Week 4: Linked Lists

Friday, September 25, 2020 11:01 AM

Structure

- head ptr
- node class
  - data
  - next ptr

*optional tail ptr

Static class

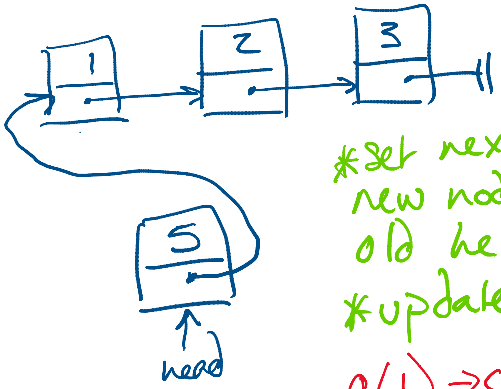
- no access outside of itself
- no access to linked list or anything outside

*save a bit of memory

Contrast: iterator class

- non-static
  - *can see outside of itself
  - *can access / change the collection

Adding at head

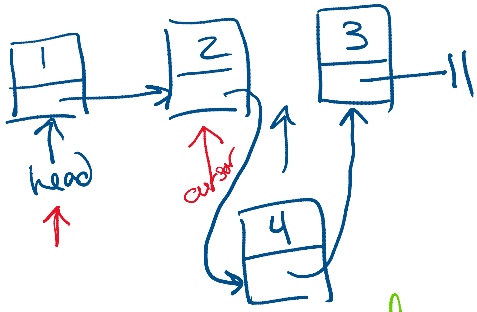


*set next of new node to old head

*update head

O(1) → constant

Add to the middle



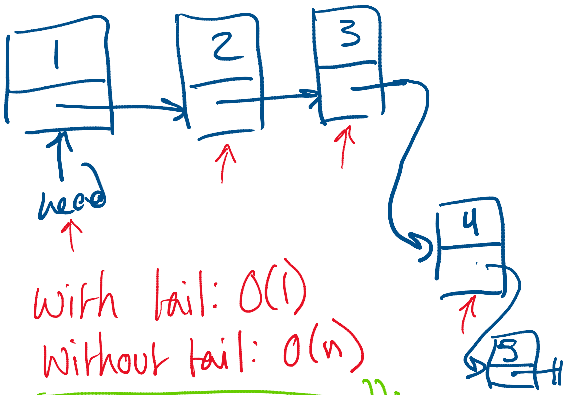
*find node before where adding

* set new node's next to cursor's next

* set cursor's next to new node

O(n) → linear

Add at end



With tail: O(1)

Without tail: O(n)

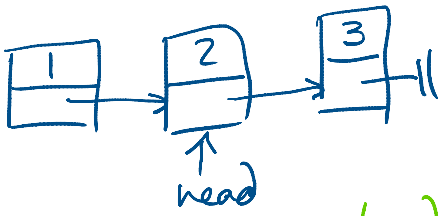
↳ Same as adding in middle → not a special case

Removal

from head

Remove from middle

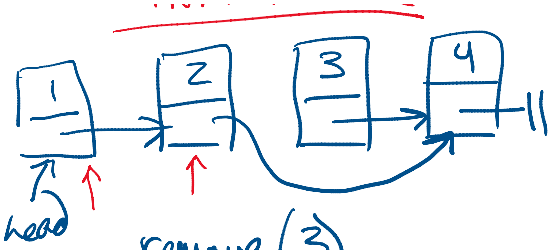
Removal from head



*point head to head.next

*Java's garbage collector will handle old node

O(1) → constant



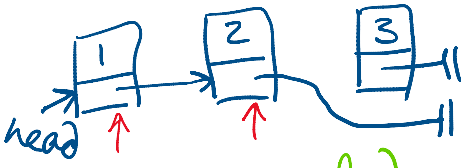
remove(3)

*iterate through & stop at the node before the one to remove

*set cursor's next to next.next

O(n) → linear

Remove from end



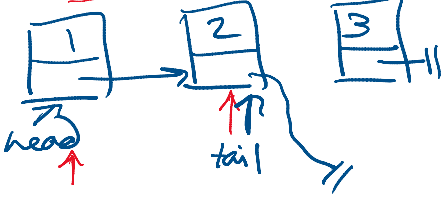
*iterate to find node before removed node

*set cursor's next to next.next

*same as removing from middle – no special case

O(n) → linear

have tail ptr



*need to iterate until cursor.next is the tail

*set cursor.next to next.next

*update tail

O(n)

Remove only item

→ head to null

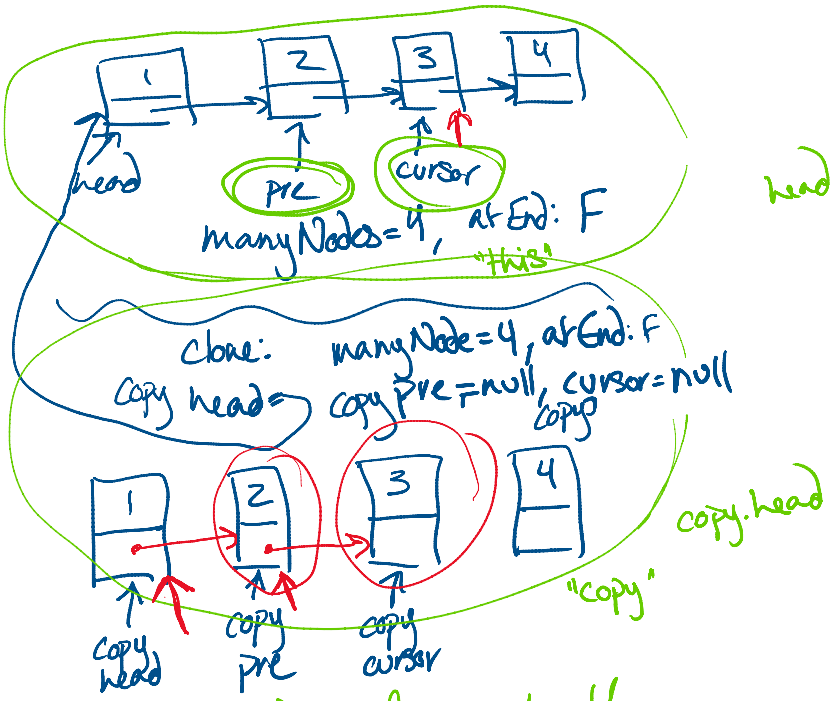
→ tail to null if tail ptr.

head=null O(1)

Cloning

→ Shallow clone:
  → primitives are set
    * don't worry about many nodes
    * atEnd
  → doesn't clone references/objects

→ clone every single node
→ set pre cursor & cursor to correct cloned nodes



* need reference to the "before" node in clone