

6.046 Recitation 1: Proving Correctness

Bill Thies, Fall 2004

Acknowledgement

Much of these notes is adapted from David Liben Nowell's notes for the same subject (thanks David!)

My contact information:

Bill Thies (last name rhymes with "peace")

thies@mit.edu

Stata 32-G890

617-253-6284

Office hours TBA, probably in this room

Announcements:

θ -notation to be covered in lecture next week

PS 1 out, due Mon 9/20

Today:

Admin

Correctness

- Recursive

- Iterative

Correctness of Recursive Algorithms

Simple algorithm for averaging two numbers, for illustrative purposes only

INPUT: int a, b: $0 \leq a \leq b$

OUTPUT: $(a+b)/2$

AVE(a,b)

 if a=b

 return a

 else

 return $0.5 + \text{AVE}(a, b-1)$

INDUCTION on n

1. Base case: f(n) is correct for n=0

2. Inductive step:

 if f(0)...f(n-1) is correct, then f(n) is correct [strong induction]

 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

 inductive hypothesis

(Don't need strong induction in this case.)

Correctness of AVE

1. Base: $a=b$

$$\text{AVE}(a,b) = \text{AVE}(a,a+0) = a =? (a+a)/2 = a \quad [\text{ok}]$$

$$b=a$$

2. Inductive step (on n)

If $\text{AVE}(a,a+n-1)$ is correct, then $\text{AVE}(a, a+n)$ is correct

$$\text{AVE}(a,b) = 0.5 + \text{AVE}(a, b-1)$$

$$b=a+n = 0.5 + \text{AVE}(a, a+n-1) \leftarrow \text{correct by ind. Hypothesis}$$

$$= 0.5 + (a+b-1) / 2$$

$$= (a+b)/2$$

To take away:

Whenever doing induction, important to say what variable or expression the induction is on.

It is not always a single variable you are doing induction on; could be a relationship between variables. In this case, induction on $b-a$ ($= n$). Other examples will come up, e.g., the number of elements of an array that are sorted.

Correctness of Iterative Algorithms

AVE(a, b)

avg \leftarrow a

for i = a to b-1

 avg \leftarrow avg + 0.5

return avg

How to prove right value is returned?

Loop invariant: relationship between variables in algorithm that holds at start of every loop iteration.

There are many such relationships. Important part is finding the ones that are useful for the algorithm.

For thinking about loop invariants, I recommend replacing for loops by while loops. It makes it clear where the loop boundaries are.

Doing this on above yields:

AVE(a,b)

avg \leftarrow a

i \leftarrow a

while (i \leq b-1)

 avg \leftarrow avg + 0.5

 i \leftarrow i+1

return avg

Procedure for Proving Correctness

1. Give PRE condition that holds before loop
2. Give POST condition that holds after loop
3. Guess loop invariant L
4. Prove L using induction
 - Base case: L holds on first iter
 - Inductive step: if L holds on i, then L holds on i+1
5. Prove that (L and "loop terminates") \Rightarrow POST
6. Prove that loop terminates

In case of AVE:

1. PRE: $avg=a, i=a$
2. POST: $avg = (a+b)/2$
3. L: $avg = (a+i)/2$
4. Base: $avg = (a+a)/2 = a$ [ok]
Ind: iter i: $avg = (a+i)/2$
iter i+1: $avg + 0.5 =? (a+i+1)/2$
 $avg = (a+i)/2$ [ok]
5. Loop exit: $i=b$, so $avg = (a+b)/2$
6. i starts at $a \leq b$, and increases every iteration until it hits b

Polynomial Evaluation

$$f(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$$

Q: Given x , evaluate $f(x) = \sum_{j=0}^n a_j x^j$

Naïve: $T(n) = \sum_{j=0}^n \Theta(j) = \Theta(n^2)$

Horner: $f(x) = a_0 + x(a_1 + x(a_2 + x(\dots x(a_n))))$

$$T(n) = \Theta(n)$$

Horner's Algorithm (Horner's Rule)

INPUT: $A[0 \dots n]$, x

OUTPUT: $\sum_{j=0}^n A[j] x^j = A[0] + A[1]x + A[2]x^2 + \dots + A[n]x^n$

HORNER($A[0 \dots n]$, x)

$val \leftarrow 0$

$i \leftarrow n$

 while ($i \geq 0$)

$val \leftarrow A[i] + x \cdot val$

$i \leftarrow i - 1$

 return val

Example:

<u>i</u>	<u>val</u>
n	0
n-1	$A[n]$
n-2	$A[n-1] + A[n]x$
n-3	$A[n-2] + A[n-1]x + A[n]x^2$

Proving Horner Correct

1. PRE: $val=0, i=n$

2. POST: $val = \sum_{j=0}^n A[j] x^j$

3. L:

- this is the tricky part

- hints:

find something “in between” PRE and POST that incorporates the loop counter, i .

at PRE, $val(i) = val(n) = 0$

at POST, $val(i) = val(-1) = \sum_{j=0}^n A[j] x^j$

look at example iterations to see what the pattern is

- start with this template, and we will fill in the ??:

L: $val(i) = \sum_{j=??}^n A[j] x^{j-??}$

look at example iterations for $i=n-3$. The coefficients involved are $A[n-2]$, $A[n-1]$ and $A[n]$. So if we are referencing $A[j]$, then j in this summation ranges from $n-2$ to n ; in general, this is $i+1$ to n :

L: $val(i) = \sum_{j=i+1}^n A[j] x^{j-??}$

Now we need the exponent of x . Look at the exponent for $A[n]$ for successive iterations of i . It is x^0, x^1, x^2 as i takes on the values $(n-1), (n-2)$, and $(n-3)$. The value of j at $A[n]$ is n . Thus, in general, we could guess that the exponent is $x^{j-(i+1)}$:

L: $val(i) = \sum_{j=i+1}^n A[j] x^{j-(i+1)}$

You might also see this by just inspecting the examples: the sum is being built from the “end” of the polynomial towards the bottom.

Remember, L only has to be a guess. If you can't see from inspection that it's correct, that's all the more reason that we are proving correctness in the first place.

Proving Horner Correct (cont'd)

We didn't cover this in recitation.

4. Prove L holds by induction:

Base: at start of loop $i=n$

L: $\text{val} = \sum_{j=n+1}^n \dots = \text{sum is empty,} = 0$

val is 0 in PRE, so it's [ok]

Ind. step: Show that if L holds for $\text{val}(i)$ then it holds for $\text{val}(i-1)$

in loop, val re-assigned:

$$\text{val} = A[i] + x \cdot \text{val}(i)$$

Then i re-assigned to $i-1$, so this value is carried to next iter:

$$\text{val}(i-1) = A[i] + x \cdot \text{val}(i)$$

By inductive hypothesis:

$$\text{val}(i-1) = A[i] + x \cdot \sum_{j=i+1}^n A[j] x^{j-(i+1)}$$

Renaming "i" to "i+1":

$$\text{val}(i) = A[i+1] + x \cdot \sum_{j=i+2}^n A[j] x^{j-(i+2)}$$

Add x^0 after $A[i+1]$, move x inside summation:

$$\text{val}(i) = A[i+1] x^0 + \sum_{j=i+2}^n A[j] x^{j-(i+1)}$$

Combine sum:

$$\text{val}(i) = \sum_{j=i+1}^n A[j] x^{j-(i+1)} \quad [\text{ok}]$$

5. At end of loop, $i=-1$.

$$\text{Evaluate } \text{val}(-1) = \sum_{j=-1+1}^n A[j] x^{j-(-1+1)} = \sum_{j=0}^n A[j] x^j$$

This is POST, so it proves correctness.

6. The loop terminates because i is initialized to n , then decremented on every loop iteration.