Lecture 18

Linked List - 1 → Lab start at 1:00pm

Why? array → if extends
↳ dynamic data
↳ contiguous memory?

Lab
↳ [illegible]
end lab

Rule:

Rule:
head is set ↑
tail is set ↓

Class LinkedList {
private class Node {
int data;
Node next;
Node(int data, Node next) {
this.data = data;
this.next = next;
}
}
int size;
int capacity;
Node head;
Node tail;
LinkedList(int capacity) {
this.capacity = capacity;
}
}

- addFirst(int data)
- addLast(")
- getNodeAt(int idx)
- Node[] toNodeArray()
- getFirst()
- getLast()
- getSize()
- getCapacity()
- removeFirst()
- removeLast()
- removeAt(int idx)
- display()
- size()
- isEmpty()

- addFirst(int data)

- addLast(int data)

getNodeAt(idx)
- Node temp = this.head;
int counter = 0;
while (counter < idx) {
temp = temp.next;
counter++;
}
return temp;

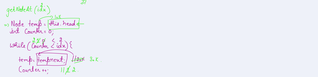
addNodeAt(int data, int idx) {
if (idx == 0) {
addFirst();
} else if (idx == size) {
addLast();
} else {
Node n = getNodeAt(idx-1);
Node node = new Node(data, n.next);
n.next = node;
size++;
}
}

getFirst → head data.
getLast → tail data.
getAt → getNodeAt(idx).data.

removeFirst() {
if (list.size() == 1) {
head = null;
tail = null;
} else {
this.head = this.head.next;
}
size--;
}

removeLast() {
if (size == 1) {
head = null;
tail = null;
} else {
Node temp = getNodeAt(size-2);
temp.next = null;
this.tail = temp;
}
size--;

removeAt(idx) {
if (idx == 0) {
removeFirst();
} else if (idx == size-1) {
removeLast();
} else {
Node temp = getNodeAt(idx);
Node temp2 = getNodeAt(idx-1);
temp2.next = temp.next;
size--;
return temp.data;
}
}



display() {
Node temp = this.head;
while (temp != null) {
System.out.print(temp.data + " -> ");
temp = temp.next;
}
System.out.print("END");
}

10 -> 20 -> 30 -> 40 -> 50 -> END

Questions → Resume at 2:45pm

Reverse
① → Reverse the data iteratively
② → [illegible] the pointers.
③ → Reverse the pointers recursively

int left = 0, right = size-1;
while (left < right) {
int temp = getAt(left);
int temp2 = getAt(right);
Node n = getNodeAt(left);
n.data = temp2;
int temp = n.data;
n = getNodeAt(right);
n.data = temp;
left++;
right--;
}

// 1
// 3
// 2
// 4
// 5
2 mins → 10

① Reverse the pointers iteratively.

public void reverseIt() {
Node prev = null;
Node curr = this.head;
while (curr != null) {
Node temp = curr.next;
curr.next = prev;
prev = curr;
curr = temp;
}
this.tail = this.head;
this.head = prev;
}

O(n)

Node temp, this.head;
this.head = this.tail;
this.tail = temp;
this.tail.next = null;

② Reverse the pointers recursively.

public void reverseIt() {
this.reverseIt(this.head);
}

private void reverseIt(Node node) {
if (node == this.tail) {
return;
}
reverseIt(node.next);
node.next.next = node;
node.next = null;
}

node
tail
1m
2m
3m
4m
5m
10K
20K
30K
40K
50K

Linked List
Doubly