Recursion

11-7-2002

Opening Discussion

- What did we talk about last class?
- Do you have any questions about the assignment?
- What is a recursive function? Why do we use them? What are they good for?

What is Recursion?

- A recursive function in mathematics is one that is defined in terms of itself. In computer science it is a function that calls itself.
- Mathematically these functions have a general relation for most input values as well as some defined "termination" values.
- The Fibonacci numbers are a nice example of a recursive function, but we can also define factorial recursively.

Recursion and the Stack

- Java and most other modern programming languages use a stack for function calls. The stack is a section of memory that sits at one end of the memory that program owns.
- Each function gets a stack frame that holds local variables and arguments to the function. Calling functions pushes these frames and returns pop them.

Recursion and Loops

- The simplest recursive functions are basically loops. They call themselves once and have an argument that works at the iteration variable.
- The call to itself has to be conditional. Otherwise it is like an infinite loop, but you get a stack overflow when it runs out of memory.

The Power of Recursion

- Recursive functions are truly powerful when they call themselves more than once. Converting these types of function to loops requires significant reworking of the logic (at best).
- The stack is what enables this because with the stack the flow of control can go off in one direction, then return to the function and go off in another direction.
- These functions have "tree" call structures.

floodFill

- You are probably all familiar with the option in paint programs that allows you to "pour" in paint to fill a region of a picture. How would you do this with loops in a program?
- With recursion we just write a function that calls itself four times. Because of the memory of the stack, it can go "around corners".

Maze Problems and Graph Traversals

- By adding a bit of extra logic to a floodFill we can turn it into a function that searches through mazes. 2D mazes are a special example of the general category of graphs. A graph has vertices that are connected by edges.
- The key to these problems is that we want to leave behind "bread crumbs" to mark our path and pick them up as we backtrack.
- We can also can leave other markers that can help speed up our algorithms depending on what we are doing.

Divide and Conquer

- Another common usage of recursive functions is in divide and conquer algorithms. In these, we repeatedly break a problem down into smaller pieces until we get to a point where we can solve it easily. We then combine the partial solutions to find a full solution.
- Some extremely simple examples could be finding the sum of the elements of an array or the min of them.

String Parsing

- Another fun application of divide and conquer is string parsing for things like equations. Here we break the problem up around the lowest priority operator and recursively parse smaller sections if the formula, then deal with then as we move back up the stack.
- With just a little more work, this type of method can be turned into a very powerful tool.

Minute Essay

- Write code for a function that counts the number of different paths from a point in a maze to an exit assuming you can't cross back over your path. Do this using a maze that is a 2D array of characters.
- Assignment #6 is due today.