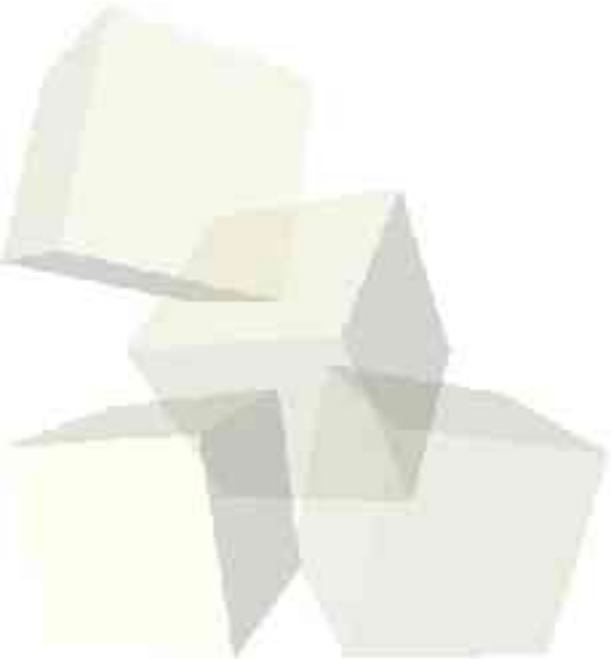
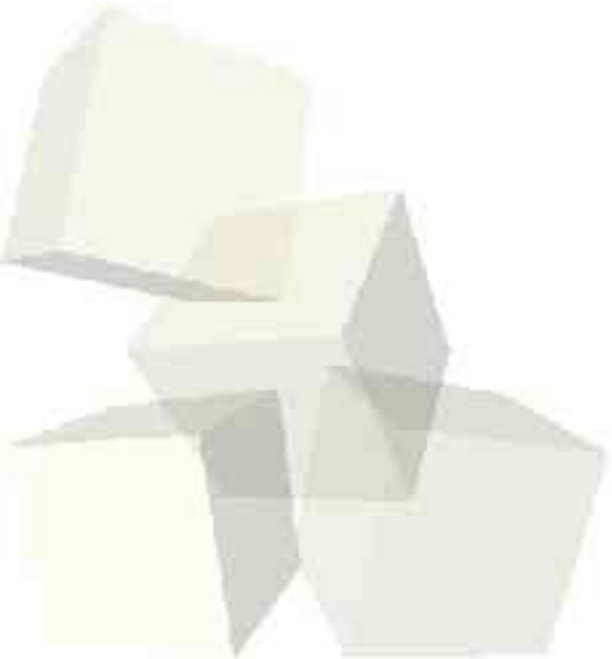
Deep Recursion

9-15-2004



Opening Discussion

- What did we talk about last class?
- Do you have any questions about the assignment?



Efficiency vs. Simplicity

- Last time I espoused the benefits of recursive functions that build their solution as they “pop back up the stack”.
- When this works well, it is the ideal way of doing things in Scheme.
- However, it doesn't always work well. Let's look at a function to reverse a list as an example of this.

Representing Trees in Lists

- Let's be more specific about how we can use lists to represent trees in Scheme. We can think of this in several different ways.
- One simple one is to think of car as a left child and cdr as a right child.
- Alternately, a list can be a subtree where each top level element is a child. If those elements are lists they represent their own subtrees.
- Both these images have the problem that data is only in leaves.

Fibonacci Numbers

- A standard example of a recursive algorithm is the Fibonacci numbers. This is a series where each element is equal to the sum of the previous two.
- We can write this as

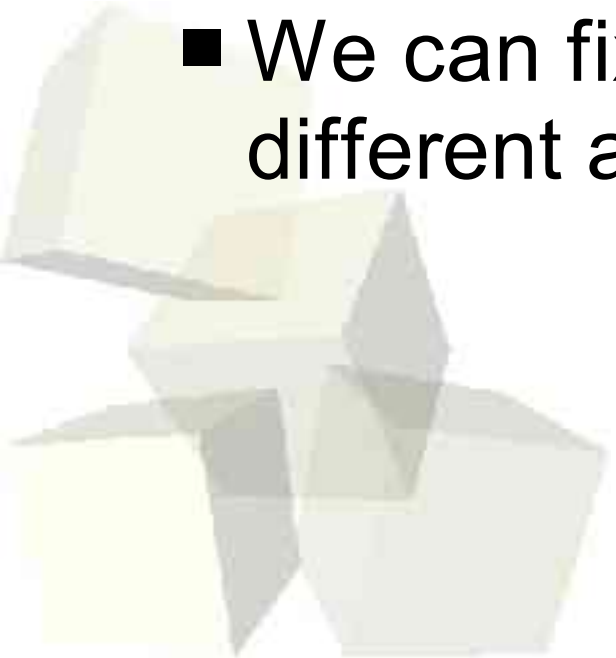
$$f(n) = \begin{cases} n & n < 2 \\ f(b-1) + f(n-2) & otherwise \end{cases}$$

Counting Calls and Adds

- The problem with this definition is that a simple program can take a while to execute for larger values of n.
- Let's explore this by writing a simple version, then writing similar methods that count how many times we call the method or how many adds we do.

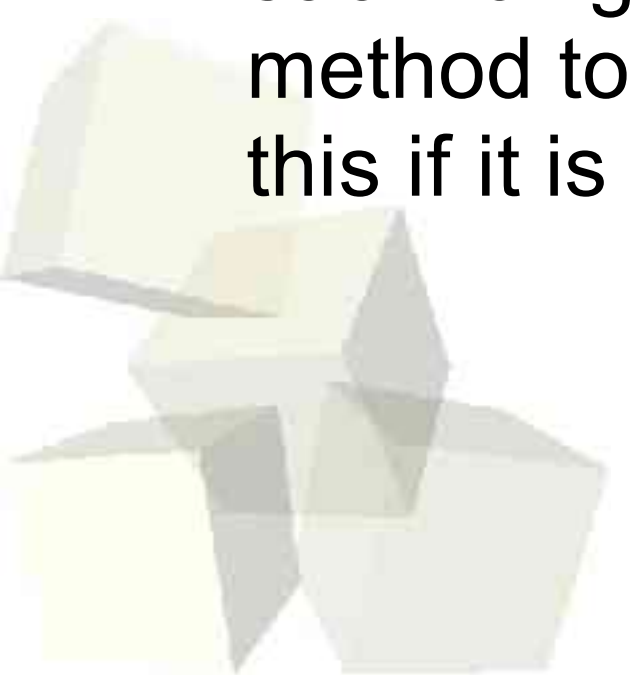
Problems of Repeating Work

- To see the real problem, let's draw out what happens when you call (f 6).
- As you can see, we repeat a lot of work. The function gets called multiple times with the same argument.
- We can fix that by using a somewhat different approach.



An Iterative Solution

- Here we can make an iterative solution that uses a helper function with 3 arguments to make this function work in linear time.
- As with the reverse method, we are sacrificing some of the elegance of our method to get efficiency. We should only do this if it is really needed.



Minute Essay

- The method of storing data in lists in Scheme leads to a certain fundamental speed issue. Can you think of what this is? Why do we have that problem? How could it be fixed?