EASTERN WASHINGTON UNIVERSITY
start something big

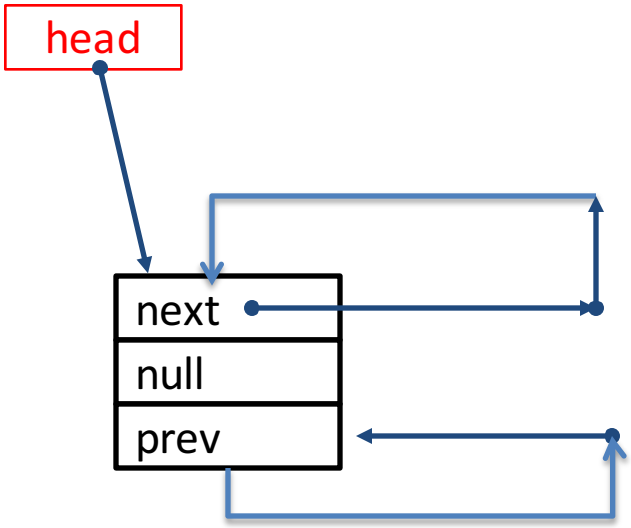
Sort Circular and Doubly Linked List

Computer Science Department
Eastern Washington University
Yun Tian (Tony) Ph.D.

Review

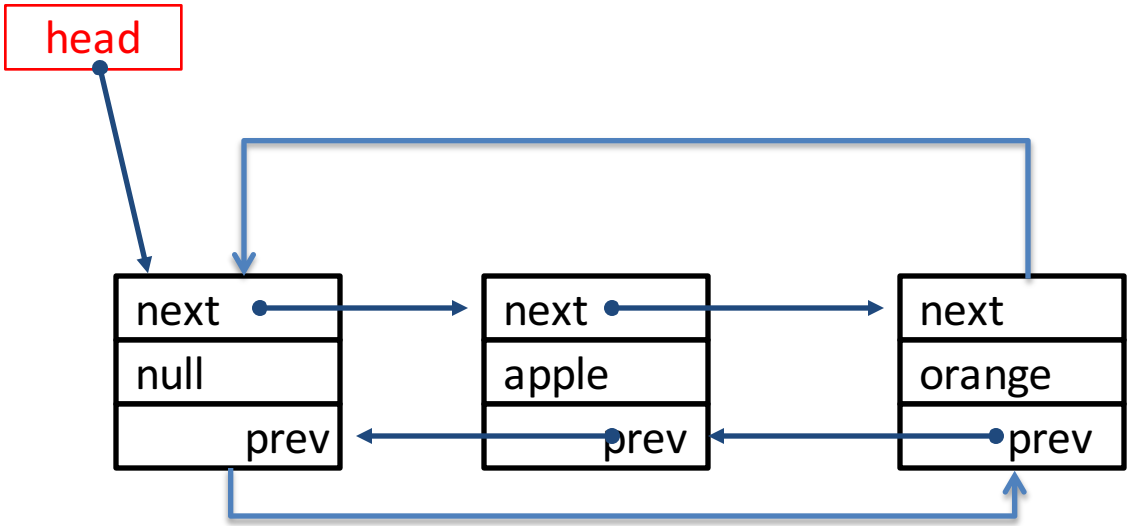
- Circular Doubly Linked List with dummy node
  - Concept and Implementation
  - add() and remove() method

Circular Doubly Linked List



Empty Doubly Linked List

Circular Doubly Linked List



1) In each node, we have three fields. **next** points the successor of current node. **prev** points to the predecessor of the current node. The last field is the data field.
2) This circular doubly linked list has a dummy node.
3) Could you imagine what an empty circular doubly linked list with a dummy node looks like?

Today

- addOrdered() and sort() in Circular Doubly Linked List

Circular Doubly Linked List

```
public class CDoublyLinkedList {
        private class Node {
                private Object data;  //comparable
                private Node next, prev;
                private Node(Object data, Node pref, Node next){
                        this.data = data;
                        this.prev = pref;
                        this.next = next;
                }
        }
        private Node head;
        private int size;
        public CDoublyLinkedList() {
                this.head = new Node(null, null, null );
                this.head.next = this.head;
                this.head.prev=this.head;
                this.size = 0;
        }
        //……
}
```

addOrdered() Method

EASTERN WASHINGTON UNIVERSITY
start something big

```
public void addOrdered( Comparable data ) {
        Node cur = this.head.next;
        while( ____cur != this.head___ && ((Comparable)cur.data).compareTo(data) < 0)
                ___cur = cur.next__;  //
                                        // insert before cur
        Node newNode = new Node( data, ___cur.prev___, __cur__ );
        __cur.prev.next__ = newNode;
        cur.prev = newNode;
        this.size ++;
}

//as we learned before, the order we do rewiring matters
//principles:
```

removeAll method

```
public boolean removeAll( Object data ) {
        boolean removed = remove(data);
        while ( remove(data) ) {
                //empty body
        }
        return removed;
}
public boolean removeAllFast(Object data) {
        boolean removed = false;
        Node cur = this.head.next;
        while ( cur != this.head ) {
                if(cur.data.equals(data)) {
                        cur.prev.next = cur.next ;
                        cur.next.prev = cur. prev ;
                        this.size --;
                        removed = true;
                }
                cur =cur.next;
        }
        return removed;
}
```

Insertion Sort On Array

EASTERN WASHINGTON UNIVERSITY
start something big

```
public void insertionSort( Comparable [] array ) {
   int lastSorted;
   int sortedWalker;
   for( lastSorted = 0; lastSorted < array.length - 1; lastSorted ++ ) {
         Comparable firstUnsortedData = array[lastSorted + 1];
         //now shift bigger values in sorted part over to make room for firstUnsortedData
         for(sortedWalker = lastSorted; sortedWalker >=0 &&
                  array[sortedWalker].compareTo(firstUnsortedData) > 0; sortedWalker --)
               array[sortedWalker + 1] = array[sortedWalker];
         //now insert
         array[sortedWalker + 1] = firstUnsortedData;
   }
}

//Insertion Sort is like organizing a handful of playing cards. You pick the first unsorted card at
//a time and insert it into its correct position.
//The whole array is divided into two subarrays, sorted part and unsorted part.
```

Insertion Sort On Circular Doubl


```
public void insertionSort() {
   Node lastSorted, sortedWalker;
   Comparable firstUnsortedData;
   for(lastSorted=this.head.next; lastSorted != this.head.prev; lastSorted = lastSorted.next ) {
         firstUnsortedData = (Comparable)lastSorted.next.data;
         for(sortedWalker=lastSorted; sortedWalker != head &&
                  ((Comparable)sortedWalker.data).compareTo(firstUnsortedData) > 0;
                  sortedWalker = sortedWalker.prev) {
              sortedWalker.next.data = sortedWalker.data;
         }
         sortedWalker.next.data = firstUnsortedData;
   }
}
//Insertion Sort is like organizing a handful of playing cards. You pick the first an unsorted card
at a time and insert it into its correct position.
```

Summary

- addOrdered on Circular Doubly Linked List
- Remove and removeAll on Circular Doubly Linked List
- Insertion sort on Array and on Circular Doubly Linked List

Next class

- Recursion