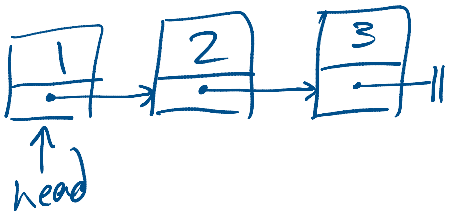
Week 4: Linked Lists

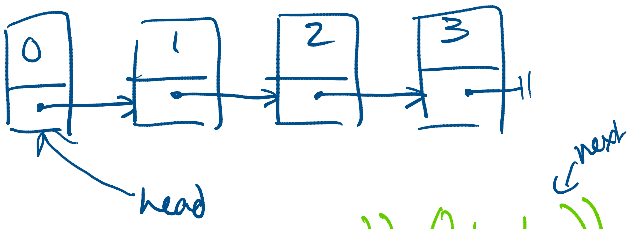
Friday, September 25, 2020 8:51 AM

Structure

- node
  - data
  - reference to next
- head
- tail (optional)



Add to head



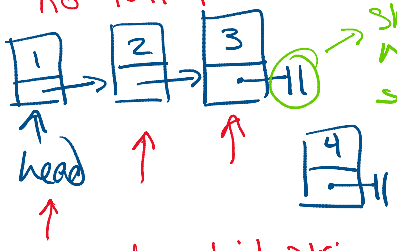
* head = new Node(data, head)

O(1) runtime – constant

Add to Tail

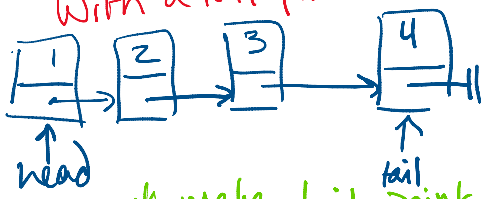
no tail ptr:

- loop through
- stop when next is null
- set next to new node



*O(n)

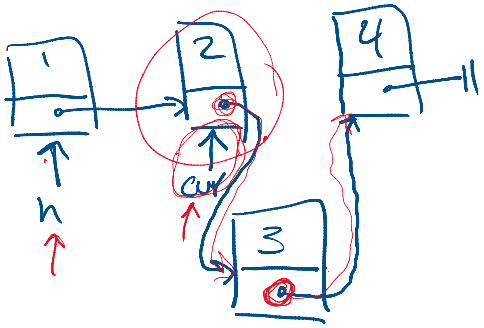
With a tail ptr:



* make tail point to new
* update tail

O(1)

Add to Middle



1. find node before where you want to add
2. make new node's point to cursor's next
3. update cursor's next

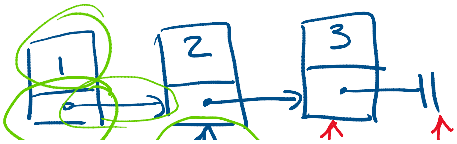
O(n) worst-case runtime

insert(index, element)

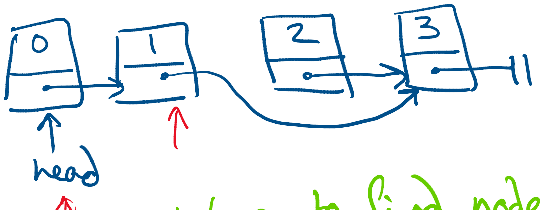
insert(3, 3)

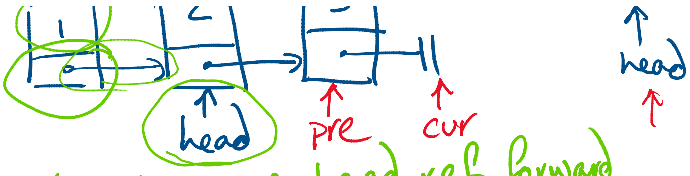
Remove

at head → remove 1



Middle *remove 2





*just move head ref forward
*old is garbage collected

O(1)

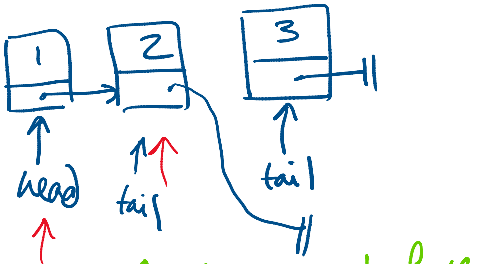
*loop to find node before the one to remove
*set node's next to next.next

```
for (Node p=head; p.next != target;
     p= p.next);
```

O(n)

Remove from tail

have tail ptr



*loop to find one before
* set before's next to next.next
* update tail

O(n)

Node class

→ nested class inside LL class
→ static class
  → can't see outside of its scope
    * can't see manyNode, precursor, cursor, etc
  → saves space in memory
  → simpler — can't mess up linked list

*private

contrasted w/ iterator
→ iterator is non-static
  → can see collection it is in

Clone

precursor

Clone

* make sure precursor & cursors point to the correct cloned node
* loop through & clone each node

Constructor for node: (data, next)