Algorithms Course Notes
Algorithm Correctness

Ian Parberry*

Fall 2001

Summary

- Confidence in algorithms from testing and correctness proof.
- Correctness of recursive algorithms proved directly by induction.
- Correctness of iterative algorithms proved using loop invariants and induction.
- Examples: Fibonacci numbers, maximum, multiplication

Correctness

How do we know that an algorithm works?

Modes of rhetoric (from ancient Greeks)

- Ethos
- Pathos
- Logos

Logical methods of checking correctness

- Testing
- Correctness proof

Testing vs. Correctness Proofs

Testing: try the algorithm on sample inputs

Correctness Proof: prove mathematically

Testing may not find obscure bugs.

Using tests alone can be dangerous.

Correctness proofs can also contain bugs: use a combination of testing and correctness proofs.

Correctness of Recursive Algorithms

To prove correctness of a recursive algorithm:

- Prove it by induction on the "size" of the problem being solved (e.g. size of array chunk, number of bits in an integer, etc.)
- Base of recursion is base of induction.
- Need to prove that recursive calls are given subproblems, that is, no infinite recursion (often trivial).
- Inductive step: assume that the recursive calls work correctly, and use this assumption to prove that the current call works correctly.

Recursive Fibonacci Numbers

Fibonacci numbers: $F_0 = 0$, $F_1 = 1$, and for all $n \geq 2$, $F_n = F_{n-2} + F_{n-1}$.

```
   function fib(n)
      comment return F_n
1.    if n ≤ 1 then return(n)
2.    else return(fib(n − 1)+fib(n − 2))
```

<u>Claim</u>: For all $n \geq 0$, fib(n) returns F_n.

Base: for $n = 0$, fib(n) returns 0 as claimed. For $n = 1$, fib(n) returns 1 as claimed.

Induction: Suppose that $n \geq 2$ and for all $0 \leq m < n$, fib(m) returns F_m.

RTP fib(n) returns F_n.

What does fib(n) return?

$$\text{fib}(n-1) + \text{fib}(n-2)$$

$$
\begin{aligned}
&= F_{n-1} + F_{n-2} \quad \text{(by ind. hyp.)} \\
&= F_n.
\end{aligned}
$$

Recursive Maximum

```
function maximum(n)
  comment Return max of A[1..n].
  if n ≤ 1 then return(A[1]) else
      return(max(maximum(n − 1),A[n]))
```

<u>Claim</u>: For all $n \geq 1$, maximum(n) returns $\max\{A[1], A[2], \ldots, A[n]\}$. Proof by induction on $n \geq 1$.

Base: for $n = 1$, maximum(n) returns $A[1]$ as claimed.

Induction: Suppose that $n \geq 1$ and maximum(n) returns $\max\{A[1], A[2], \ldots, A[n]\}$.

RTP maximum$(n + 1)$ returns

$$\max\{A[1], A[2], \ldots, A[n + 1]\}.$$

What does maximum$(n + 1)$ return?

$$
\begin{aligned}
& \max(\text{maximum}(n), A[n + 1]) \\
= \; & \max(\max\{A[1], A[2], \ldots, A[n]\}, A[n + 1]) \\
& \quad \text{(by ind. hyp.)} \\
= \; & \max\{A[1], A[2], \ldots, A[n + 1]\}.
\end{aligned}
$$

Recursive Multiplication

<u>Notation</u>: For $x \in \mathbb{R}$, $\lfloor x \rfloor$ is the largest integer not exceeding x.

```
function multiply(y, z)
  comment return the product yz
  if z = 0 then return(0) else
  if z is odd
    then return(multiply(2y, ⌊z/2⌋)+y)
    else return(multiply(2y, ⌊z/2⌋))
```

<u>Claim</u>: For all $y, z \geq 0$, multiply(y, z) returns yz. Proof by induction on $z \geq 0$.

Base: for $z = 0$, multiply(y, z) returns 0 as claimed.

Induction: Suppose that for $z \geq 0$, and for all $0 \leq q \leq z$, multiply(y, q) returns yq.

RTP multiply$(y, z + 1)$ returns $y(z + 1)$.

What does multiply$(y, z + 1)$ return?

There are two cases, depending on whether $z + 1$ is odd or even.

If $z + 1$ is odd, then multiply$(y, z + 1)$ returns

$$
\begin{aligned}
& \text{multiply}(2y, \lfloor (z + 1)/2 \rfloor) + y \\
= \; & 2y\lfloor (z + 1)/2 \rfloor + y \quad \text{(by ind. hyp.)} \\
= \; & 2y(z/2) + y \quad \text{(since } z \text{ is even)} \\
= \; & y(z + 1).
\end{aligned}
$$

If $z + 1$ is even, then multiply$(y, z + 1)$ returns

$$
\begin{aligned}
& \text{multiply}(2y, \lfloor (z + 1)/2 \rfloor) \\
= \; & 2y\lfloor (z + 1)/2 \rfloor \quad \text{(by ind. hyp.)} \\
= \; & 2y(z + 1)/2 \quad \text{(since } z \text{ is odd)} \\
= \; & y(z + 1).
\end{aligned}
$$

Correctness of Nonrecursive Algorithms

To prove correctness of an iterative algorithm:

- Analyse the algorithm one loop at a time, starting at the inner loop in case of nested loops.
- For each loop devise a *loop invariant* that remains true each time through the loop, and captures the "progress" made by the loop.
- Prove that the loop invariants hold.
- Use the loop invariants to prove that the algorithm terminates.
- Use the loop invariants to prove that the algorithm computes the correct result.

Notation

We will concentrate on one-loop algorithms.

The value in identifier x immediately after the ith iteration of the loop is denoted x_i ($i = 0$ means immediately before entering for the first time).

For example, x_6 denotes the value of identifier x after the 6th time around the loop.

Iterative Fibonacci Numbers

```
function fib(n)
    comment Return F_n
    if n = 0 then return(0) else
        a := 0; b := 1; i := 2
        while i ≤ n do
            c := a + b; a := b; b := c; i := i + 1
    return(b)
```

Claim: fib(n) returns F_n.

Facts About the Algorithm

$$\begin{aligned} i_0 &= 2 \\ i_{j+1} &= i_j + 1 \end{aligned}$$

$$\begin{aligned} a_0 &= 0 \\ a_{j+1} &= b_j \end{aligned}$$

$$\begin{aligned} b_0 &= 1 \\ b_{j+1} &= c_{j+1} \end{aligned}$$

$$c_{j+1} = a_j + b_j$$

The Loop Invariant

For all natural numbers $j \geq 0$, $i_j = j + 2$, $a_j = F_j$, and $b_j = F_{j+1}$.

The proof is by induction on j. The base, $j = 0$, is trivial, since $i_0 = 2$, $a_0 = 0 = F_0$, and $b_0 = 1 = F_1$.

Now suppose that $j \geq 0$, $i_j = j + 2$, $a_j = F_j$ and $b_j = F_{j+1}$.

RTP $i_{j+1} = j + 3$, $a_{j+1} = F_{j+1}$ and $b_{j+1} = F_{j+2}$.

$$\begin{aligned} i_{j+1} &= i_j + 1 \\ &= (j+2) + 1 \quad \text{(by ind. hyp.)} \\ &= j + 3 \end{aligned}$$

$$\begin{aligned} a_{j+1} &= b_j \\ &= F_{j+1} \quad \text{(by ind. hyp.)} \end{aligned}$$

$$\begin{aligned} b_{j+1} &= c_{j+1} \\ &= a_j + b_j \\ &= F_j + F_{j+1} \quad \text{(by ind. hyp.)} \\ &= F_{j+2}. \end{aligned}$$

Correctness Proof

Claim: The algorithm terminates with b containing F_n.

The claim is certainly true if $n = 0$. If $n > 0$, then we enter the while-loop.

Termination: Since $i_{j+1} = i_j + 1$, eventually i will equal $n + 1$ and the loop will terminate. Suppose this happens after t iterations. Since $i_t = n + 1$ and $i_t = t + 2$, we can conclude that $t = n - 1$.

Results: By the loop invariant, $b_t = F_{t+1} = F_n$.

Iterative Maximum

```
function maximum(A, n)
    comment Return max of A[1..n]
    m := A[1]; i := 2
    while i ≤ n do
        if A[i] > m then m := A[i]
        i := i + 1
    return(m)
```

Claim: maximum(A, n) returns

$$\max\{A[1], A[2], \ldots, A[n]\}.$$

Facts About the Algorithm

$$\begin{aligned} m_0 &= A[1] \\ m_{j+1} &= \max\{m_j, A[i_j]\} \end{aligned}$$

$$\begin{aligned} i_0 &= 2 \\ i_{j+1} &= i_j + 1 \end{aligned}$$

The Loop Invariant

Claim: For all natural numbers $j \geq 0$,

$$\begin{aligned} m_j &= \max\{A[1], A[2], \ldots, A[j+1]\} \\ i_j &= j + 2 \end{aligned}$$

The proof is by induction on j. The base, $j = 0$, is trivial, since $m_0 = A[1]$ and $i_0 = 2$.

Now suppose that $j \geq 0$, $i_j = j + 2$ and

$$m_j = \max\{A[1], A[2], \ldots, A[j+1]\},$$

RTP $i_{j+1} = j + 3$ and

$$m_{j+1} = \max\{A[1], A[2], \ldots, A[j+2]\}$$

$$\begin{aligned} i_{j+1} &= i_j + 1 \\ &= (j+2) + 1 \quad \text{(by ind. hyp.)} \\ &= j + 3 \end{aligned}$$

$$\begin{aligned} & m_{j+1} \\ =\ & \max\{m_j, A[i_j]\} \\ =\ & \max\{m_j, A[j+2]\} \quad \text{(by ind. hyp.)} \\ =\ & \max\{\max\{A[1], \ldots, A[j+1]\}, A[j+2]\} \\ & \quad \text{(by ind. hyp.)} \\ =\ & \max\{A[1], A[2], \ldots, A[j+2]\}. \end{aligned}$$

Correctness Proof

Claim: The algorithm terminates with m containing the maximum value in $A[1..n]$.

Termination: Since $i_{j+1} = i_j + 1$, eventually i will equal $n+1$ and the loop will terminate. Suppose this happens after t iterations. Since $i_t = t+2$, $t = n-1$.

Results: By the loop invariant,

$$\begin{aligned} m_t &= \max\{A[1], A[2], \ldots, A[t+1]\} \\ &= \max\{A[1], A[2], \ldots, A[n]\}. \end{aligned}$$

Iterative Multiplication

```
   function multiply(y, z)
      comment Return yz, where y, z ∈ ℕ
1.    x := 0;
2.    while z > 0 do
3.       if z is odd then x := x + y;
4.       y := 2y; z := ⌊z/2⌋;
5.    return(x)
```

Claim: if $y, z \in \mathbb{N}$, then multiply(y, z) returns the value yz. That is, when line 5 is executed, $x = yz$.

A Preliminary Result

Claim: For all $n \in \mathbb{N}$,

$$2\lfloor n/2 \rfloor + (n \bmod 2) = n.$$

Case 1. n is even. Then $\lfloor n/2 \rfloor = n/2$, $n \bmod 2 = 0$, and the result follows.

Case 2. n is odd. Then $\lfloor n/2 \rfloor = (n-1)/2$, $n \bmod 2 = 1$, and the result follows.

Facts About the Algorithm

Write the changes using arithmetic instead of logic.

From line 4 of the algorithm,

$$\begin{aligned} y_{j+1} &= 2y_j \\ z_{j+1} &= \lfloor z_j/2 \rfloor \end{aligned}$$

From lines 1,3 of the algorithm,

$$\begin{aligned} x_0 &= 0 \\ x_{j+1} &= x_j + y_j(z_j \bmod 2) \end{aligned}$$

The Loop Invariant

Loop invariant: a statement about the variables that remains true every time through the loop.

Claim: For all natural numbers $j \geq 0$,

$$\boxed{y_j z_j + x_j = y_0 z_0.}$$

The proof is by induction on j. The base, $j = 0$, is trivial, since then

$$\begin{aligned} y_j z_j + x_j &= y_0 z_0 + x_0 \\ &= y_0 z_0 \end{aligned}$$

Suppose that $j \geq 0$ and

$$y_j z_j + x_j = y_0 z_0.$$

We are required to prove that

$$y_{j+1} z_{j+1} + x_{j+1} = y_0 z_0.$$

By the Facts About the Algorithm

$$\begin{aligned} & y_{j+1} z_{j+1} + x_{j+1} \\ = \; & 2y_j \lfloor z_j/2 \rfloor + x_j + y_j(z_j \bmod 2) \\ = \; & y_j(2\lfloor z_j/2 \rfloor + (z_j \bmod 2)) + x_j \\ = \; & y_j z_j + x_j \quad \text{(by prelim. result)} \\ = \; & y_0 z_0 \quad \text{(by ind. hyp.)} \end{aligned}$$

Correctness Proof

Claim: The algorithm terminates with x containing the product of y and z.

Termination: on every iteration of the loop, the value of z is halved (rounding down if it is odd). Therefore there will be some time t at which $z_t = 0$. At this point the while-loop terminates.

Results: Suppose the loop terminates after t iterations, for some $t \geq 0$. By the loop invariant,

$$y_t z_t + x_t = y_0 z_0.$$

Since $z_t = 0$, we see that $x_t = y_0 z_0$. Therefore, the algorithm terminates with x containing the product of the initial values of y and z.

Assigned Reading

Problems on Algorithms: Chapter 5.