Correctness

• How do you know if it BinarySearch works correctly?
  • First we need to precisely state what the algorithm does through the precondition and postcondition
  • The precondition states what may be assumed to be true initially:
    • $a \leq b + 1$ and $A[a..b]$ is a sorted array
    • $\text{found} = \text{BinarySearch}(A, a, b, x)$;
    • Post: $\text{found} = x \in A[a..b]$ and $A$ is unchanged

Outline

• How to specify what an algorithm does
• How to prove the correctness of a recursive algorithm
• How to prove the correctness of an iterative algorithm

Binary Search

• **Problem:** Determine whether a number $x$ is present in a sorted array $A[a..b]$
• **Binary Search Solution:**
  – Compare the middle element $mid$ to $x$
  – If $x = mid$, stop
  – If $x < mid$, throw away larger elements
  – If $x > mid$, throw away smaller elements
  – If there is no element left, $x$ is not in the array

Binary Search Code

```java
BinarySearch(A, a, b, x)
1  if a > b then
2    return false
3  else
4    mid ← ⌊(a+b)/2⌋
5    if x = A[mid] then
6      return true
7    if x < A[mid] then
8      return BinarySearch(A, a, mid-1, x)
9    else
10   return BinarySearch(A, mid+1, b, x)
```

Running time calculations:
On each iteration, more than half of elements are removed. Program will run while
$n(0.5)^k > 1$
$k < \log_2 n$

Correctness of Recursive Algorithm

• Proof must take us from the precondition to the postcondition.
  • **Base case:** $n = b-a+1 = 0$
    • The array is empty, so $a = b + 1$
    • The test $a > b$ succeeds and the algorithm correctly returns false
  • **Inductive step:** $n = b-a+1 > 0$
    • **Inductive hypothesis:** Assume
      BinarySearch(A,a',b',x) returns the correct value for all $j$ such that $0 \leq j \leq n-1$ where $j = b' - a' + 1$. 

Recitation 1: Correctness 7

The algorithm first calculates \( \text{mid} = \lfloor (a+b)/2 \rfloor \), thus \( a \leq \text{mid} \leq b \).

If \( x = A[\text{mid}] \), clearly \( x \in A[a..b] \) and the algorithm correctly returns true.

If \( x < A[\text{mid}] \), since \( A \) is sorted (by the precondition), \( x \) is in \( A[a..b] \) if and only if it is in \( A[a..\text{mid}-1] \). By the inductive hypothesis, \( \text{BinarySearch}(A,a,\text{mid}-1,x) \) will return the correct value since \( 0 \leq (\text{mid}-1)-a+1 \leq n-1 \).

The case \( x > A[\text{mid}] \) is similar.

We have shown that the postcondition holds if the precondition holds and \( \text{BinarySearch} \) is called.

Recitation 1: Correctness 8

Summing an Array

- Problem: Given an array of numbers \( A[a..b] \) of size \( n = b - a + 1 \geq 0 \), compute their sum.

```plaintext
// Pre: a \leq b + 1
1  i ← a, sum ← 0
2  while i ≠ b + 1 do     // exit condition, called guard G
3     sum ← sum + A[i]
4     i ← i + 1
// Post: sum = \sum_{j=a}^{b} A[j]
```

Recitation 1: Correctness 9

Correctness of Iterative Algorithms

- The key step in the proof is the invention of a condition called the loop invariant, which is supposed to be true at the beginning of an iteration and remains true at the beginning of the next iteration.
- The steps required to prove the correctness of an iterative algorithms is as follows:
  1. Guess a condition \( I \)
  2. Prove by induction that \( I \) is a loop invariant
  3. Prove that \( I \land \neg G \Rightarrow \text{Postcondition} \)
  4. Prove that the loop is guaranteed to terminate

Recitation 1: Correctness 10

- In the example, we know that when the algorithm terminates with \( i = b+1 \), the following condition must hold: \( \text{Postcondition} \)
- Use as invariant. Show that at the beginning of the \( k \)-th loop, the condition holds:
  - Base Case: \( k = 1 \)
    - Initialized to \( i = a \) and \( \text{sum} = 0 \). Therefore \( \sum_{j=a}^{1} A[j] = 0 \)
  - Inductive hypothesis: Assume \( \sum_{j=a}^{i-1} A[j] \) at the start of the loop’s \( k \)-th execution

Recitation 1: Correctness 11

- Let \( \text{sum}' \) and \( i' \) be the values of the variables \( \text{sum} \) and \( i \) at the beginning of the \( (k+1) \)-st iteration.
- In the \( k \)-th iteration, the variables were changed as follows:
  - \( \text{sum}' = \text{sum} + A[i] \)
  - \( i' = i + 1 \)
- Using the inductive hypothesis, we have
  \[
  \text{sum}' = \text{sum} + A[i] = \sum_{j=a}^{i+1} A[j] = A[i] + \sum_{j=a}^{i} A[j] = \sum_{j=a}^{i+1} A[j]
  \]

Recitation 1: Correctness 12

- We have proven the loop invariant \( I \).
- Now we must show: \( I \land \neg G \Rightarrow \text{Postcondition} \)
  - We have \( \neg G \Rightarrow i = b + 1 \). Substituting into the invariant:
    \[
    \text{sum} = \sum_{j=a}^{b+1} A[j] = \sum_{j=a}^{b} A[j] + A[b+1] = \text{Postcondition}
    \]
- Remains to show that \( G \) will eventually be false.
  - Note that \( i \) is monotonically increasing since it is incremented inside the loop and not modified elsewhere.
  - From the precondition, \( i \) is initialized to \( a \leq b + 1 \).
Summary

• How to specify an algorithm:
  – Precondition
  – Postcondition

• How to prove correctness of recursive algorithm:
  – Induction

• How to prove correctness of iterative algorithm
  – Prove a loop invariant
  – Show that the invariant and terminating condition implies the postcondition
  – Shows that the loop is guaranteed to terminate.