing item without shifting
- Can expand/contract (flexibile) as necessary

LINKED LISTS - DISADVANTAGES

- Overhead of links:
  - used only internally, pure overhead
- If dynamic, must provide
  - destructor
  - copy constructor
  - assignment operator
- No longer have direct access to each element of the list
  - Many sorting algorithms need direct access
  - Binary search needs direct access
- Access of n^{th} item now less efficient
  - must go through first element, then second, and then third, etc.

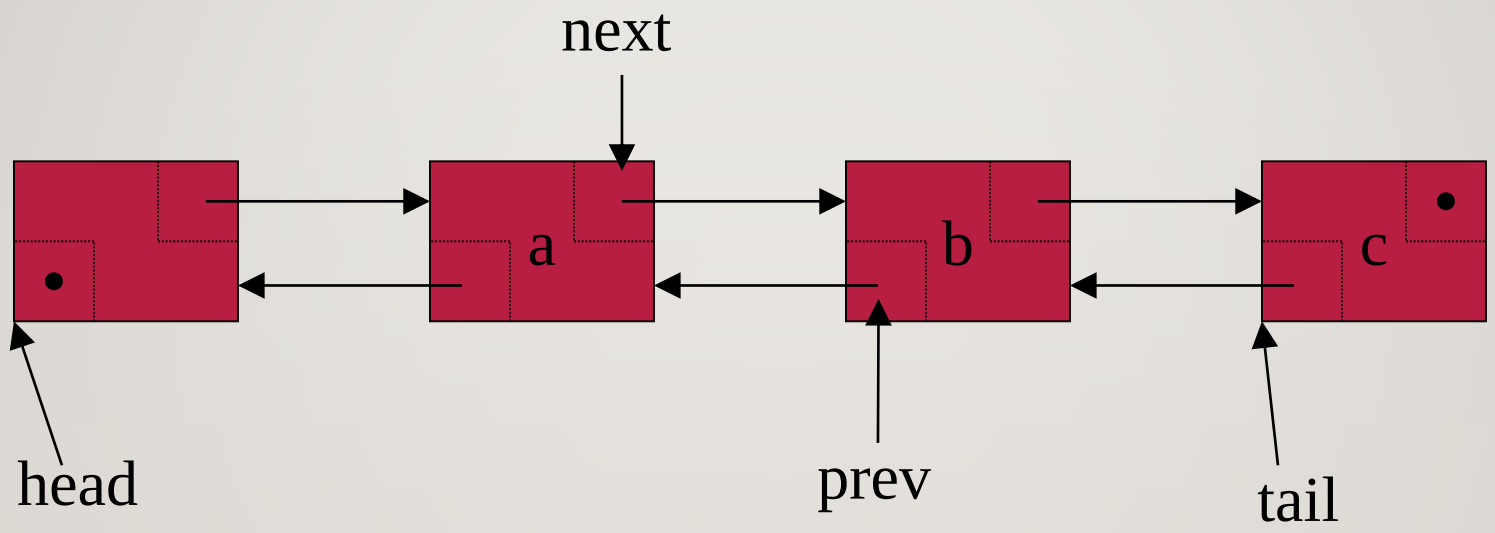
SOME APPLICATIONS?

- A linked list would be a reasonably good choice for implementing any of the following:

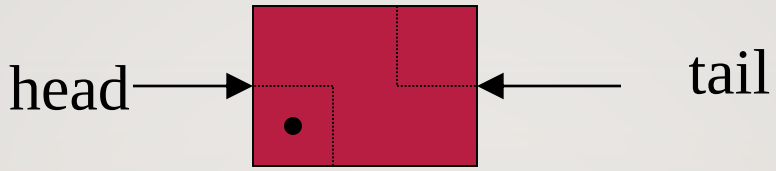
1. Applications that have an MRU list (a linked list of file names)
2. The cache in your browser that allows you to hit the BACK button (a linked list of URLs)
3. Undo functionality in Photoshop or Word (a linked list of state)
4. A list in the GPS of the turns along your route

Can we go back in current implementation?

DOUBLY LINKED LISTS

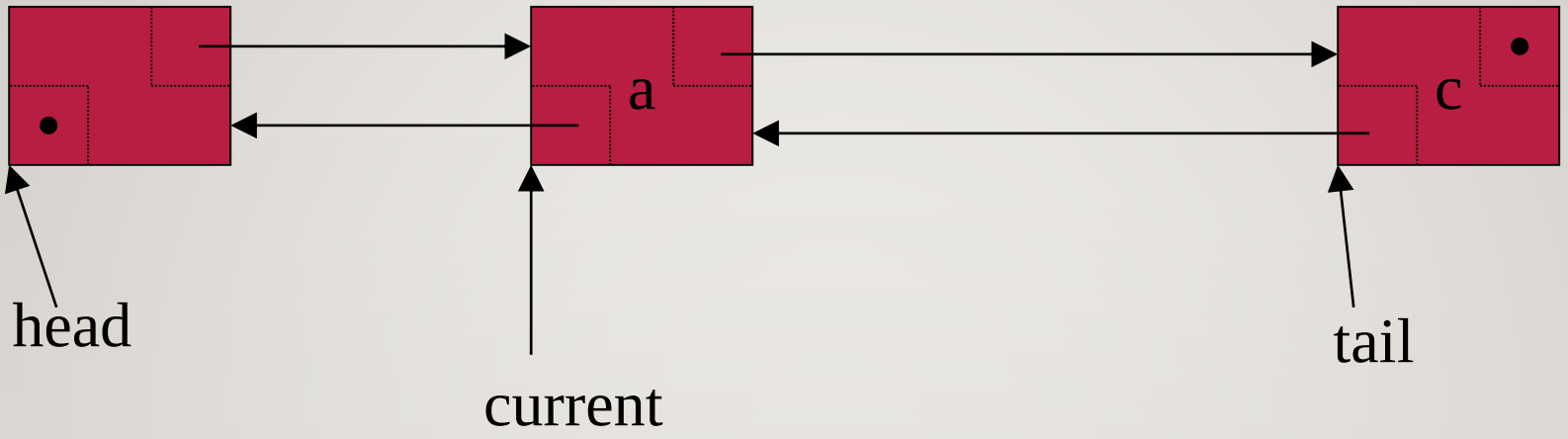


Consider how hard it is to back up in a singly linked list.



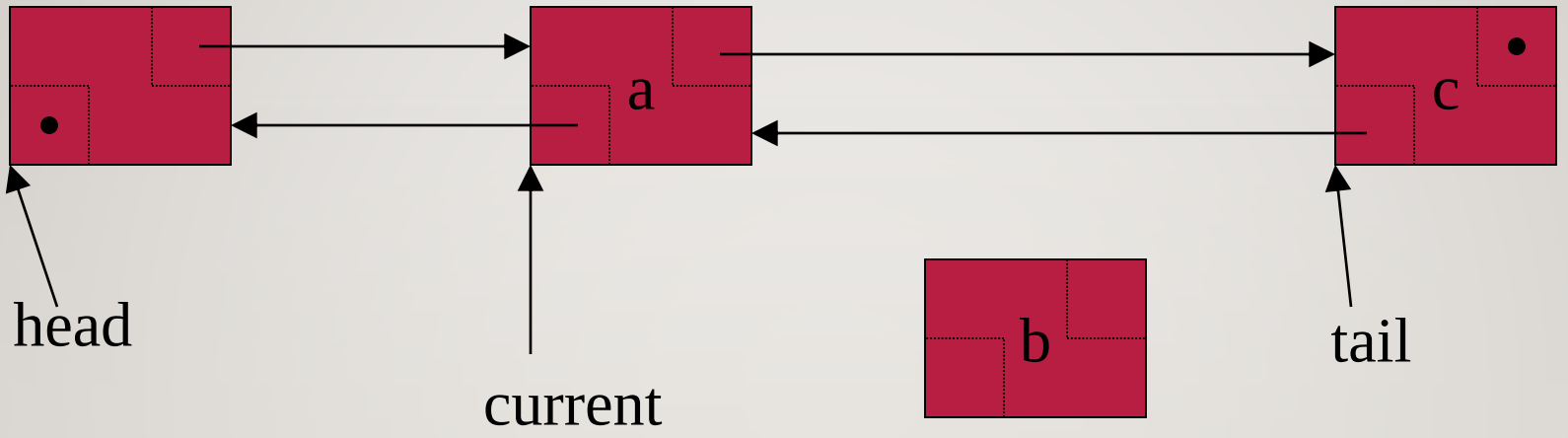

```
// Adding first node
head = new DoubleListNode;
head->next = null;
head->prev = null;
tail = head;
```

INSERTING INTO A DOUBLY LINKED LIST



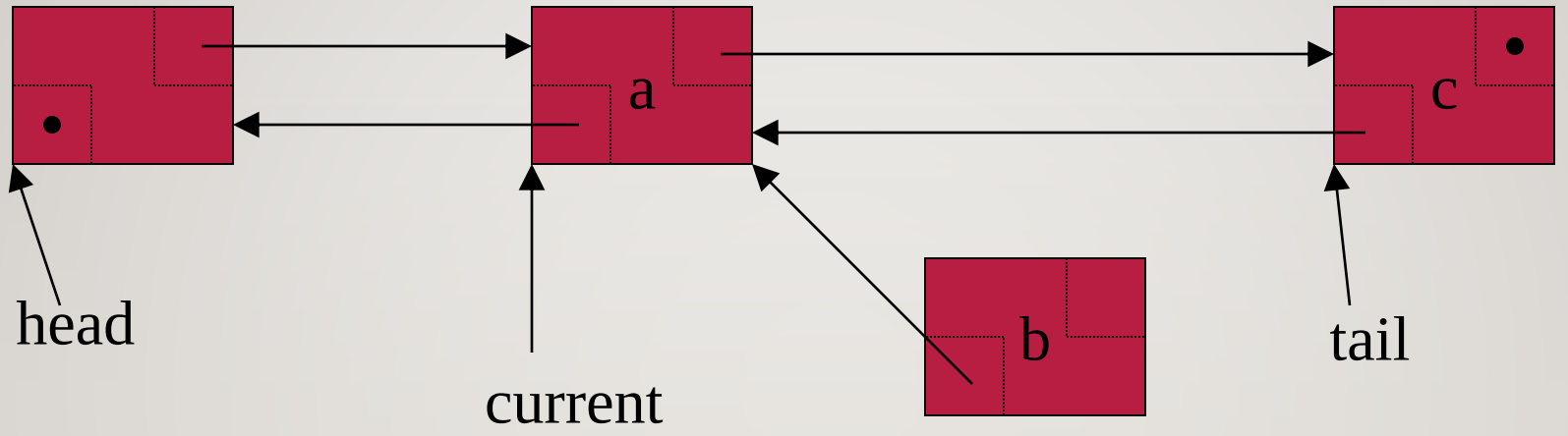

```
newNode = new DoublyLinkedListNode
newNode->prev = current;
newNode->next = current->next;
newNode->prev->next = newNode;
newNode->next->prev = newNode;
current = newNode
```

INSERTING INTO A DOUBLY LINKED LIST



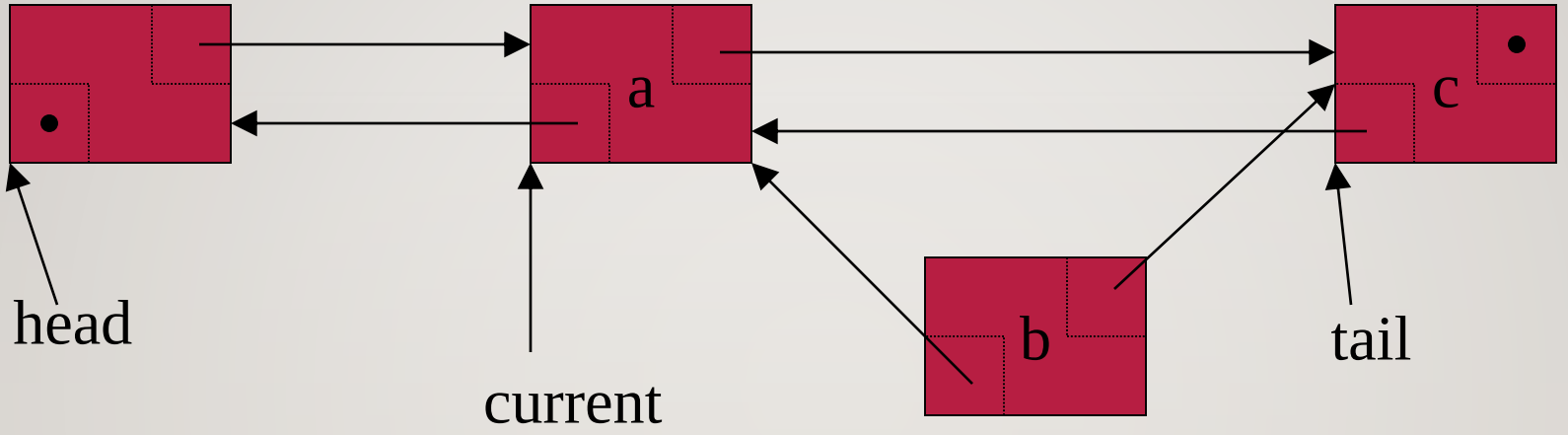

```
newNode = new DoublyLinkedListNode
newNode->prev = current;
newNode->next = current->next;
newNode->prev->next = newNode;
newNode->next->prev = newNode;
current = newNode
```

INSERTING INTO A DOUBLY LINKED LIST



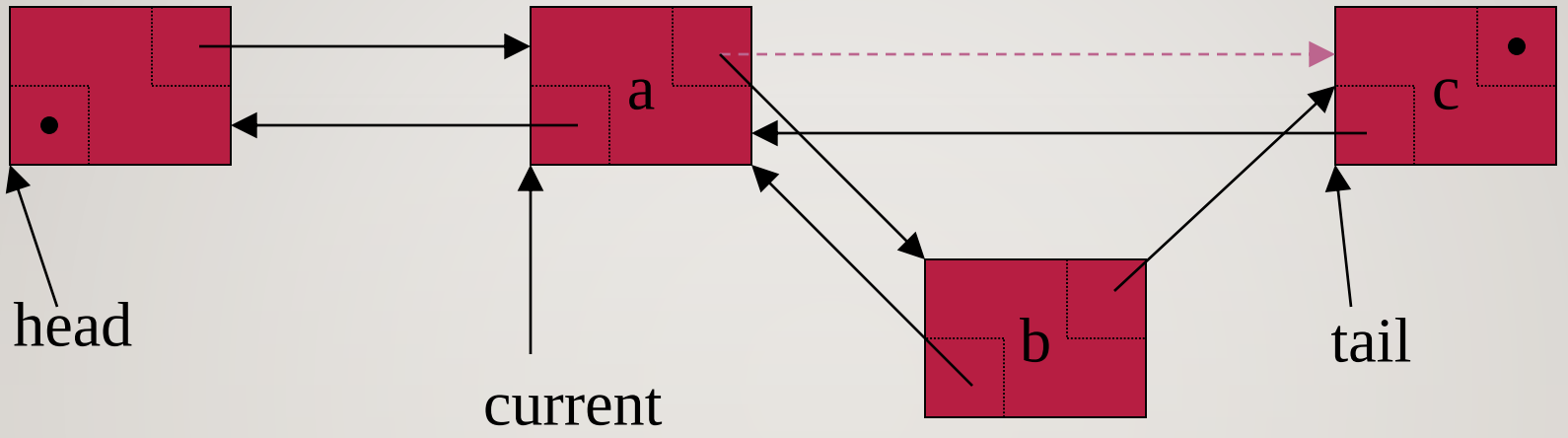

```
newNode = new DoublyLinkedListNode
newNode->prev = current;
newNode->next = current->next;
newNode->prev->next = newNode;
newNode->next->prev = newNode;
current = newNode
```

INSERTING INTO A DOUBLY LINKED LIST



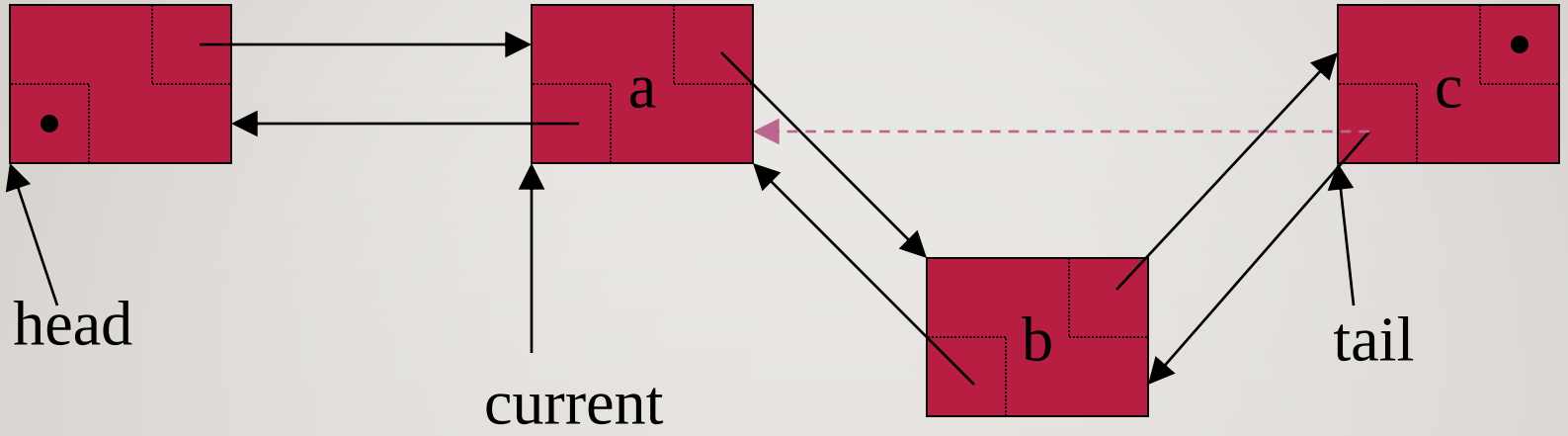

```
newNode = new DoublyLinkedListNode
newNode->prev = current;
newNode->next = current->next;
newNode->prev->next = newNode;
newNode->next->prev = newNode;
current = newNode
```

INSERTING INTO A DOUBLY LINKED LIST



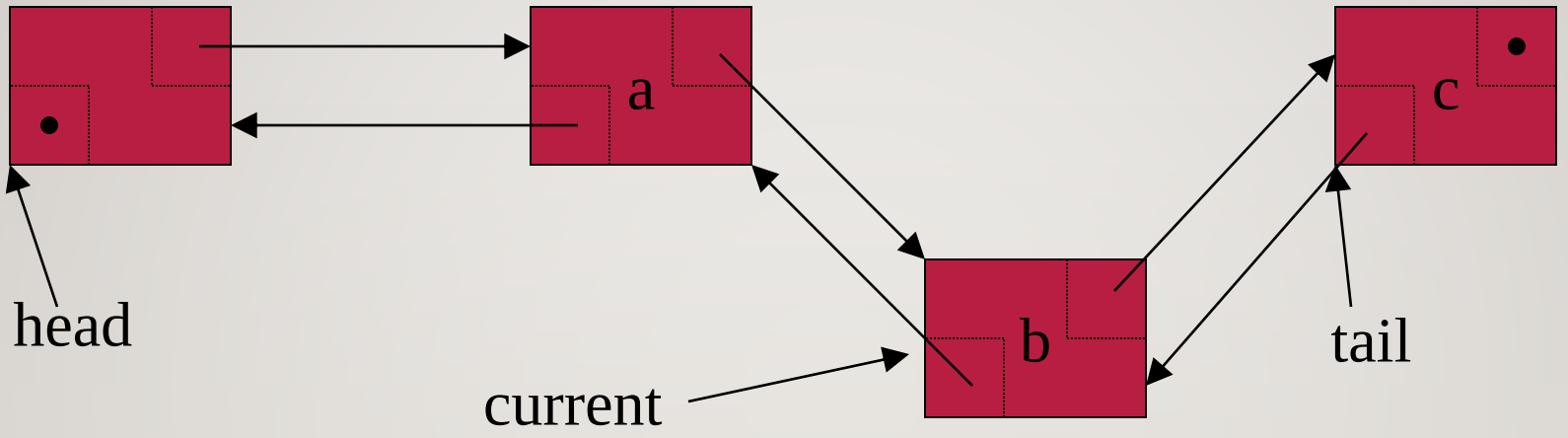

```
newNode = new DoublyLinkedListNode
newNode->prev = current;
newNode->next = current->next;
newNode->prev->next = newNode;// current->next=newNode;
newNode->next->prev = newNode;
current = newNode
```

INSERTING INTO A DOUBLY LINKED LIST



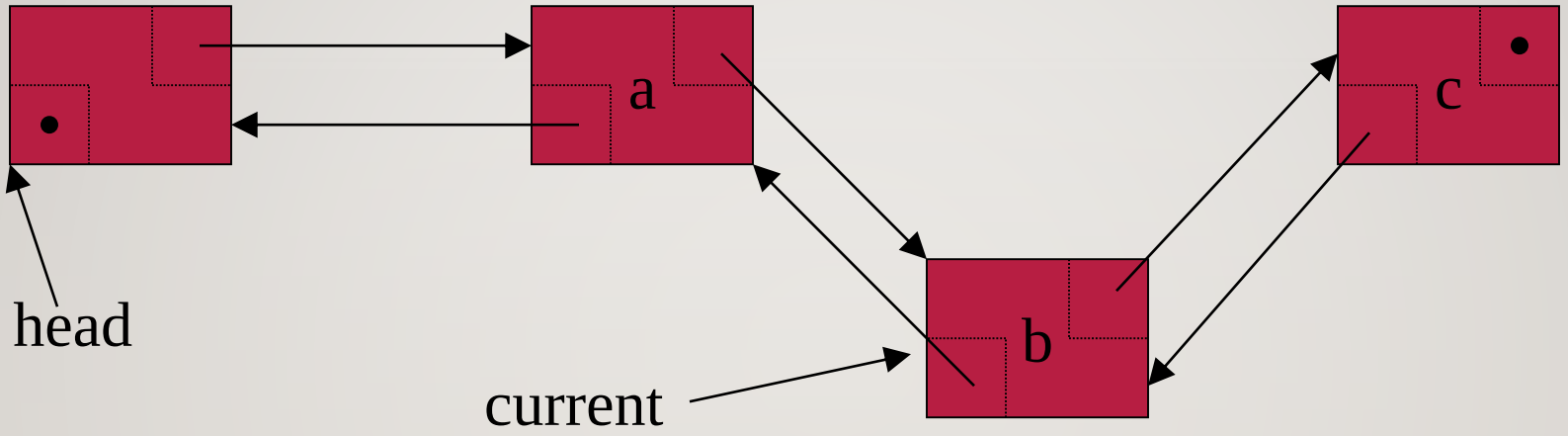

```
newNode = new DoublyLinkedListNode
newNode->prev = current;
newNode->next = current->next;
newNode->prev->next = newNode;
newNode->next->prev = newNode;
current = newNode
```

INSERTING INTO A DOUBLY LINKED LIST



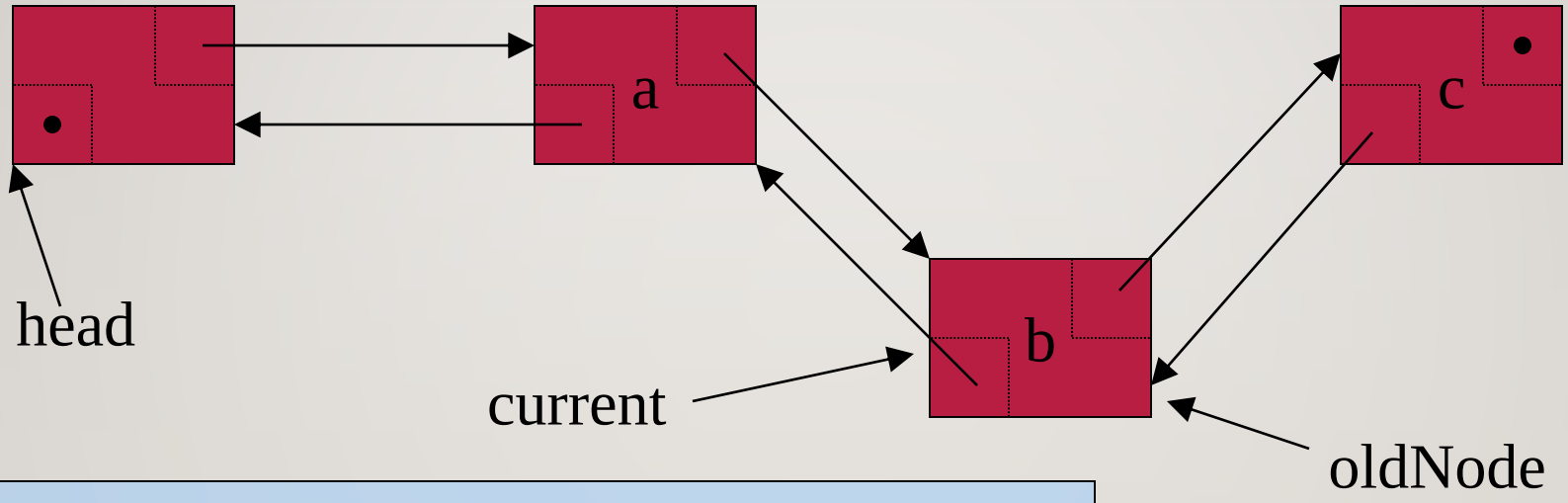

```
newNode = new DoublyLinkedListNode
newNode->prev = current;
newNode->next = current->next;
newNode->prev->next = newNode;
newNode->next->prev = newNode;
current = newNode
```

DELETING AN ELEMENT FROM A DOUBLE LINKED LIST



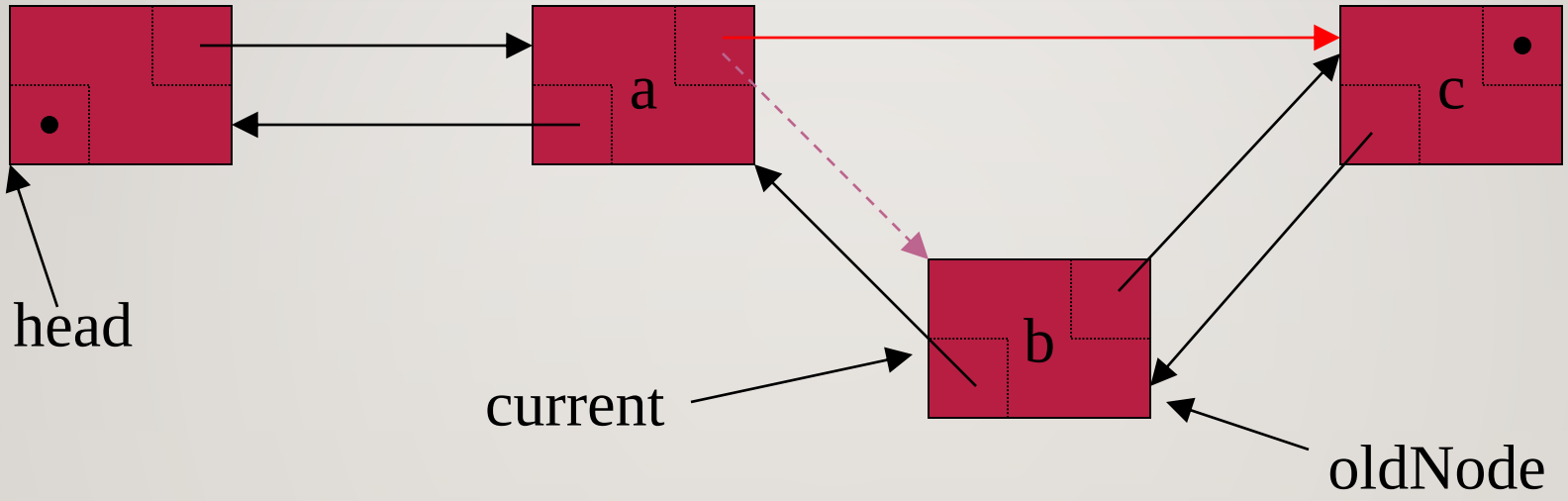

```
oldNode=current;
oldNode->prev->next = oldNode->next;
oldNode->next->prev = oldNode->prev;
current = oldNode->prev;
delete oldNode;
```

DELETING AN ELEMENT FROM A DOUBLE LINKED LIST



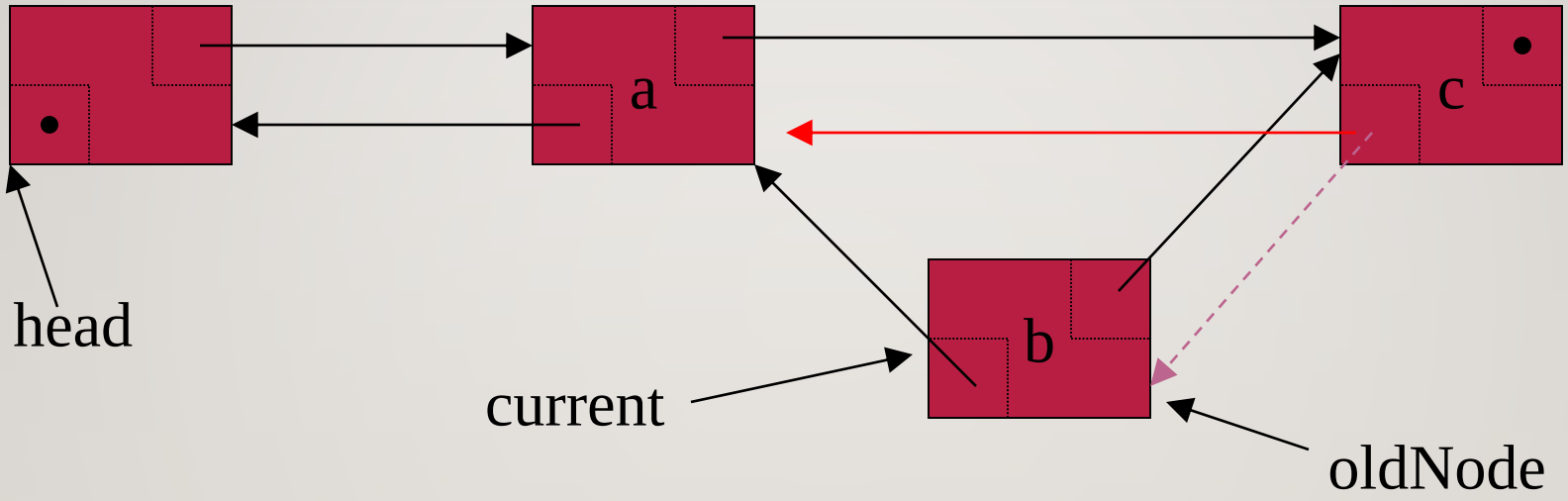

```
oldNode=current;
oldNode->prev->next = oldNode->next;
oldNode->next->prev = oldNode->prev;
current = oldNode->prev;
delete oldNode;
```

Deleting an element from a double linked list



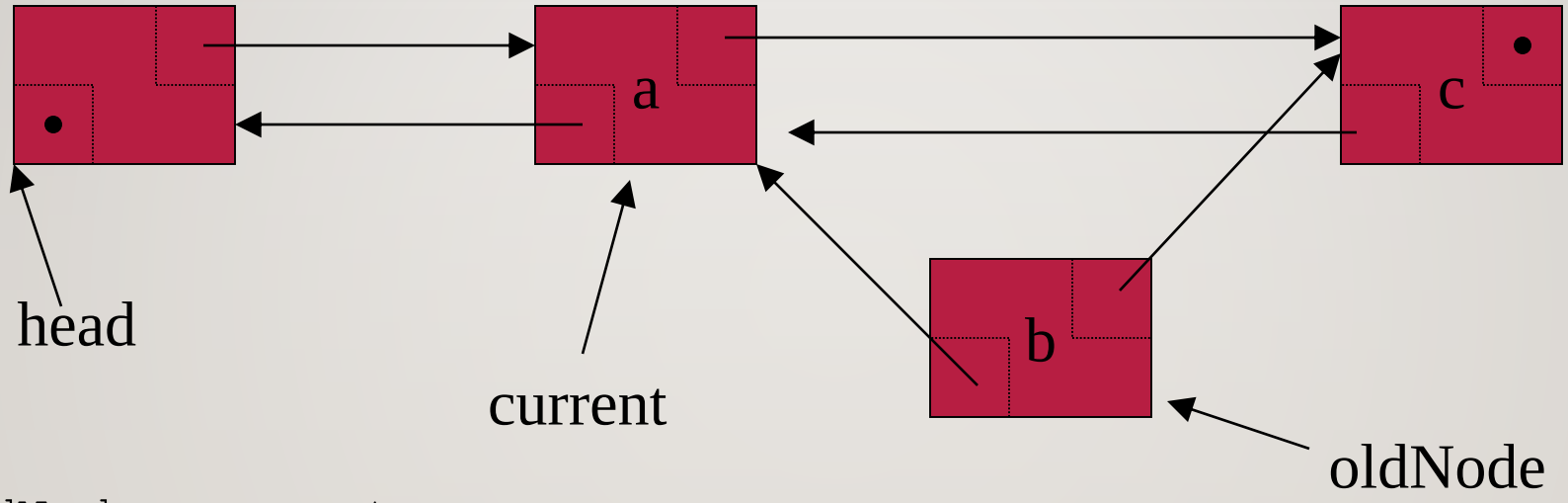

```
oldNode=current;
oldNode->prev->next = oldNode->next;
oldNode->next->prev = oldNode->prev;
current = oldNode->prev;
delete oldNode;
```

Deleting an element from a double linked list



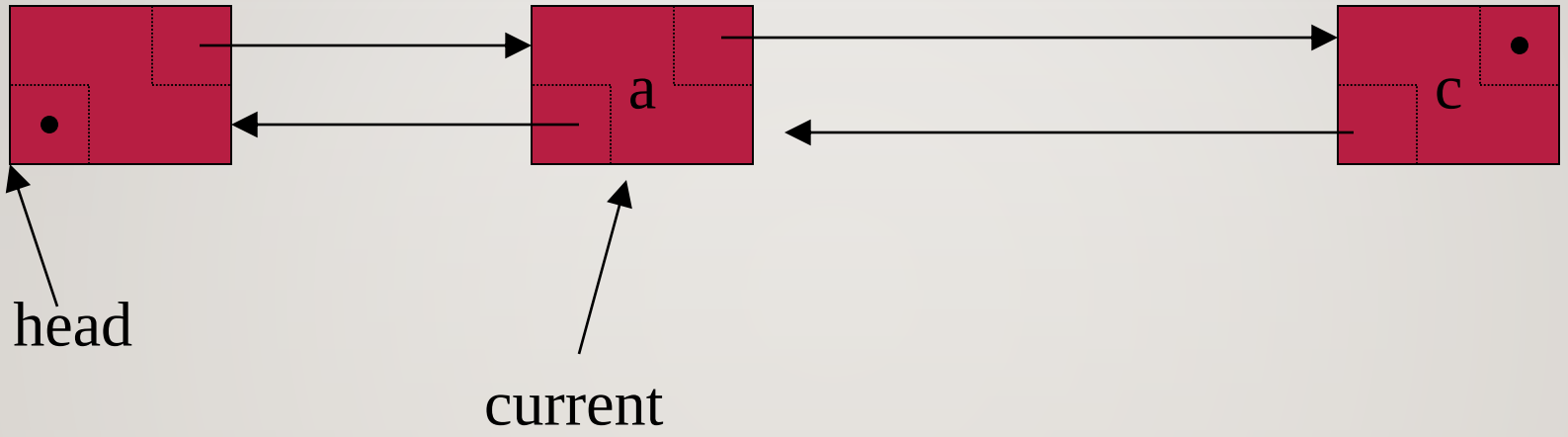

```
oldNode=current;
oldNode->prev->next = oldNode->next;
oldNode->next->prev = oldNode->prev;
current = oldNode->prev;
delete oldNode;
```

Deleting an element from a double linked list



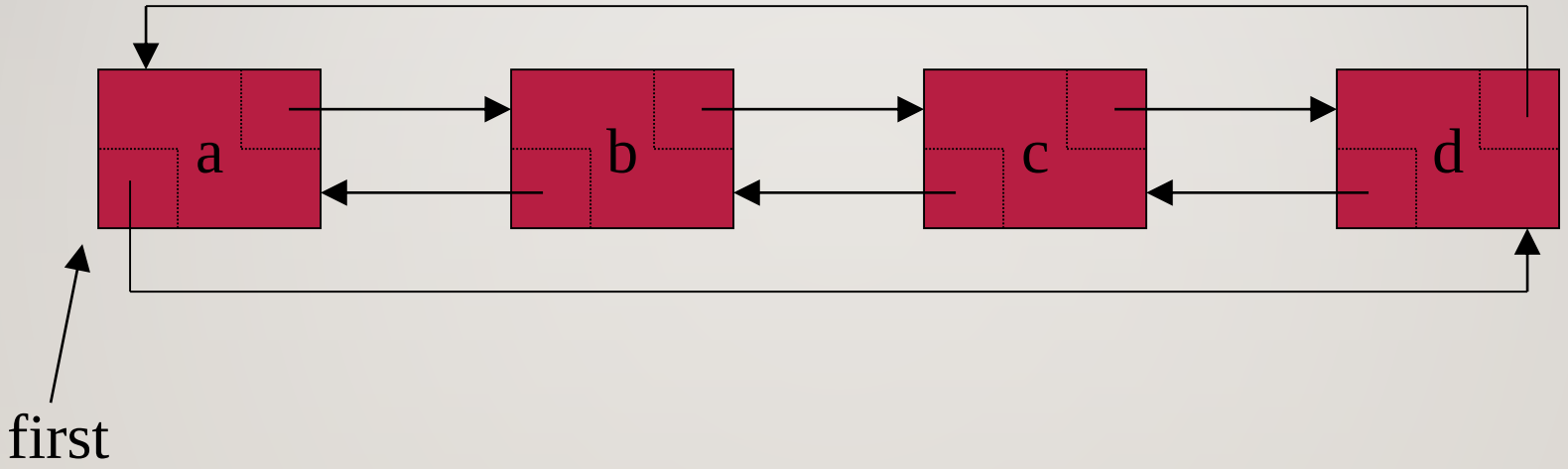

```
oldNode=current;
oldNode->prev->next = oldNode->next;
oldNode->next->prev = oldNode->prev;
current = oldNode->prev;
delete oldNode;
```

Deleting an element from a double linked list




```
oldNode=current;
oldNode->prev->next = oldNode->next;
oldNode->next->prev = oldNode->prev;
current = oldNode->prev;
delete oldNode;
```

CIRCULAR LINKED LISTS



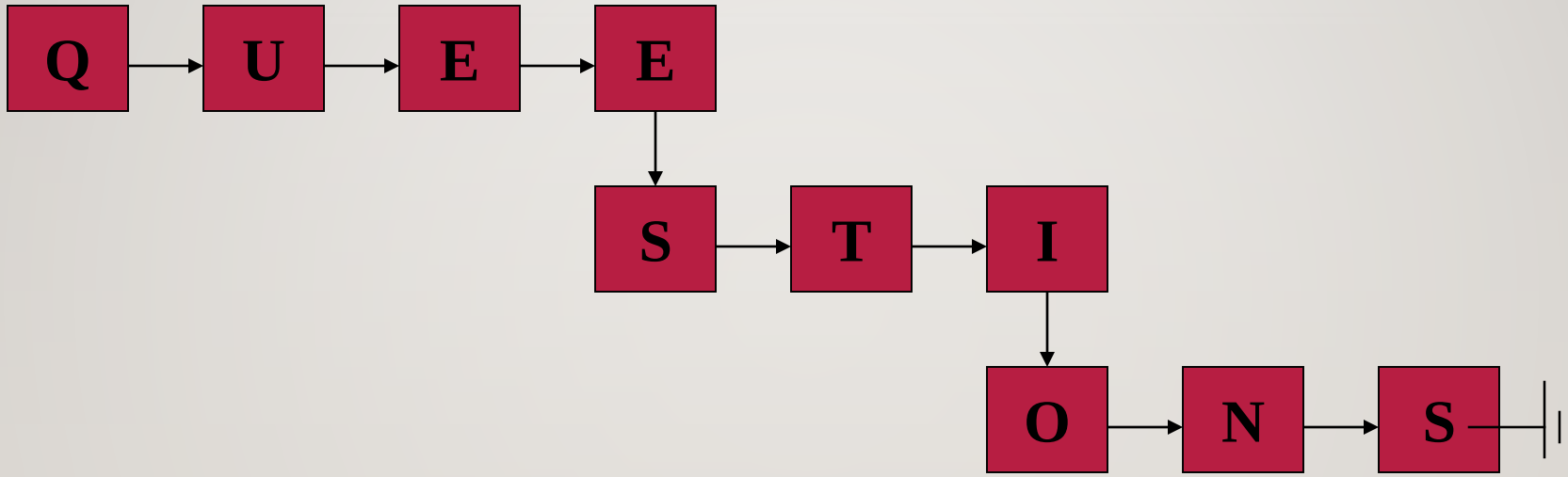
SORTED LINKED LIST

A sorted linked list is one in which items are in sorted order. It can be derived from a list class.

What is improved?

InsertNode operation? No

DeleteNode & SearchNode operations? Yes


Q
U
E
E
S
T
I
O
N
S