CS151 Intro to Data Structures

LinkedLists

Warmup: Comparing Linked Lists

Write a method that determines whether two linked lists contain the same strings in the same order.

```
public static boolean areEqual(Node list1, Node list2) {
    ....
}
```

Comparing Strings in Java

`compareTo()` method is used to compare two strings lexicographically.

```
int result = str1.compareTo(str2);
```

It compares two strings character by character based on their Unicode values.
The method returns:

- `0` if `str1` is equal to `str2`
- A **negative value** if `str1` is lexicographically smaller than `str2`
- A **positive value** if `str1` is lexicographically greater than `str2`

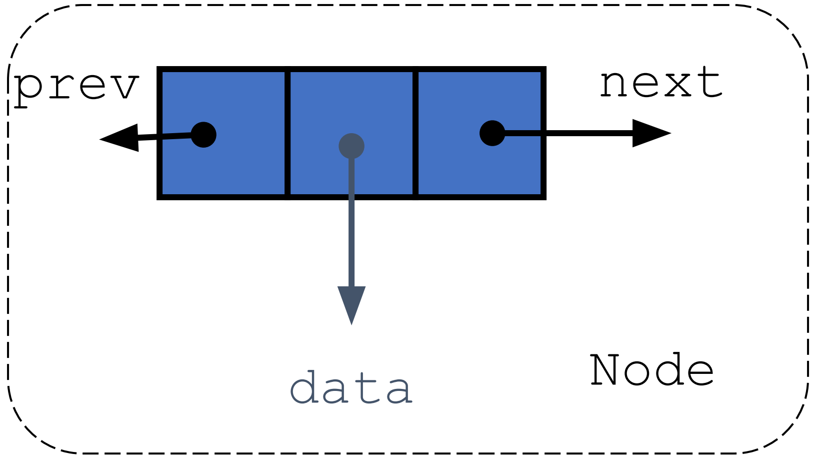
Announcements

- HW02 and Lab 3 due 2/16 (next Monday)
- Prof office hours today 3-5pm Park 259

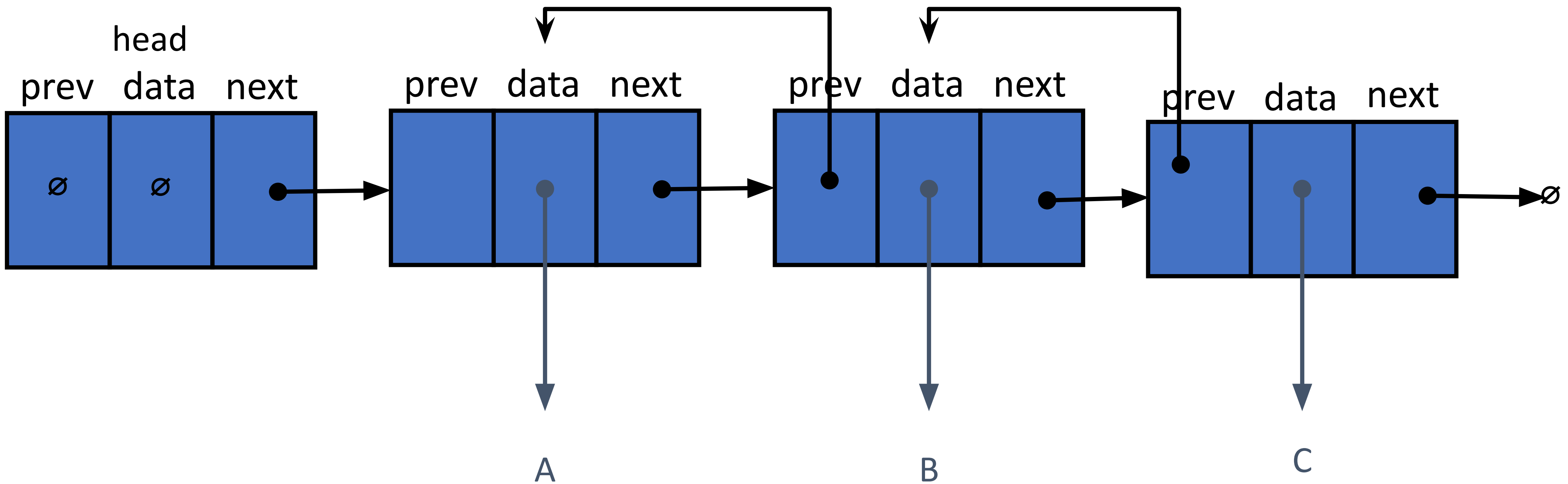
Doubly Linked Lists

A node

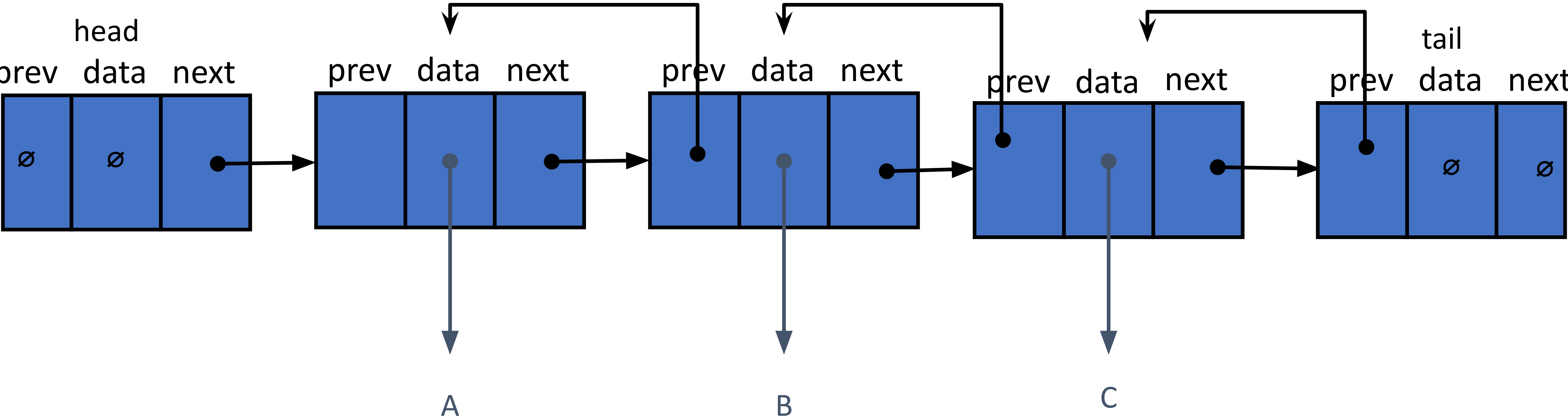
```
public class Node<T> {
   private T data;
   private Node next;
   private Node prev;
}
```



Doubly Linked List



Doubly Linked List



Exercise

Write a `moveToFront(String data)` 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 a “test” Doubly Linked List and execute your method on it.

```java
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;
        }
    }

    public void moveToFront(String data) {

    }
}
```

Exercise 2

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?

Summary

- DoublyLinkedList:
  - tail member variables
  - Nodes have a prev member variable
- Allows for reverse traversal
- Allows for more efficient insert at back (tail)
  - O(n) on SLL
  - O(1) on DLL