reminder: 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

same binary search, except now iterative (uses a loop):

draw pictures of before, during, after

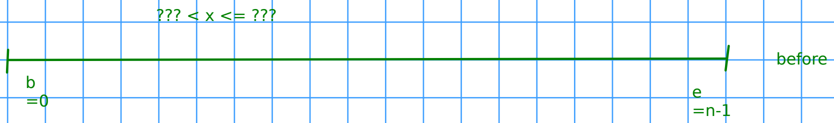
pre: A sorted, comparable with x |A| = n > 0

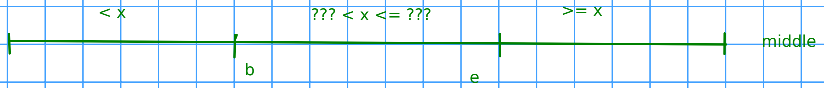
post: $0 \le b \le n$ and A[0:b] $< x \le$ A[b:n-1]

may be empty

may be empty

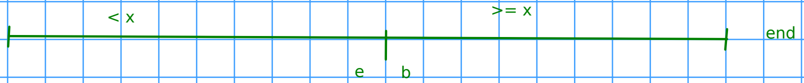
---> draw pictures...





use pictures to develop our loop:

```
while b<=e:
     m = (b+e)//2
     if A[m] < x:A
        b = m+1   # want A[b-1]< x
     else: # A[m] >= x
        e = m-1 # want A[e+1]>=x
  return 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 \N
Postcondition: 0<=b<=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<=b_i<=e_i+1<=n AND b_i, e_i+1 \in \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 \N, P(i)

base case, i=0: by precondition 0=b_0 <= n = e_0+1, also b_0=0 and e_0=n-1 \in \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 \N. Assume P(i). Will show that P(i+1) follows. If there is an (i+1)th iteration, (by code)
b_i<=e_i. Also by code m = (b_i+e_i)//2, thus m\in\N (sum of natural numbers, integer divided),
and b_i = 2b_i//2<= m <= 2e_i//2 = e_i.

case A[m] < x:
By code b_{i+1} = m+1 and e_{i+1} = e_i
m <= e_i+1 = e_{i+1}+1 <= 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 \N, so e_{i+1}+1 \in \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 \N, so m+1=b_{i+1}\in \N. By IH 0<= b_i <= m+1 = b_{i+1} <= e_{i+1}+1=e_i+1

“derive” conditions from pictures

need notation for mutation

case A[m]>=x:

By code, b_{i+1} = b_i and e_{i+1} = m-1.
By IH 0<=b_i=b{i+1} and b_i=b_{i+1}\in \N, and all([j<x for j in A[0:b_i]]) ==> all([j<x for j in A[0:b_{i+1}]])
Also A[e{i+1}+1] >= x (by case) and since A is sorted all([k>=x for k in A[e_{i+1}+1:n]])
Since m\in \N, then m-1+1 \in \N, hence e_{i+1}+1\in \N. Also m<= e_i < n => e_{i+1} + 1 <= n.

partial correctness

precondition+execution+termination imply postcondition

loop invariant helps get us closer

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 <= 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 \N AND 0<= b_f <= n 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 >= b_i, so e_i+1-b_i>=0 AND b_i, e_i+1\in\N, so their difference is \in\Z, hence (non-negative) \in \N.

Recall, earlier, that if there is an (i+1)th iteration then b_i <= m <= 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
e_{i+1}+1-b_{i+1} = e_i + 1 - (m+1) = e_i - m < e_i + 1 - m <= e_i+1-b_i   # since b_i<=m

case A[m]>=x: Then e_{i+1} = m-1 AND b_{i+1} = b_i. Then
e_{i+1}+1-b_{i+1} = m-1+1-b_i = m - b_i < m + 1 - b_i <= e_i + 1 - b_i   # since m <= e_i

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 \N AND m_i <= y

- precondition? x\in\R. y\in\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 \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 \N by precondition, and
m_0 = 0 \in \N. AND m_0 = 0 <= y, since y \in \N. So P(0) follows.

inductive step: Let i \in \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 * x = x^{m_i} * x (by IH) = x^{m_i+1} = x^{m_{i+1}}
Also m_i \in \N (by IH), so m_{i+1} = m_i + 1 \in \N (closure under addition)
Also m_i < y => m_i + 1 <= y (both integers) => m_{i+1} <= y.

So P(i+1) follows.

partial correctness: show that pre+execution+termination ==> postcondition

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

m_f>=y # by loop condition
m_f <=y # by P(f)

Thus m_f = y. By P(f) we have z_f = x^{m_f} = x^y ==> postcondition.

prove termination

associate a decreasing sequence in $\mathbb{N}$ with loop iterations

it helps to add claims to the loop invariant

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 \N is finite, and hence has a last (smallest) element.

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

```
>>> 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']
```