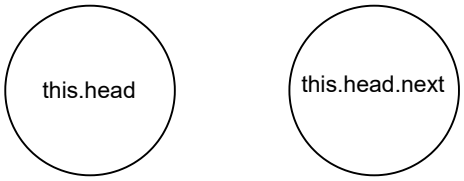
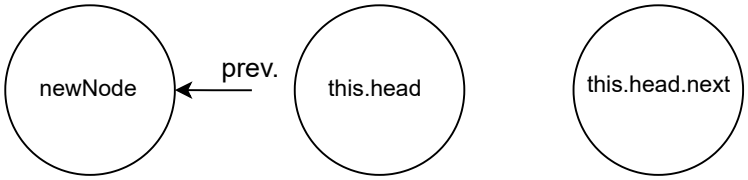
prepend(item:T)

0. Check if there is head, if there is no head set head and tail to new node.
1. Set **this.head.prev** to be the **newNode**
2. Set **newNode.next** to be **this.head**
3. Set **this.head** to be **newNode**

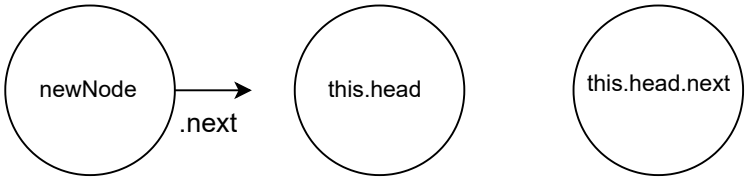
Step 0: No changes



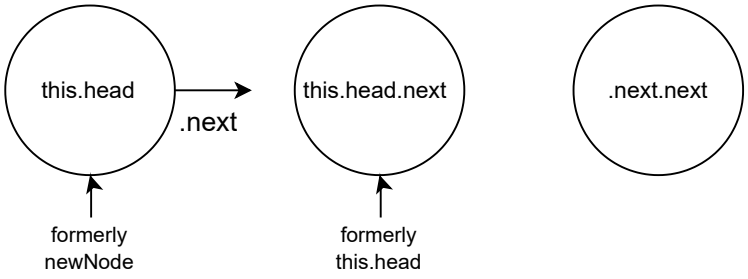
Step 1: Creates newNode.
Sets **this.head.prev** to newNode



Step 2:
Sets **newNode.next** to this.head



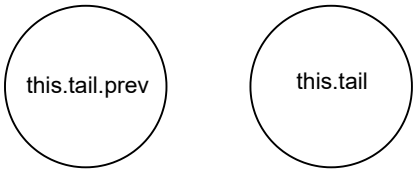
Step 3:
Sets **this.head to newNode**



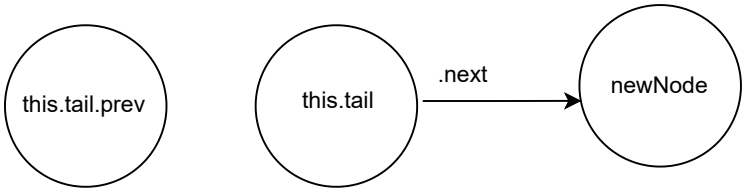
append(item:T)

0. Check if there is head, if there is no head set head and tail to new node.
1. Set **this.tail.next** to be the **newNode**
2. Set **newNode.prev** to be **this.tail**
3. Set **this.tail** to be **newNode**

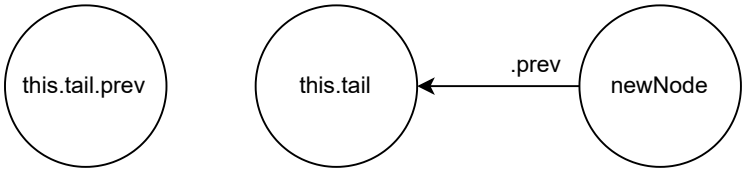
Step 0: No changes



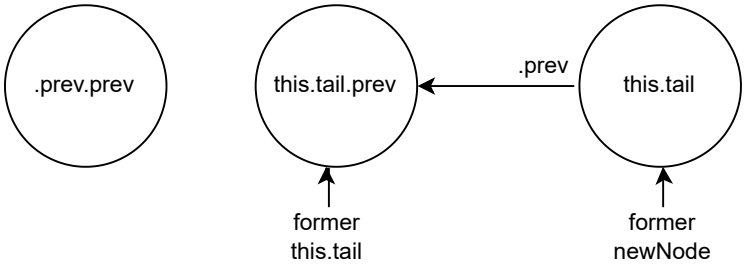
Step 1: Creates newNode.
Sets **this.tail.next** to newNode



Step 2:
Sets **newNode.prev** to this.tail



Step 3:
Sets **this.tail to newNode**



insertAt(item:T, idx: number)

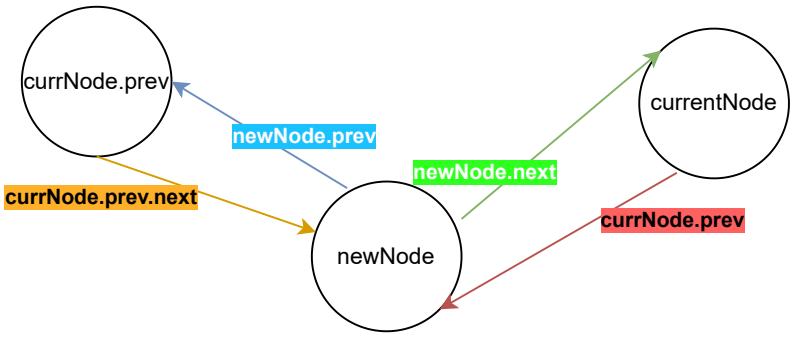
Step 0: Traverse to the **idx** and get the **currentNode**

Step 1: Set **currNode.prev.next** to **newNode**

Step 2: Set **newNode.prev** to **currNode.prev**

Step 3: Set **newNode.next** to **currentNode**

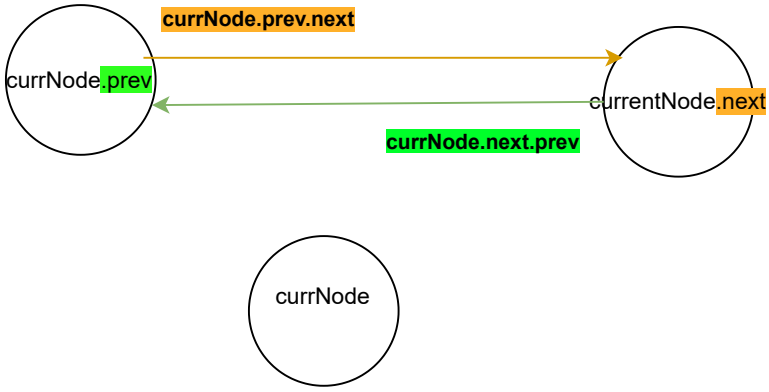
Step 4: Set **currNode.prev** to **newNode**



currNode.next

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;