CS 106
INTRODUCTION TO DATA STRUCTURES

SPRING 2020

PROF. SARA MATHIESON

HAVERFORD COLLEGE

FEB 13 OUTLINE

- Review check-in
- Singly Linked Lists
- Doubly Linked Lists

- Lab 2 due next Wednesday
- Before lab tomorrow, begin pre-lab (okay to finish in lab)

FEB 13 OUTLINE

- **Review check-in**
- Singly Linked Lists
- Doubly Linked Lists

REVIEW CHECK-IN

1. What is the big-O runtime of the following code block? Answer in terms of n.

```
n = int(input("n: "))
for i in range(n):
  print(i)
for j in range(10*n):
  print(j)
```

REVIEW CHECK-IN

1. What is the big-O runtime of the following code block? Answer in terms of n.

```
n = int(input("n: "))
for i in range(n):
  print(i)
for j in range(10*n):
  print(j)
```

$\boxed{O(n)}$

2. What is the big-O runtime of the following code block? Answer in terms of n.

```
n = int(input("n: "))
for i in range(n):
  for j in range(n):
    for k in range(n):
      print(i, j, k)
```

REVIEW CHECK-IN

1. What is the big-O runtime of the following code block? Answer in terms of n.

```
n = int(input("n: "))
for i in range(n):
  print(i)
for j in range(10*n):
  print(j)
```

$\boxed{O(n)}$

2. What is the big-O runtime of the following code block? Answer in terms of n.

```
n = int(input("n: "))
for i in range(n):
  for j in range(n):
    for k in range(n):
      print(i, j, k)
```

$\boxed{O(n^3)}$

3. What is the big-O runtime of the following code block? Answer in terms of n.

```
n = int(input("n: "))
while n > 1:
  print(n)
  n = n/2
```

REVIEW CHECK-IN

1. What is the big-O runtime of the following code block? Answer in terms of n.

```
n = int(input("n: "))
for i in range(n):
  print(i)
for j in range(10*n):
  print(j)
```

$\boxed{O(n)}$

2. What is the big-O runtime of the following code block? Answer in terms of n.

```
n = int(input("n: "))
for i in range(n):
  for j in range(n):
    for k in range(n):
      print(i, j, k)
```

$\boxed{O(n^3)}$

3. What is the big-O runtime of the following code block? Answer in terms of n.

```
n = int(input("n: "))
while n > 1:
  print(n)
  n = n/2
```

$\boxed{O(n \log(n))}$

REVIEW CHECK-IN

4. Circle and correct the error in the code for the `add` method of `ArrayList`. Then fill in the missing line.

```
public void add(T elem) {
    int l = array.length;
    if (numElem == l) {
        T[] arrayNew = (T[]) new Object[l+2];
        for (int i=0; i < l; i++) {

            __________________________________________;
        }
        array = arrayNew;
    }
    array[numElem] = elem;
    numElem += 1;
}
```

REVIEW CHECK-IN

4. Circle and correct the error in the code for the `add` method of `ArrayList`. Then fill in the missing line.

```
public void add(T elem) {
    int l = array.length;
    if (numElem == l) {
        T[] arrayNew = (T[]) new Object[l+2];
        for (int i=0; i < l; i++) {

            ________________________________________;
        }
        array = arrayNew;
    }
    array[numElem] = elem;
    numElem += 1;
}
```

Constant size increases will lead to quadratic runtime!
Need to increase by a **factor** of the existing size (i.e. l*2)

REVIEW CHECK-IN

4. Circle and correct the error in the code for the `add` method of `ArrayList`. Then fill in the missing line.

```
public void add(T elem) {
    int l = array.length;
    if (numElem == l) {
        T[] arrayNew = (T[]) new Object[l+2];
        for (int i=0; i < l; i++) {
            arrayNew[i] = array[i];  // copy over contents
        }
        array = arrayNew;
    }
    array[numElem] = elem;
    numElem += 1;
}
```

Constant size increases will lead to quadratic runtime!
Need to increase by a **factor** of the existing size (i.e. l*2)

REVIEW CHECK-IN

5. Say I initialize an array with `int[] nums = new int[100000];`. Why does it *not* take 100 steps to execute the following line of code? `int n = nums[100];`

REVIEW CHECK-IN

5. Say I initialize an array with `int[] nums = new int[100000];`. Why does it *not* take 100 steps to execute the following line of code? `int n = nums[100];`

An array is a continuous block of space in memory, and each cell has equal size. Therefore we can use addition to find the memory location of the data at any given cell. For example, if the array started at memory location **87**, we could find the given value with:

87 + 4*100 = 487

6. In a singly linked list, what instance variables do `Node`s have? In other words, what information does a `Node` store?

REVIEW CHECK-IN

5. Say I initialize an array with `int[] nums = new int[100000];`. Why does it *not* take 100 steps to execute the following line of code? `int n = nums[100];`

An array is a continuous block of space in memory, and each cell has equal size. Therefore we can use addition to find the memory location of the data at any given cell. For example, if the array started at memory location **87**, we could find the given value with:

$$87 + 4*100 = 487$$

6. In a singly linked list, what instance variables do `Node`s have? In other words, what information does a `Node` store?

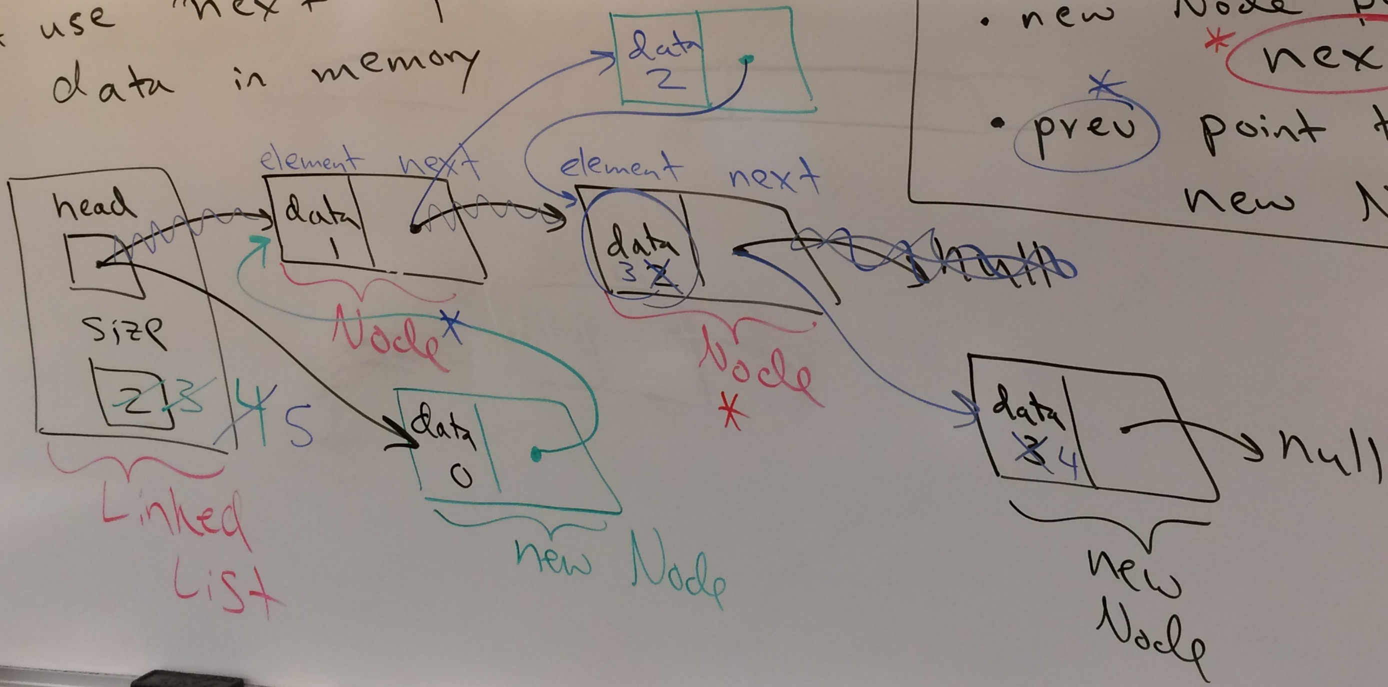
- Important data point
- Reference to the “next” node in the list

FEB 13 OUTLINE

- Review check-in
- **Singly Linked Lists**
- Doubly Linked Lists

Singly Linked Lists

* abandon idea that data must be continuous in memory
* use "next" pointer to find data in memory



③ add to middle

index : 2

O(n)

* traverse until I find the index
* new Node points to next
* prev point to new Node

update size

① add to end

O(n)

right now: slow!

- traverse list to the end
- make last node point to new node ← update size

② add to beginning

O(1)

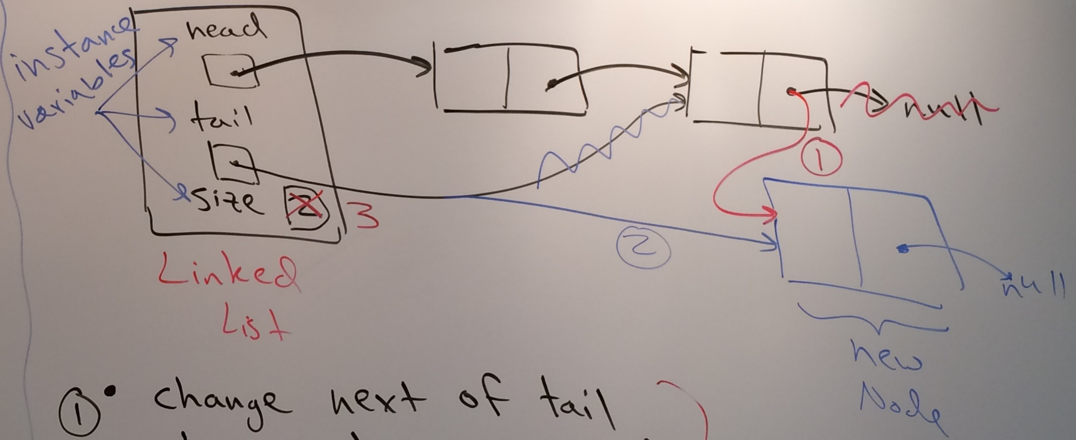
- have new Node point to head
- have head point to new Node

update size

Lists with tail

(fast add to end)

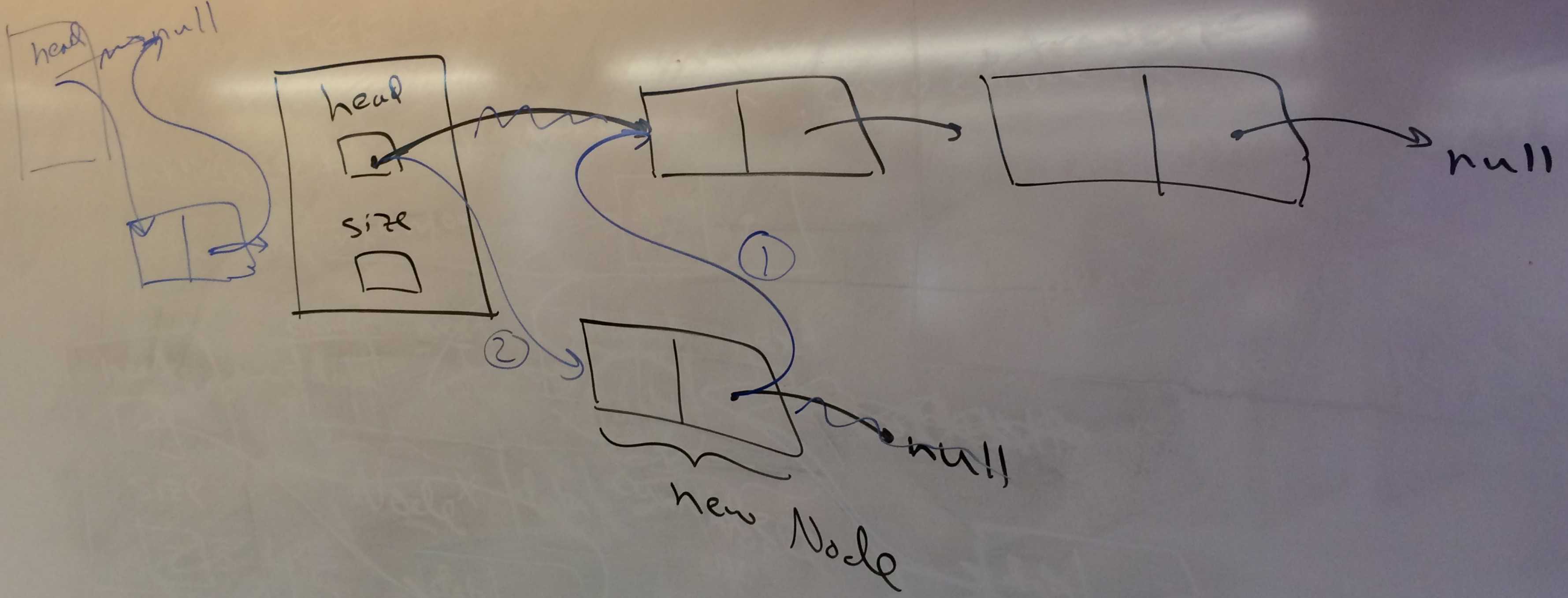
O(1)



① • change next of tail to point to new Node

② • change tail to point to new Node

} change size


head
null
head
size
null
1
2
null
new Node

NODE CLASS

```java
/**
 * Node class for use in LinkedLists. A Node contains
 * its data (generic type) and a reference to the next Node.
 * @author Sara Mathieson
 * @version February 11, 2020
 * @param <E> the type of the data each Node contains.
 */
public class Node<E> {

    private E data;        // important data
    private Node<E> next;  // reference to the next node

    /**
     * Constructor creates the Node and sets its next to null.
     * @param initData: the data this node will contain
     */
    public Node(E initData) {
        this.data = initData;
        this.next = null;
    }

    /**
     * Setter for the next Node (flexibility!)
     * @param next: the Node that comes after this one
     */
    public void setNext(Node<E> next) {
        this.next = next;
    }

    /**
     * Getter for the next Node (convention to call it next)
     * @return the Node after this one
     */
    public Node<E> next() {
        return this.next;
    }
}
```

FEB 13 OUTLINE

- Review check-in
- Singly Linked Lists
- Doubly Linked Lists

Next time!