CSE 101 Class Notes Today: Recursion vs. Iteration

October 16, 2006

On a different subject, here we explore the relationship between recursion and iteration, using as an example the simple problem of finding the maximum of an array. This case is a particular example of how, in general, correctness easier to show for recursive algorithms, while running time is easier to find for iterative ones.

Iterative

```
max <- A[1]
for i in 2..size(A)
    if A[i] > max
        max <- A[i]
return max
```

Time: Clearly $O(n)$ – we can verify this by counting the number of loop iterations, and by noting that each operation within the loop takes constant time.

Correctness: First, we need to find a helpful loop invariant. Applying the "cleverness" technique, we find that after loop k, `max` is the maximum of `A[1..k]`. We then need to prove (by induction) that this invariant is satisfied on every iteration of the loop. We also must show that the invariant's being satisfied on the last iteration implies that our algorithm is correct, i.e. that `max` is the maximum element in `A[1..n]`.

Recursive

```
max(A[1..n]) =
    if n == 1 return A[1]
    else
        m <- max(A[1..n-1])
        if   m > A[n]  return m
        else           return A[n]
```

Correctness: Easy – because we don't have to come up with a loop invariant. The shape of the inductive proof is present in the algorithm: line 2 is our base case, and lines 3-6 are our inductive case. We just need to run through this proof to show that the function `max` works.

Time: Harder – while all the basic steps (except line 4) take constant time, it's not nearly as obvious how many times the body executes (i.e. how many total recursive calls are made). Usually, to find a recursive algorithm's running time we have to formulate and solve a recurrence. In the example above:

- $T(1) = c$ (i.e. line 2)
- $T(n) = T(n-1) + d$ (i.e. line 4 plus lines 5-6)

Using induction, we can show that

$$T(n) = d(n-1) + c$$

for constants c and d, and therefore that the algorithm is $O(n)$.