Computer Science 331

Solutions to Selected Tutorial #3 Questions

Questions

Consider the following code segment:

```
// Pre-condition: A is a non-null array of integers
//   that contains at least one element
// Post-condition:  max is the largest element in the array A
n = A.length;
max = A[0];
for i from 1 to n-1 do
  if (A[i] > max)
    max = A[i];
end do
```

1. Write a loop invariant for this loop. Explain how you would prove that this loop invariant is correct.

   Solution: The following is a loop invariant for the loop in the example:

   $$I(j): \ i = j + 1; \ max \text{ is the largest of the elements } A[k] \text{ for } 0 \leq k < i; \ 1 \leq i \leq n \ .$$

   To prove that this loop invariant is correct, we would need to show the following.

   - The loop invariant is satisfied when $j = 0$, i.e., $I(0)$ holds before the loop body executes for the first time.
   - If $I(j)$ is satisfied after the jth execution of the loop body and there is a $(j+1)$st execution, then $I(j+1)$ is satisfied after the $(j+1)$st execution, for each integer $j \geq 0$. In other words
   - If there is a jth execution but not a $(j+1)$st execution then $I(j)$ implies the post-condition (again, for each integer $j \geq 0$).

Explanation: Recall from the notes that a loop invariant is an assertion $I(j)$ that is true *immediately* after the loop body has been executed j times, *if* the loop body is actually *executed* j times, for all $j \geq 0$. In the case of a for-loop, you should consider this being placed in the code immediately after the loop index is initialized or updated but before the termination test.

The first part in this example is to express the loop index i as a function of j, the number of iterations. The initial value of i is 1, and this occurs when $j = 0$, i.e., the loop has iterated 0 times. Thus, we have $i = j + 1$.

The second part is to give upper and lower bounds on the loop index i. Before the loop iterates for the first time we have already argued that $i = 1$, and when the loop terminates we have $i = n$. Thus, $1 \leq i \leq n$.

Finally, we need to provide one or more logical statements succinctly describing the effect of the loop on the loop's variables. The purpose of the example, computing the maximum value in the array, comes from the post-condition provided. The progress made towards this goal after an iteration of the loop comes from the fact that, after an iteration of the loop, we know that max is the maximum of all the elements searched to that point. Initially, i.e., before any iterations of the loop, max is initialized with $A[0]$, and is therefore the maximum of the single element $A[0]$. After the first iteration, max is the maximum of $A[0]$ and $A[1]$, and after the third iteration, max is the maximum of $A[0], A[1]$, and $A[2]$. In general, after the jth iteration, we have that max is the maximum of $A[0], \ldots, A[i-1]$. Notice that this is true for any iteration, including when $j = 0$ (before any iterations) and when $i = n$ (after the last iteration).

Putting all this together, we have that $I(j)$ as defined above is a loop invariant for the loop in the example.

To prove that this loop invariant is in fact correct, we proceed as follows.

- We first show that $I(0)$ is satisfied before the first execution of the loop. Initially we have have $max = A[0]$ and $i = 1$. Thus, the first part of the loop invariant ($i = j + 1$) and the third ($1 \leq i \leq n$) hold. When $j = 0$, the second part becomes "max is the largest of the elements $A[k]$ for $0 \leq k < 1$," which also holds because the only value of k in this range is 0.
- Next, we show that if $I(j)$ is satisfied after the jth execution and there is a $j + 1$st execution, then $I(j + 1)$ is satisfied after the $j + 1$st execution, for each integer $j \geq 0$. First, note that the first and third conditions of the loop invariant, $i = j + 1$ and $1 \leq i \leq n$ hold after the jth execution of the loop for any j. To prove that the second condition of $I(j+1)$ holds after the jth execution of the loop, let max_j be the value of max after executing the loop body j times and max_{j+1} be the value of max after the $j + 1$st execution. The assumption that $I(j)$ is correct implies that max_j is the largest of the array elements $A[k]$ for $0 \leq k < j + 1$ (recall that $i = j + 1$). To show that

$I(j+1)$ holds after the $j+1$st execution, we need to show that after executing the loop body we will have max_{j+1} is the largest of $0 \leq k < j+2$. We already know that $max_j \geq A[k]$ for $0 \leq k < j+1$ by the assumption that $I(j)$ holds. When the loop body is executed, we check whether $max_j > A[j+1]$, and, if so, max_{j+1} is assigned the value of $A[j+1]$; otherwise, max is unchanged and we have $max_{j+1} = max_j$. Thus, after completion of the $j+1$st iteration, we have that max is the largest of $A[0], \ldots, A[j+1]$ as required.

- Finally, we show that if there is a jth execution but not a $j+1$st execution then $I(j)$ implies the postcondition (again, for each integer $j \geq 0$). The loop terminates when $i = n$. Thus, if $I(j)$ is true when $i = n$, the loop invariant implies that max is the largest of $A[0], \ldots, A[n-1]$, and this is precisely what is stated in the post-condition.

Notice that the first two parts above constitute a proof by induction of the following claim:

Theorem 0.1. *The loop invariant $I(j)$ is true for all $j \geq 0$ for which the loop iterates j times.*

The case $j = 0$ is the base case, and the second parts completes the inductive proof, using $I(j)$ as the induction hypothesis.

2. Explain briefly how you would use this loop invariant to prove partial correctness for this example.

   Solution: We need to show that:

   - if the pre-condition is true, then the loop invariant $I(j)$ is true for $j = 0$, i.e., before any iterations of the loop,
   - $I(j)$ implies the post-condition whenever the loop body is executed exactly j times.

   The second of these was established in the previous question. The pre-condition specifies that A is a non-null array of integers containing at least one element. Thus, setting max to $A[0]$ will terminate correctly, and when i is initialized to 1 by the loop we have that $I(0)$ holds.

3. Prove that this loop terminates by giving a loop variant for it. Use the loop variant to give an upper bound on the number of times that the loop will iterate.

   Solution: We claim that $f(n,i) = n - i$ is a loop variant for this loop. To prove this, we note that $f(n,i)$ is an integer-valued function (because n and i are both integers), and show that $f(n,i)$ satisfies the remaining two properties of a loop variant:

   - $f(n,i) = n - i$ decreases after every iteration of the loop because i increases and n remains constant.
   - $f(n,i) \leq 0$ when $i \geq n$, and the for-loop terminates when $i = n$.

Explanation: To find a correct loop variant, we need to construct a function involving the loop's variables that decreases after each iteration of the loop body and implies the termination of the loop when ≤ 0. In this case, the loop variant will be a function of the length of the array n and the loop index i, as these are the only variables involved in deciding the loop's termination. The loop terminates when $i = n$, so the function must be ≤ 0 whenever $i \geq n$ and > 0 whenever $i < n$ (note that i increases after each execution of the loop body). Putting these together, we see that $f(n, i) = n - i$ satisfies the requirements.