Proving correctness

Proving correctness

- Proof based on loop invariants
  - Loop invariant: An assertion which is satisfied before each iteration of a loop
  - At termination, the loop invariant provides important property that is used to show correctness
- Steps of proof:
  - Initialization (similar to induction base)
  - Maintenance (similar to induction proof)
  - Termination

More on the steps

- Initialization: Show loop invariant is true before (or at start of) the first execution of a loop
- Maintenance: Show that if the loop invariant is true before an iteration of a loop, it is true before the next iteration
- Termination: When the loop terminates, the invariant gives us an important property that helps show the algorithm is correct

Example: Finding maximum

```
Findmax(A, n)
    maximum = A[0];
    for (i = 1; i < n; i++)
        if (A[i] > maximum)
            maximum= A[i]
    return maximum
```

- What is a loop invariant for this code?

Proof of correctness

- Loop invariant for Findmax(*A*):

  "Before the i^{th} iteration (for $i = 1, \ldots, n$) of the for loop maximum = $max\{A[0], A[1], \ldots, A[i-1]\}$"

Initialization

- We need to show loop invariant is true at the start of the execution of the *for* loop
- Line 1 sets maximum=*A*[0]
- So the loop invariant is satisfied at the start of the *for* loop.

Maintenance

- Assume that at the start of the i^{th} iteration of the *for* loop

  maximum = $max\{A[j] \mid j = 0, \ldots, i - 1\}$
- We will show that before the $(i + 1)^{th}$ iteration, maximum $= max\{A[j] \mid j = 0, \ldots, i\}$
- The code computes

  maximum=max(maximum, $A[i]) = max(max\{A[j] \mid j = 0, \ldots, i - 1\}, A[i]) = max\{A[j] \mid j = 0, \ldots, i\}$

Termination

- The loop terminates when $i = n$
- So $maximum = max\{A[j] | j=0,\ldots,n - 1\}$

Example: Insertion sort

```
Insertion_Sort(A)
{
for (i = 1; i < n; i++)
    for (j = i; j >= 1 and a[j] < a[j-1]; j--)
            swap a[j] and a[j-1]
}
```

→ Loop invariant?

Proof of correctness

- Loop invariant for INSERTION_SORT(*A*):

At the start of the i^{th} iteration of the for loop

- *A*[0.. i-1] contains the elements originally in *A*[1.. i -1]
- *A*[0.. i-1] is sorted

Initialization

- We need to show loop invariant is true at the start of the execution of the *for* loop
- After line 1 sets $i = 1$ and before it compares i to n (= length[A]), we have:
  - Subarray A[0.. 1 - 1]= A[0] contains the original element in A[0]
  - A[0] is sorted.
- So the loop invariant is satisfied

Maintenance

- If the loop invariant is true before this execution of a loop, it is true before the next execution
- Assume that at the start of the i^{th} iteration of the for loop, *A*[0.. i-1] contains the elements originally in

  A[0.. i-1] and *A*[0.. i-1] is sorted

Maintenance

- We will show that the loop invariant is maintained before the (i + 1)th iteration.
  - We will show that at the start of the (i + 1)th iteration of the for loop, A[0.. i] contains the elements originally in A[0.. i] and A[0.. i] is sorted

Maintenance

- The body of the loop keeps moving to the left until the proper position for $A[j]$ is found and then inserts $A[j]$ into the subarray $A[1..\ j\ ]$.
- Thus, the sub array $A[1..\ i]$ contains the elements originally in $A[1..\ i]$ and $A[1..\ i]$ is sorted

Termination

- i = length[A] .
- The array A[0.. length[A]-1] contains the elements originally in

A[0 .. length[A] - 1] and

A[0 .. length[A] - 1] is SORTED!

Loop invariants

```
1.    sum =0;
2.    for (i = 0; i < n; i++)
3.        sum = sum + A[i];
```

- What is a loop invariant for this code?