CS151 - Class Activity

DoublyLinkedLists

Consider the following DoublyLinkedList of Strings. Assume it has correct methods for insertion.

1. Read the following segment of code and find the bug. Draw an example DoublyLinkedList for which `reverseToString` method crashes. What node is `current` equal to when it crashes? What error would java report?

2. Write the `moveToFront` method. It should move the node containing the specified string `data` to the front of the doubly linked list. The node is assumed to be present in the list. This method updates the head of the list to point to the moved node and adjusts all necessary links in the list to maintain the structure.

   Draw 3 "test" Doubly Linked Lists and execute your method on them. Make sure to consider edge cases. Edge cases are special circumstances like when the list has only one element.

```
public class DoublyLinkedList {
    private Node head;
    private Node tail;
    private int size;

    public DoublyLinkedList() {
        head = new Node();
        tail = new Node();

        head.next = tail;
        tail.prev = head;
    }

    public class Node {
        public String data;
        private Node next;
        private Node prev;

        public Node(String data) {
            this.data = data;
        }
    }

    /**
     * Returns a string representation of the elements in the DLL
     * in reverse order, starting from the tail.
     **/
    public String reverseToString() {
        String s = "";

        Node current = tail.prev;
        while (current != null) {
            s += current.data.toString() + "-";
            current = current.prev;
        }

        return s.trim();
    }

    public void moveToFront(String data) {

   }
}
```

ExpandableArray

Assume correct `insert` and `expand` methods exist. Write the `moveToFront` method.

It should move the specified element to the front of the expandable array. It should search for the first occurrence of the given element in the array. Upon finding it, the method shifts all elements between the start of the array and the found element one position to the right. The order of the remaining elements should be preserved.

Draw 3 “test” arrays and execute your method on them. Again, make sure to consider edge cases.

```
public class ExpandableArray<E> {
    private E[] data;
    private int size;


    public void moveToFront(E data) {

    }

}
```

Algorithmic Analysis

What is the big-o notation of the following operations?

1. `moveToFront` - DoublyLinkedList
2. `moveToFront` - ExpandableArray

Compare the two operations. Is one cheaper? Why or why not?