remind: A2 is due Friday at 3 p.m.

Office hours this week:          Danny: today 2:30--4:30, Friday 1:00--3:00
course TAs: tomorrow 2:00--6:00

CSC236 fall 2018

correct after & before

iterative (loopwise) correctness...

Danny Heap
heap@cs.toronto.edu / BA4270 (behind elevators)
http://www.teach.cs.toronto.edu/~heap/236/F18/
416-978-5899

Using Introduction to the Theory of Computation,
Chapter 2
Outline

iterative binary search

power

notes
correctness by design

draw pictures of before, during, after

pre: A sorted, comparable with $x$ $|A| = n > 0$

post: $0 \leq b \leq n$ and $A[0:b] < x \leq A[b:n-1]$

may be empty may be empty

---------------------------------------------------------------------------->
draw pictures...
before

?? < x <= ??

b = 0
e = n-1

middle

?? < x <= ??

< x

>= x

b
e

use pictures to develop our loop:

while b <= e:
    m = (b + e) // 2
    if A[m] < x:
        b = m + 1  # want A[b-1] < x
    else:
        e = m - 1  # want A[e+1] >= x
    return b

end

< x

>= x

e b
“derive” conditions from pictures

need notation for mutation at the end of the ith iteration of the loop value of e is e_i
value of b is b_i

Precondition: A is sorted nondecreasing, \(|A| = n > 0, b = 0, e = n - 1, \in \mathbb{N}\)
Postcondition: \(0 \leq b \leq n\) AND all([j < x for j in A[0:b]]) AND all([k >= x for k in A[b:n]])

idea: loop invariant should yield the postcondition after last iteration, should also be true!

define P(i): at the end of the ith iteration (if it occurs): \(0 \leq b_i \leq e_i + 1 \leq n\) AND b_i, e_i + 1 \in \mathbb{N}
AND all([j < x for j in A[0:b_i]]) AND all([k >= x for k in A[e_i + 1:n]])

Proof, by simple induction, that \(\forall i \in \mathbb{N}, P(i)\)

base case, i=0: by precondition \(0 = b_0 \leq n = e_0 + 1\), also \(b_0 = 0\) and \(e_0 = n - 1 \in \mathbb{N}\). Both slices are empty, and any universally quantified claim is true of the empty set. So \(P(0)\) follows.

inductive step: Let \(i \in \mathbb{N}\). Assume \(P(i)\). Will show that \(P(i+1)\) follows. If there is an \((i+1)\)th iteration, (by code) \(b_i \leq e_i\). Also by code \(m = (b_i + e_i) / 2\), thus \(m \in \mathbb{N}\) (sum of natural numbers, integer divided), and \(b_i = 2b_i / 2 = m \leq e_i + 1 \leq n\) by IH, also by IH all([k >= x for k in A[e_i + 1:n]]) ==>
all([k >= x for k in A[e_i + 1 + 1:n]]) also by IH \(e_i + 1 \in \mathbb{N}\), so \(e_{i+1} + 1 \in \mathbb{N}\)
\(A[b_{i+1} - 1] = A[m] < x\) (by code), also A is sorted so all([j < x for j in A[0:b_{i+1}]]), Also, \(m \in \mathbb{N}\), so \(m + 1 = b_{i+1} \in \mathbb{N}\). By IH \(0 \leq b_i \leq m + 1 = b_{i+1} \leq e_{i+1} + 1 = e_{i+1}\)
“derive” conditions from pictures

need notation for mutation

\text{case } A[m] \geq x:\n
\text{By code, } b_{i+1} = b_i \text{ and } e_{i+1} = m-1.\n\text{By IH } 0 \leq b_i = b_{i+1} \text{ and } b_i = b_{i+1} \in \mathbb{N}, \text{ and } \text{all}([j<x \text{ for } j \text{ in } A[0:b_i]]) \implies \text{all}([j<x \text{ for } j \text{ in } A[0:b_{i+1}]])\n\text{Also } A[e_{i+1}+1] \geq x \text{ (by case) and since } A \text{ is sorted all}([k \geq x \text{ for } k \text{ in } A[e_{i+1}+1:n]])\n\text{Since } m \in \mathbb{N}, \text{ then } m-1+1 \in \mathbb{N}, \text{ hence } e_{i+1} + 1 \in \mathbb{N}. \text{ Also } m \leq e_i < n \implies e_{i+1} + 1 \leq n.
So $P(i+1)$ follows in both cases.

At this point, separate termination and assume loop terminates after some iteration. Let $f$ be the index of the final iteration.

By loop invariant $b_f - 1 \leq e_f$ # by $P(f)$  
by code $b_f > e_f$

so $b_f = e_f + 1$, so I can replace all references to $e$ in loop invariant:

$P(f): b_f \in \mathbb{N} \text{ AND } 0 \leq b_f \leq n \text{ AND all([j<x for j in A[0:b_f]]) AND all([k>=x for k in A[b_f:n]])}$

this is the Postcondition. Thus precondition+execution+termination imply postcondition (partial correctness).

It remains to prove termination...
do we have termination?

Many beginner get this wrong. Reasoning that the loop condition is "eventually" violated is extremely difficult... there be dragons! Don't *ever* do that! Instead find an expression based on the values in the loop that is (a) a natural number and (b) is strictly decreasing with each loop iteration. This yields a decreasing sequence of natural numbers. Such sequences are finite, since they have a smallest element. The index of the smallest element is = index of the last loop iteration. So the loop terminates.

\(<e_i + 1 - b_i>\) is a good candidate. First, is it a natural number? By \(P(i)\) we know that \(e_i + 1 \geq b_i\), so \(e_i+1-b_i\geq0\ AND \ b_i, \ e_i+1\in\mathbb{N}\), so their difference is \(\in\mathbb{Z}\), hence (non-negative) \(\in\mathbb{N}\).

Recall, earlier, that if there is an \((i+1)\)th iteration then \(b_i \leq m \leq e_i\).

Suppose there is an \((i+1)\)th iteration. There are two cases to consider:

**case A\[m\]<x:** Then \(e_{i+1} = e_i\ AND \ b_{i+1} = m+1\). Then
\[
\begin{align*}
  e_{i+1}+1-b_{i+1} &= e_i + 1 - (m+1) = e_i - m < e_i + 1 - m \leq e_i+1-b_i \quad \# \ since \ b_i \leq m
\end{align*}
\]

**case A\[m\]>=x:** Then \(e_{i+1} = m-1\ AND \ b_{i+1} = b_i\). Then
\[
\begin{align*}
  e_{i+1}+1-b_{i+1} &= m-1+1-b_i = m - b_i < m + 1 - b_i \leq e_i + 1 - b_i \quad \# \ since \ m \leq e_i
\end{align*}
\]

Thus we have exhibited a decreasing sequence of natural numbers linked to loop iterations. The last element of this sequence has the index of the last loop iteration, so the loop terminates.
correctness by discovery

integer power

def power(x, y):
    z = 1
    m = 0
    while m < y:
        z = z * x
        m = m + 1
    return z

P(i): after the ith iteration of the loop (if it occurs)

    z_i = x^{m_i} AND m_i \in \mathbb{N} AND m_i <= y

▶ precondition? \( x \in \mathbb{R}, \ y \in \mathbb{N} \)
▶ postcondition? \( z = x^y \)
▶ notation for mutation Let \( m_i \) be \( m \) after the ith iteration, and \( z_i \) be \( z \) after ith iteration.
partial correctness
precondition+execution+termination imply postcondition
a loop invariant helps get us closer

Prove \( \forall i \in \mathbb{N}, P(i) \) using simple induction on \( i \).

**base case:** \( m_0 = 0, z_0 = 1 \) (by initialization), \( x^0 = x^{m_0} = 1 = z_0 \). Also \( y \in \mathbb{N} \) by precondition, and \( m_0 = 0 \in \mathbb{N} \). AND \( m_0 = 0 \leq y \), since \( y \in \mathbb{N} \). So \( P(0) \) follows.

**inductive step:** Let \( i \in \mathbb{N} \) and assume \( P(i) \). Show that \( P(i+1) \) follows. If there is an \((i+1)\)th loop iteration.

Then \( m_{i+1} = m_i + 1 \) # by code
Also \( z_{i+1} = z_i \cdot x = x^{m_i} \cdot x \) (by IH) = \( x^{m_i+1} = x^{m_{i+1}} \)
Also \( m_i \in \mathbb{N} \) (by IH), so \( m_{i+1} = m_i + 1 \in \mathbb{N} \) (closure under addition)
Also \( m_i < y \Rightarrow m_i + 1 \leq y \) (both integers) \( \Rightarrow m_{i+1} \leq y \).

So \( P(i+1) \) follows.

**partial correctness:** show that \( \text{pre}+\text{execution}+\text{termination} \implies \text{postcondition} \)

If the loop terminates after, say, iteration \( f \), then the following must be true:

\[
\begin{align*}
m_f &\geq y & \# \text{ by loop condition} \\
m_f &\leq y & \# \text{ by } P(f)
\end{align*}
\]

Thus \( m_f = y \). By \( P(f) \) we have \( z_f = x^{m_f} = x^y \Rightarrow \text{postcondition} \).
Many beginners mess this up by trying to prove the loop condition is "eventually" violated. Don't *ever* do this. Instead devise a sequence of natural numbers whose elements are associated with loop iterations and which is strictly decreasing. A strictly decreasing sequence in \( \mathbb{N} \) is finite, and hence has a last (smallest) element.

Try the sequence \( <y - m_i> \). By the precondition \( y \in \mathbb{N} \) and by the loop invariant \( P(i), m_i \in \mathbb{N} \) and \( m_i \leq y \), so \( y - m_i \) is an integer, and \( m_i \leq y \iff y - m_i \geq 0 \), so each element of the sequence is \( \in \mathbb{N} \).

It remains to show that the sequence is strictly decreasing. Suppose there is an \( (i+1) \)th iteration of the loop. Then \( y - m_{i+1} = y - (m_i + 1) < y - m_i \), so the sequence is strictly decreasing.

Thus, the loop terminates.
that vexing invariant...

```python
>>> colour_list_0 = ["r", "b", "b", "g"]
>>> green, red = 0, 4
>>> colour_list_0[:green] + colour_list_0[red:]
[]
>>> # loop iterates somewhat...
>>> colour_list_2 = ["g", "b", "b", "r"]
>>> green, red = 1, 3
>>> colour_list_2[:green] + colour_list_2[red:]
['g', 'r']
>>> # same colours as before, possibly permuted...
>>> colour_list_0[:green] + colour_list_0[red:]
['r', 'g']
```