Michael Rea

Great Ideas in Computer Science

The Usefulness of Recursion

Recursion is defined as “the repeated application of a repetitious procedure or definition”. This is helpful in computer science because computers can work incredibly quickly and don’t mind repetition. Recursion is basically the idea of taking a complex problem and attempting to take a single step toward solving it, which could then be repeated numerous times to eventually achieve a solution. A classic example of this would be finding the factorial of a number. For example,

4! is essentially finding the value of this product:

*4 * 3 * 2 * 1*

However, sometimes telling a computer to do this is not so easy. A recursive solution is the most straightforward way of solving this problem, because “4!” is essentially the same as 4 times “3!”. By making this step, we have essentially already solved the problem because we can follow this process again and again until we deconstruct the problem to a level that is more easily understood. In this case, we will continue to break down the factorial until we get to “1!” which we know is just 1. This highlights the key to recursion. A recursive solution must:

a) Make a step toward making a problem simpler, and
b) Eventually terminate with an established base case.

Some solutions may only achieve one of these cases. In that situation, a computer may run indefinitely trying to solve a problem. Recursion can sometimes take a little bit of extra thinking, because it is a unique way of deconstructing a problem. In some cases,

recursion may not necessarily be the best solution, in which case a programmer may just be making his/her job even more difficult. However, recursive solutions can be desirable because they are often extremely compact and elegant. Recursive programs tend to be significantly shorter than linear programs because there will be many calls of the same function. The only trade-off here is that particularly intensive recursive solutions can be tough on a computer because there may be hundreds of instances of functions, sitting suspended as others are evaluated, essentially creating a traffic jam of function calls as others are evaluated. Thankfully computers can keep track of this much better as humans. For example, have you ever tried to get a favor done from somebody? But in order to get them to do it, you must do them a favor which you cannot do? Thus, you must ask somebody else to help you out with *that* favor, and you could begin a long chain of IOU's and favors until you finally trace your way back to the initial problem. Sometimes this can get confusing to keep track of, whereas computers are incredibly smart in some simple ways. In conclusion, recursion can be tricky, but it is a unique way of problem solving that has very real uses in computer science and beyond.