Linked Lists

Wednesday, February 12, 2020 10:05 PM

Components

- many items
- head, tail, cursor/pre.
- node
  - data
  - next pointer

*nested static class

- static: cannot see anything in the class it is nested in
- unlike iterator, which is a nested non-static class
  - can see collection

*what should next point to if it is the last in the list?

- null

Invariant?

*manyNodes is correct

*tail must be last in list started @ head

*if head is null, tail must be null

*no cycles

*precursor is null or points to a node in this list

have sorted linked list w/ head & tail

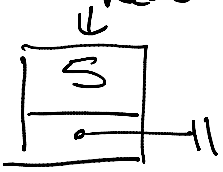
```
private static node {
    int data;
    Node next;
```

```
int data;
Node next;
public Node(int d, Node n){
    data=d;
    next=n;
}
}
head= null;
cursor = null;
```

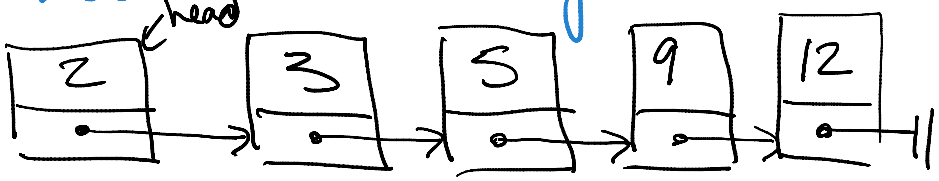
tail = null

add a 5 to empty list

```
head= new Node(5, null);
```

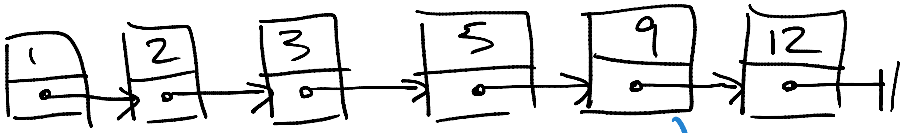


add to a longer list



add 1 (at beginning)

```
head= new Node(1, head);
```



add 15 (at end)

w/ tail:

tail = tail.next = new Node(15, null);

w/out tail:

start @ head & find tail

for(Node n = head; n.next != null; n = n.next);

n.next = new Node(15, null);

* add in between

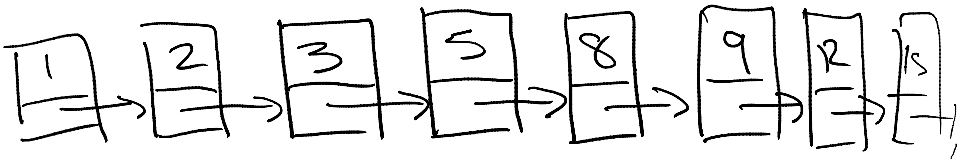
→ add 8

*what do we need?

→ ptr to node that comes before 8.

for(Node n = head; n.next != null;
n = n.next)
if(n.next.data >= 8) break;

n.next = new Node(8, n.next);



*What is the runtime if we are not guaranteed to add at the head?

O(n)

*what if we always add at the head?

O(1)

*think about the ball sequence. What does append do?

→ if current elem, add after

→ else, at head

*what was the runtime w/ a dynamic array?

O(n) *shift

*what is runtime w/ linked list & cursor?

O(1)

*removal from beginning

head = head.next;

*why is this sufficient?

*why is this sufficient?

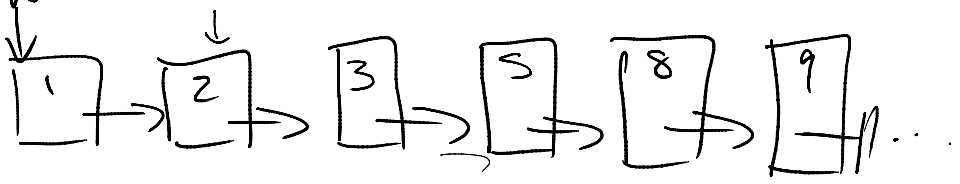
→ if there is no reference to an object, the garbage collector gets rid of it.

→ the old head will be removed

*remove from end or middle?

→ need to find the prev node

remove(3)




```
for(Node n = head; n.next != null;
    n = n.next)
    if(n.next.data == target){
        n.next = n.next.next;
        return true;
    }
return false;
```

*remove only item?

*remove only front:

head = tail = null;

*what is the runtime?

O(n)

* what if only remove at head (stack) or w/ a cursor?

O(1)

	array w/cursor	linked list w/cursor prev
add	O(n)	O(1)
remove	O(n)	O(1)

remove	O(n)	O(1)
getCurrent	O(1)	O(1)
get ith	O(1)	O(n)

*pros of linked lists?
→ faster insert/remove
→ no need to resize

*pro of array?
→ fast random access

*watch out for null pointers!
→ if list is empty, all pointers are null
→ last element's next is null
→ make sure to check
→ if cursor is at the head → precursor is null!
- make sure you

head → previous ...

→ make sure you handle that.

* how do you handle clone?

→ clone creates fields, but doesn't create a new list.