Outline

- Recursion
  - Definition and Setup
  - Tracing
  - Pitfalls
  - Running Time
- Examples
- Recursive Data Structures
- Conclusion
Some data structures can also be recursive

- What does it mean?
  - Has another version of itself in class definition

Already seen one: Linked Lists

- Base case:
  - A trivial linked list has no nodes (i.e., is null)

- Recursive case:
  - A general linked list consists of a head node that points to a linked list
Lists are recursive structures.

- I'll use LinkedLists to demonstrate the recursive nature

When defining anything recursively, we need base and general cases.
Focus on the *base case*.

- What does the *trivial* LinkedList look like?

Maybe a List containing one node? (Can it get simpler than this?)

- How about an empty list?
Now the *general case*: a general List can be described recursively:

- A head node followed by a sub-list.
Lists (Cont.)

- Recursive Def'n of Lists.
  - A List is:
    - Null, or (Base case)
    - A head node followed by a List (General case)
Recursive Def'n of Lists.

- A List is:
  - Null, or (Base case)
  - A head node followed by a List (General case)
Recursive Def'n of Lists.

- A List is:
  - Null, or (Base case)
  - A head node followed by a List (General case)
Recursive Def'n of Lists.

- A List is:
  - Null, or (Base case)
  - A head node followed by a List (General case)
Recursive Def'n of Lists.

- A List is:
  - Null, or (Base case)
  - A head node followed by a List (General case)
Recursive List Operations

- Since Linked Lists are recursive, we can easily define (or redefine) their operations to be recursive too!

- Let's (re)define these recursively *(RecLinkedList code)*
  
  1. **String toString()**
     - Returns the string representation of the list
     - For example, "[David David Bob]"
  2. **void replaceAll(E target, E new_item)**
     - Replaces all occurrences of target with new_item
@Override
public String toString() {
    return "[" + this.toString_helper(head) + "]";
}

/**
 * Helper method for getting the string recursively. If the current node
 * is null, then the string is empty. Otherwise, the string is the data stored
 * in the current node, and the string representation of the remaining list.
 */
private String toString_helper(Node<E> node) {
    if (node == null) {
        return "";
    }
    return node.data.toString() + this.toString_helper(node.next);
}

- Your turn: Modify it so that commas are added between each item
  - But not after the last item
public void replaceAll(E target, E newItem) {
    this.replaceAll_helper(this.head, target, newItem);
}

/**
 * Recursive helper method. The base case is when the current node is null,
 * then nothing needs to be done. The recursive case is to check if the current
 * node needs to be replaced. If so, do it, followed by the replacement of the
 * remaining nodes.
 */
private void replaceAll_helper(Node<E> node, E target, E newItem) {
    if (node != null) {
        if (node.data.equals(target)) {
            // this node's data needs to be replaced!
            node.data = newItem;
        }
        // now move on to the next node
        this.replaceAll_helper(node.next, target, newItem);
    }
}

Your turn: Modify it so it also returns the number of items replaced!
Outline

- Recursion
  - Definition and Setup
  - Tracing
- Examples
  - Factorial
  - Fibonacci Numbers
  - List Sum
  - Checking Palindrome
  - Binary Search Revisited
- Recursive Data Structures
- Conclusion
Conclusion

- A recursive method is one that calls itself to solve a problem
  - One of the key concepts in CS and math
  - Many problems are naturally recursive: factorial, Fibonacci, etc.
    - *Recursion is for programming-simplicity*

- Base case is important
  - Recursion won't terminate without it!
    - If the base-case is never met, running the code will result in *a stack overflow*

- General case
  - Does a few trivial steps to reduce the problem size (toward the base case), and makes a recursive call on the smaller subproblem
Start thinking about midterm 2

Due dates:
- Hwk 5 due Friday
- Hwk 4 autocomplete re-grade due Friday

Talk Tonight!
- Machine Learning at Amazing
Hwk 4 returned, regrade next Wednesday

Reminders:

- Hwk 5 due Friday
- Start looking at Review Guide 2 in preparation for exam II (next Tues)
  - Should be able to solve all the problems after today

Lab 8 post-mortem

- Recursive code reading (What does it do? What are the bugs?), some code writing
- Solutions posted
Volunteers needed for ICPC (tomorrow)

Hwk 4 regrade due next Wednesday

Reminders:
  • Hwk 5 due tonight!

Last time...
  • Telescoping method for analysis
  • Recursive data structures
  • Start Trees