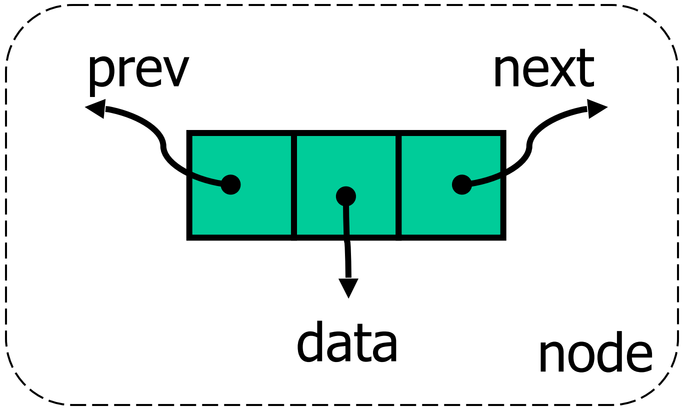
Doubly Linked Lists

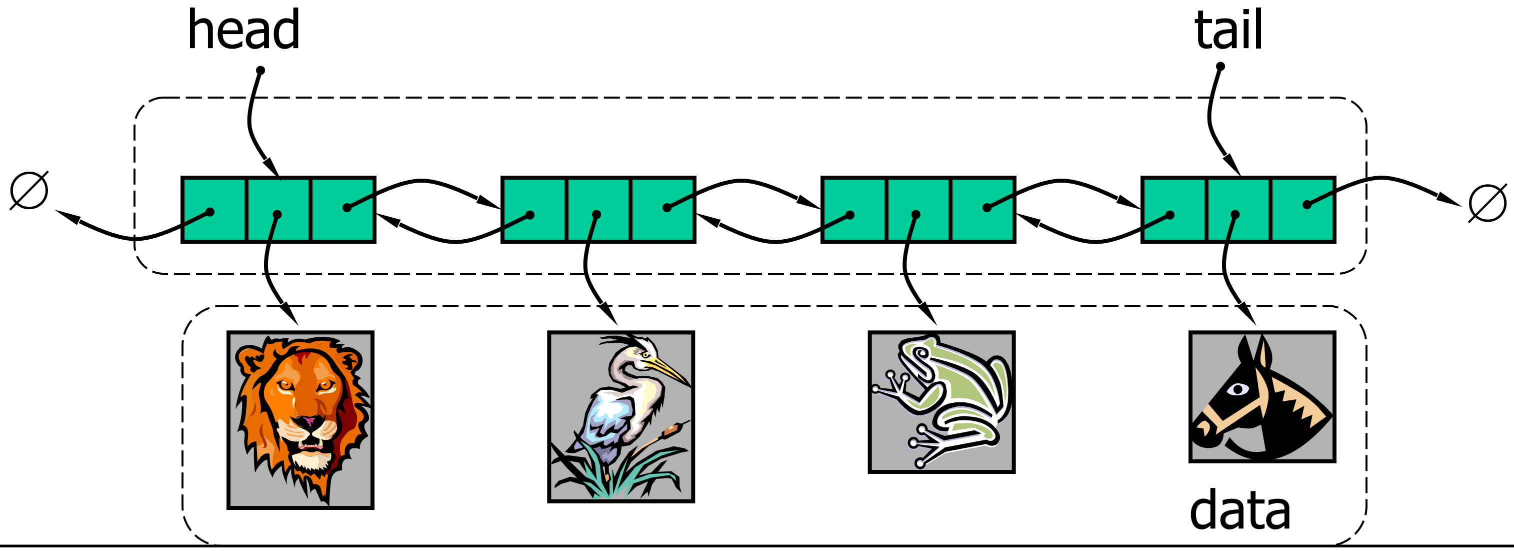
cs206

Lec 19

Doubly Linked List

- Can be traversed forward and backward
- Nodes store an extra reference





Double Linked List interface

```
public interface LinkedListInterface<E extends Comparable<E>> {
    int size();
    boolean isEmpty();
    Comparable<E> first();
    Comparable<E> last();
    void addLast(Comparable<E> c);
    void addFirst(Comparable<E> c);
    Comparable<E> removeFirst();
    Comparable<E> removeLast();
    Comparable<E> remove(Comparable<E> r);
    Comparable<E> find(Comparable<E> iD);
}
```

This is identical to the single linked list!!!

Node & DLL start

```
public class DoubleLinkedList<T extends Comparable<T>> implements
LinkedListInterface<T> {
    protected class Node<V extends Comparable<V>> {
       public Comparable<V> data;
       public Node<V> next;
       public Node<V> prev;
       public Node(Comparable<V> data, Node<V> prev, Node<V> next) {
            this.data = data;
            this.next = next;
            this.prev = prev;
        }
    }
   private Node<T> head = null;
   private Node<T> tail = null;
   private int size = 0;
```

Basics

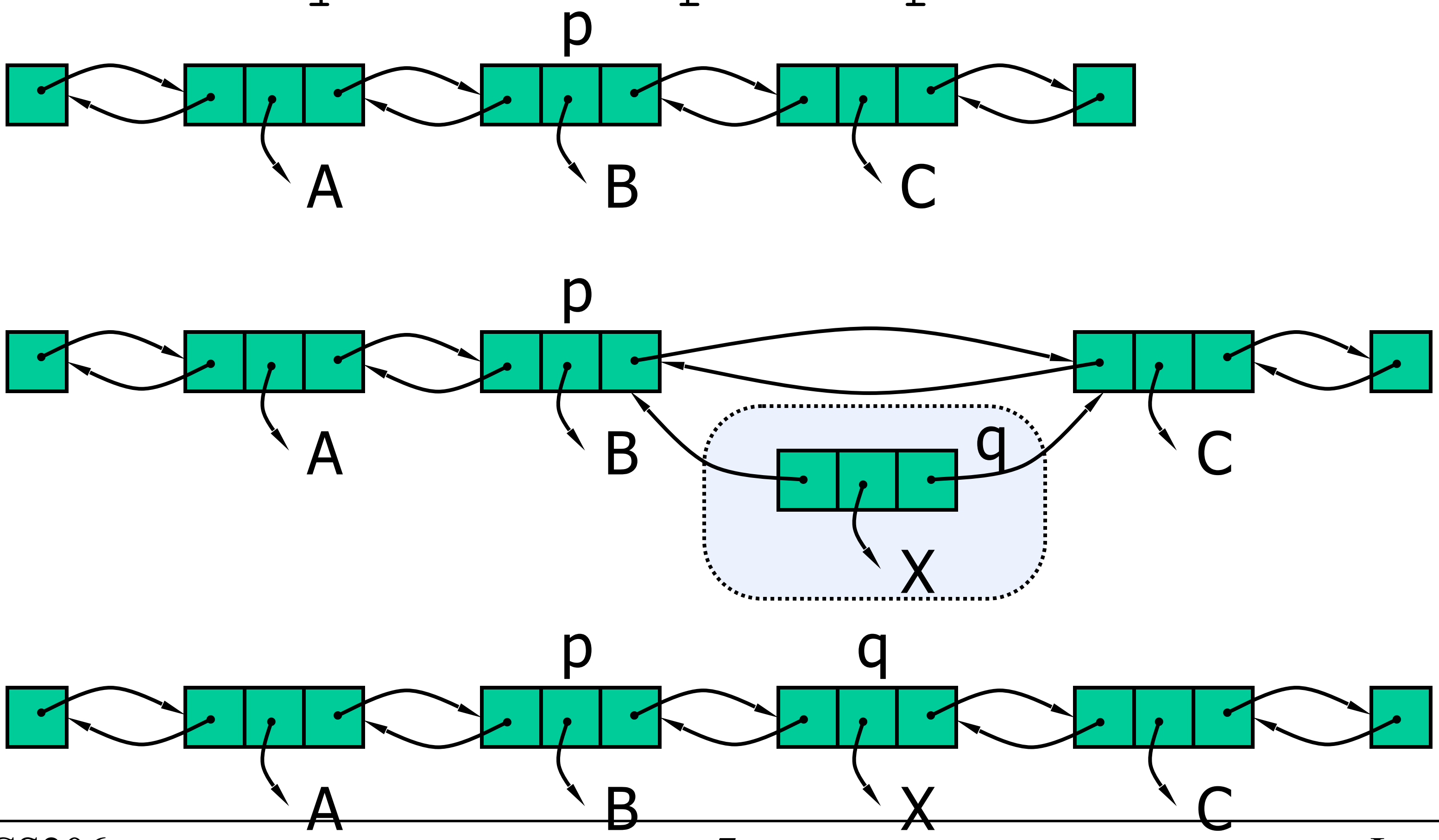
```
   @Override
   public int size() {
        return size;
    }
   @Override
    public boolean isEmpty() {
        return size == 0;
    }

   @Override
       public Comparable<T> first() {
        if (head == null)
            return null;
        return head.data;
    }
   @Override
    public Comparable<T> last() {
        if (head == null)
            return null;
        return tail.data;
    }
```

Insertion: AddFirst, AddLast

Add Between

- Insert `q` between `p` and `p.next`



Add Between

```
private void addBtw(T c, Node prev, Node next) {
    Node newest = new Node(c, prev, next);
    prev.next = newest;
    next.prev = newest;
    size++;
}
```

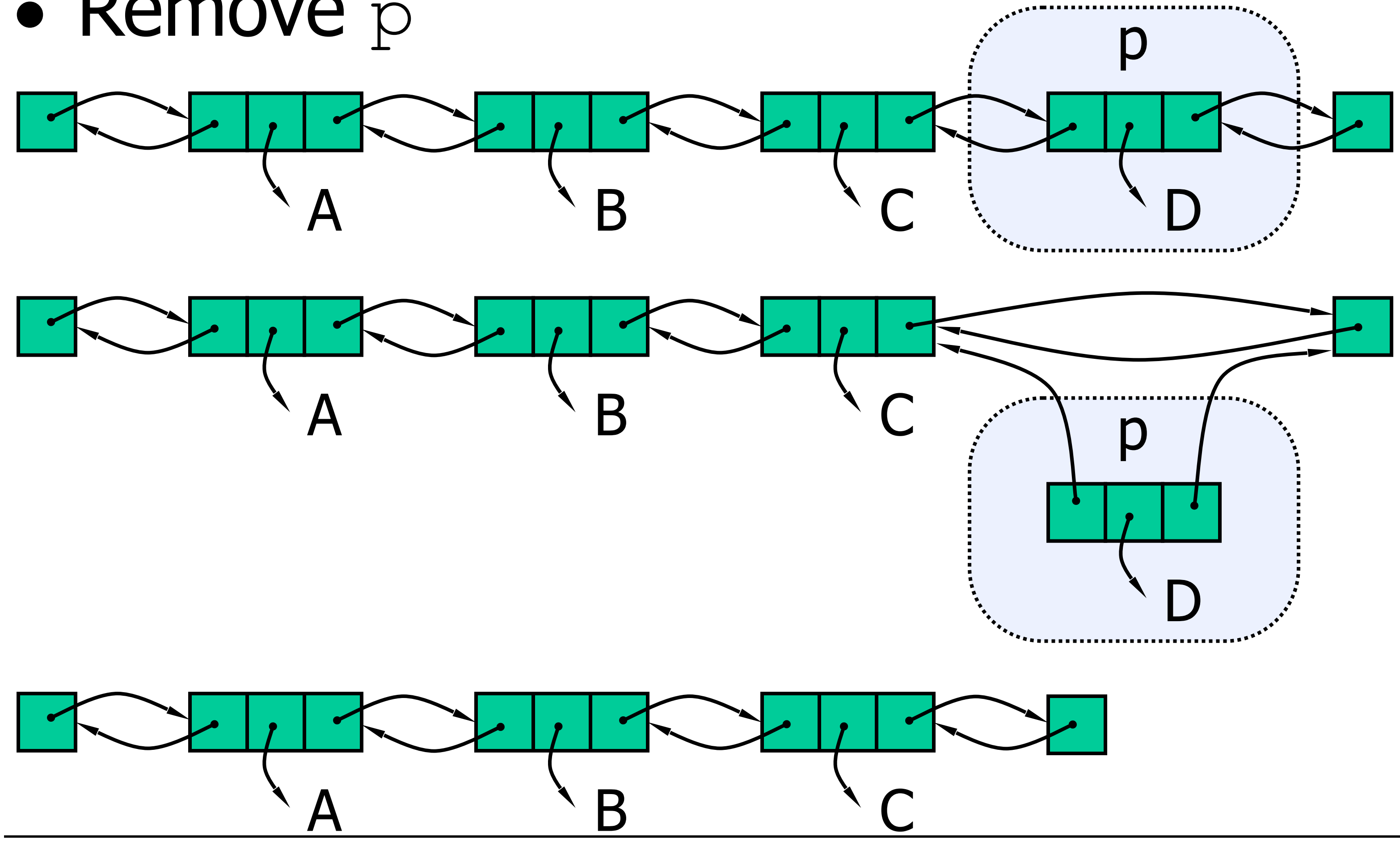
Problems??

Deletion — first element

```
    @Override
    @SuppressWarnings("unchecked")
    public T removeFirst() {
        if (head == null)
            return null;
        Comparable<T> rtn = head.data;
        head = head.next;
        if (head == null)
            tail = null;
        else
            head.prev = null;
        size--;
        return (T) rtn;
    }
```

Deletion

- Remove `p`



Deletion

```
@Override
    public T remove(T r) {
        // Do something much like find, but need to trak the previous node
        Node<T> curr = head;
        while (curr != null) {
            if (0 == curr.data.compareTo(r)) {
                break;
            }
            curr = curr.next;
        }
        if (curr == null) {
            // 1. the data item was not found
            return null;
        }
        size–;
        T ret = curr.data;
        if (curr.prev != null)
            curr.prev.next = curr.next;
        if (curr.next != null)
            curr.next.prev = curr.prev;
        if (curr == tail)
            tail = curr.prev;
        return ret;
    }
```

Sorted Linked Lists

```
public class SortedDLL<T extends Comparable<T>> extends
DoubleLinkedList<T> {
     public void addSorted(Comparable<T> t) {
      // lots of thought here
      }
```

Mini-LAB: What should be done with addFirst & addLast