1. Set this.head.prev to be the newNode prepend(item:T) 2. Set newNode.next to be this.head 3. Set this.head to be newNode Step 0: No changes this.head.next this.head Step 1: Creates newNode. prev. Sets this.head.prev to newNode this.head.next newNode this.head Step 2: Sets newNode.next to this.head this.head.next newNode this.head .next Step 3: Sets this.head to newNode this.head this.head.next .next.next .next formerly formerly this.head newNode

node.

0. Check if there is head, if there is no head set head and tail to new

append(item:T)

0. Check if there is head, if there is no head set head and tail to new node.

this.tail

former

newNode

- 1. Set this.tail.next to be the newNode
- 2. Set newNode.prev to be this.tail
- 3. Set this.tail to be newNode

Step 0: No changes this.tail this.tail.prev Step 1: Creates newNode. .next newNode Sets this.tail.next to newNode this.tail.prev this.tail Step 2: .prev Sets newNode.prev to this.tail this.tail.prev this.tail newNode Step 3: .prev Sets this.tail to newNode this.tail.prev this.tail .prev.prev former

insertAt(item:T, idx: number)

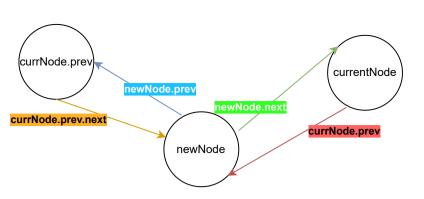
Step 0: Traverse to the **idx** and get the **currentNode**

Step 1: Set currNode.prev.next to newNode

Step 2: Set newNode.prev to currNode.prev

Step 3: Set newNode.next to currentNode

Step 4: Set currNode.prev to newNode





remove(item:T)

removeAt() is the same but traverse to idx

Step 0: Traverse while there is currNode. Get the currNode with the searched item/value.

Step 1: Check if there is **currentNode.prev**, if there is then => Set **currNode.prev.next** to **currNode.next**;

Step 2: Check if there is **currentNode.next**, if there is then => Set **currNode.next.prev** to **currNode.prev**

Step 3: Check if currentNode is head, if it is then => set this.head to currentNode.next;

Step 4: Check if currentNode is tail, if it is then => set this.tail to currentNode.prev;

Step 5: Set currNode.next = currNode.prev = null;

