Alexander Memering CSSE221 Defoe

William Bismayer Capsule 2 Recursion Summary

Recursion is the process of performing an action that repeats itself before it finishes. In programming, recursive methods, like iterative methods (for and while loops), repeat a process until a desired result or end statement is reached. Usually anything that can be done with an iterator can be done with a recursive method, and vice versa; however what sets recursive methods apart is their behavior.

Most recursive methods work by taking an input size, making it smaller, and calling itself with the smaller input size. All recursive methods have an end condition: an 'if' statement that takes the smaller input size, returns a fixed statement or value, and stops the repetition. A recursive method without an end condition will repeat indefinitely in an infinite loop.

As an example, here's a recursive method that takes an integer input n, and returns the nth number in the Fibonacci sequence.

```
public static int fib(int n){
if(n<= 2){
      return 1;
}
return fib(n-1) + fib(n-2);
}
```

Recursive Helper methods are methods that are called within a recursive method, and process and make the input size smaller for the recursive method. Helper Methods are useful because they can improve the program's efficiency and make the structuring of the recursive method easier.

One problem you might encounter with recursion is that when a recursive method calls itself multiple times at the end of single loop, such as in the fib() method above, the number of repetitions grows astronomically and the runtime increases dramatically for large inputs. It is recommended that a helper method that collects the outputs for the values of n (probably with HashMaps) and returns those outputs for the other runs with those input sizes be used. This approach is called memoization.

Mutual recursion is the occurrence of multiple methods which call each other in a recursive manner.