Doubly Linked Lists and Circular Lists

Weiss, section 6.9 (pages 274–276)

I am indebted to my colleague, John David Stone, for assistance with this handout.
This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/4.0/
or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.
Header Node

• Special cases with linked lists
  – E.g., insertion method may add a new item as the first element.
  – Write code to avoid special cases

• Header node
  – Extra node in a linked list that has no data
  – Space overhead
  – Search routines must skip over the header node.
Doubly Linked List

• A node maintains a link to the next node and a link to the previous node.
• Use head and tail to find out whether the start or end of the list is reached
Insertion

```java
newNode = new DoublyLinkedListNode( x );
newNode.prev = current;
newNode.next = current.next;
newNode.prev.next = newNode;
newNode.next.prev = newNode;
current = newNode;
```
Removal

current.prev.next = current.next;
Current.next.prev = current.prev;
current = head;
Circularly Linked List
public class PriorityQueueTester {
    private int a;

    private static class CountyComparatorA {
        PriorityQueueTester obj1 = new PriorityQueueTester();
        obj1.a = 2;
    }

    private class CountyComparatorB { a = 2; }

    public static void main( String[] args ) {
        PriorityQueue<County> pq = new PriorityQueue<County>( new CountyComparator( ) );
    }
}