CSC236 fall 2012

recursion, induction, correctness

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~heap/236/F12/
416-978-5899

Using Introduction to the Theory of Computation, Chapter 2

Outline

binary search

recursive binary search

```
def recBinSearch(x, A, b, e) :
  if b == e :
    if x <= A[b] :
      return b
    else :
      return e + 1
  else :
    m = (b + e) // 2 # midpoint
    if x <= A[m] :
      return recBinSearch(x, A, b, m)
    else :
      return recBinSearch(x, A, m+1, e)
```

conditions, pre- and post-

- x and elements of A are comparable
- e and b are valid indices, $b \leq e$
- $A[b..e]$ is sorted non-decreasing

RecBinSearch(x, A, b, e) terminates and returns index p

- $b \leq p \leq e + 1$
- $b < p \Rightarrow A[p-1] < x$
- $p \leq e \Rightarrow x \leq A[p]$

(except for boundaries, returns p so that $A[p-1] < x \leq A[p]$)

precondition $\Rightarrow$ termination and postcondition

Proof: induction on $n = e - b + 1$

Base case, $n = 1$: Terminates because there are no loops or further calls, returns $x \leq A[b = p] \Leftrightarrow p = b = e$ is returned. $x > A[b = p - 1] \Leftrightarrow p = b + 1$ returned, so postcondition satisfied. Notice that the choice forces if-and-only-if.

Induction step: Assume $n > 1$ and that the postcondition is satisfied for inputs of size $1 \leq k < n$ that satisfy the precondition. Call RecBinSearch(A,x,b,e) when $n = e - b + 1 > 1$. Since $b < e$ in this case, the test on line 1 fails, and line 7 executes. Exercise: $b \leq m < e$ in this case. There are two cases, according to whether $x \leq A[m]$ or $x > A[m]$.

Case 1: $x \leq A[m]$

- Show that IH applies to RBS(x,A,b,m)
- Translate the postcondition to RBS(x,A,b,m)

- Show that RBS(x,A,b,e) satisfies postcondition

Case 2: $x > A[m]$

- Show that IH applies to RBS(x,A,m+1,e)
- Translate postcondition to RBS(x,A,m+1,e)

- Show that RBS(x,A,b,e)

what could go wrong?

- $m = \lceil \frac{e+b}{2.0} \rceil$
- $x < A[m]$
- $\ldots$
- Either prove correct, or find a counter-example

recursive and iterative

mergesort

```
 MergeSort(A,b,e):
 1. if b == e:  return
 2. m = (b + e) / 2  # integer division
 3. MergeSort(A,b,m)
 4. MergeSort(A,m+1,e)
    # merge sorted A[b..m] and A[m+1..e] back into A[b..e]
 5. for i = b,...,e:  B[i] = A[i]
 6. c = b
 7. d = m+1
 8. for i = b,...,e:
 9.     if d > e or (c <= m and B[c] < B[d]):
10.         A[i] = B[c]
11.         c = c + 1
        else:  # d <= e and (c > m or B[c] >= B[d])
12.         A[i] = B[d]
13.         d = d + 1
```

conditions, pre- and post-

- b and e are nature numbers, $0 \leq b \leq e < \text{len}(A)$.
- elements of A are comparable

- $A'[b..e]$ contains the same elements as $A[b..e]$, but sorted in non-decreasing order (use notation A' for A after calling MergeSort(A,b,e)). All other elements of A' are unchanged.

Proof of correctness of MergeSort(A,b,e)

by induction on $n = e - b + 1$ for all arrays of size n,
(precondition+execution)⇒(termination+postcondition)

Base case, $1 = e - b + 1$: Assume MergeSort(A,b,e) is called with $len(A) = 1$ preconditions satisfied. Then $0 \leq e \leq b \leq 0$, so $e == b$, and the algorithm terminates with a (trivially) sorted A', satisfying the precondition.

Induction step: Assume $n \in \mathbb{N}$, $n > 1$, and for all natural numbers k, $1 \leq k < n$, that MergeSort on all arrays of size k that satisfy the precondition and run will terminate and satisfy the postcondition. Assume MergSort(A,b,e) is executed and $n = e - b + 1$.

The test on line 1 fails, and m is set to $(b + e)//2$, strictly less than e (exercise).

Does the IH apply to MergeSort(A,b,m) and MergeSort(A,m+1,e)? Translate the IH into postconditions for MergeSort(A,b,m) and MergeSort(A,m+1,e).

Now we need **iterative correctness** for the merge...

iterative correctness

partial correctness plus termination

- Preconditions plus termination imply the postcondition. Probably needs a loop invariant

- termination — construct a decreasing sequence in $\mathbb{N}$.