EECS2011 Fundamentals of Data Structures (Winter 2022)

Q&A - Week 4 Lecture

Thursday, February 10

Announcements

- Written Test 1 due next Monday or Tuesday
- Revised start time of Written Test 1
- Example questions for Written Test 1 released

- Assignment 1 (on SLLs) due next Tuesday

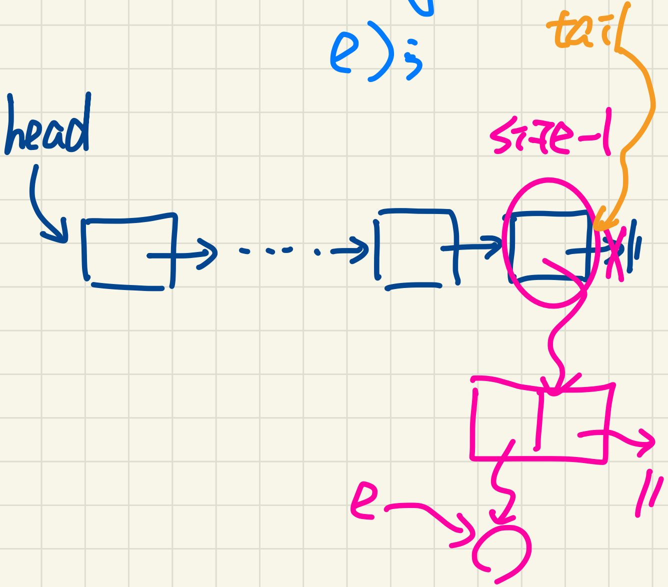
- Lecture W5 postponed until next Wednesday

```
void addAt (int i, String e) {
  if (i < 0 || i > size) {
    throw new IllegalArgumentException("Invalid Index.");
  }
  else {
    if (i == 0) {
      addFirst(e);
    }
    else {
      Node nodeBefore = getNodeAt(i - 1);
      Node newNode = new Node(e, nodeBefore.getNext());
      nodeBefore.setNext(newNode);
      size ++;
    }
  }
}
```

size

```
else if (i == size){
    addLast(e);
}
```

```
list.addAt(list.getSize(),
           e);
```



Problem on SLL: Removing the N^th Node from the End

You are asked to program this method:

```
public ListNode removeNthFromEnd(ListNode head, int n)
```

Remove the n^th node from the end of the chain starting from head.

Requirement: n ≤ number of nodes in the input chain

https://leetcode.com/problems/remove-nth-node-from-end-of-list/

```java
@Test
public void test_1() {
    ListNode input =
        new ListNode(1,
        new ListNode(2,
        new ListNode(3,
        new ListNode(4,
        new ListNode(5, null)))));
    ListUtilities util = new ListUtilities();
    ListNode output = util.removeNthFromEnd(input, 2);
    assertTrue(input == output);
    assertTrue(input.val == 1);
    assertTrue(input.next.val == 2);
    assertTrue(input.next.next.val == 3);
    assertTrue(input.next.next.next.val == 5);
}
```

```java
@Test
public void test_2() {
    ListNode input = new ListNode(1, null);
    ListUtilities util = new ListUtilities();
    ListNode output = util.removeNthFromEnd(input, 1);
    assertTrue(output == null);
}
```

$O(size)$ ✓ 1. go throug list, get $\underline{size}$ 5.

$n==2$

$O(1)$ 2. calculate $\underline{index}$ of node prior to the none to remove: $\underline{size-n-1}$ 3

Worst case: $size - \underline{n} - 1$ max
↳ $n==1$
↓
$size-2$
linear

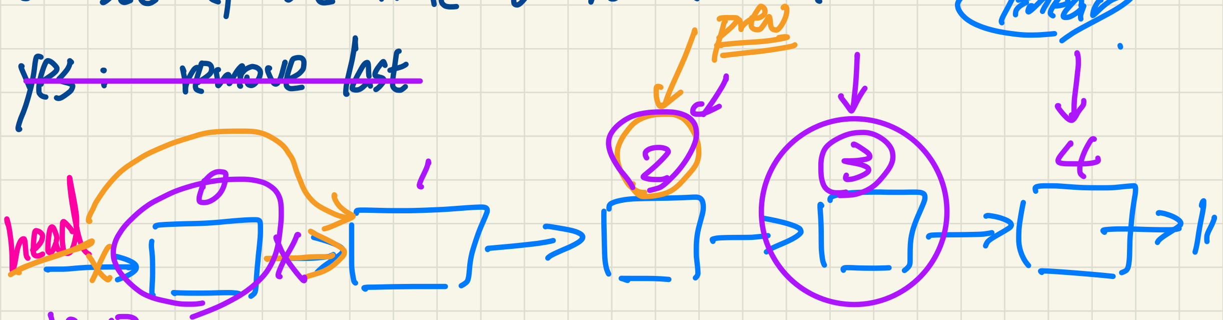
$O(size)$ ✓ 3. ~~check to see if the node to remove is last~~

~~yes: remove last~~

linear

loop to index $size-n-1$

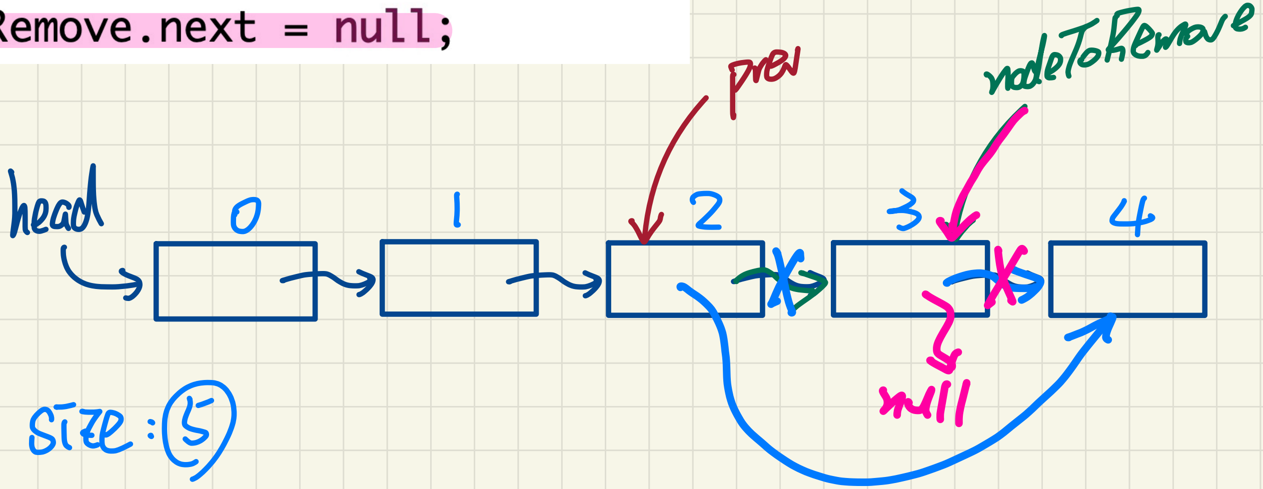
store 3 to temp
set 2
reset temp



Special Case

removeNthFromEnd(head, $\underline{5}$);
size
↳ index of 'prev': $5-5-1 == -1$

```
ListNode nodeToRemove = prev.next;
prev.next = nodeToRemove.next;
nodeToRemove.next = null;
```



size : 5

removeNthFromEnd(head, 2);

indexOfPrev == 5-2-1 == 2