More recursion and its running time

2/1/16
Quiz 2
Announcements

• HW due tomorrow night (RPN calculator)
• Reading: start Chapter 8
• Make up quizzes: TBA soon
• Quiz on Friday: Writing a recursive list method
Recurrence relations

• $T(n) =$ time to run countdown on $n$ numbers
Recurrence relations

- $T(n) =$ time to run countdown on $n$ numbers
- $T(n) = 1 + T(n-1)$
- $T(1) = 1$

Constant time to print $n$
By definition, the time for recursive call
Recurrence relations

- \( T(n) = \) time to run countdown on \( n \) numbers
  - \( T(n) = 1 + T(n-1) \)
  - \( T(1) = 1 \)
  - Can prove by induction that \( T(n) = n \)
Recurrence relations

- $T(n) = \text{time to run countdown on } n \text{ numbers}$
- $T(n) = 1 + T(n-1)$
- $T(1) = 1$

- Can prove by induction that $T(n) = n$
- I prefer to do it graphically; draw recursive calls going down and compute non-recursive work for each level
Fibonacci numbers

\[ F_0 = 0 \]
\[ F_1 = 1 \]
\[ F_n = F_{n-1} = F_{n-2} \text{ for } n > 1 \]
Fibonacci numbers

\[ F_0 = 0 \]
\[ F_1 = 1 \]
\[ F_n = F_{n-1} = F_{n-2} \text{ for } n > 1 \]

**Never** compute these with the obvious code:
public int fib(int n) {
    if(n <= 1)
        return n;
    return fib(n-1) + fib(n-2);
}
More recursive methods

- `numOccurrences(T)`
- `remove(T)`
- `toString()`
- `get(int)`
- `indexOf(T)`
- ...
Which of the following correctly implements the numOccurrences method in EmptyNode?

A. return 0;
B. return 1;
C. if(value.equals(val)) return 1 else return 0;
D. Node curr = head;
   int count = 0;
   while(curr != null) {
       if(curr.value.equals(val)) count++;
       curr = curr.next;
   }
   return count;
E. Not exactly one of the above
Which of the following correctly implements the numOccurrences method in EmptyNode?

A. `return 0;`
B. `return 1;`
C. `if(data.equals(val)) return 1 else return 0;`
D. `Node curr = head;
   int count = 0;
   while(curr != null) {
      if(curr.data.equals(val)) count++;
      curr = curr.next;
   }
   return count;`
E. Not exactly one of the above