Proving Termination and Analyzing the Running Time of a Simple Algorithm with a While Loop

Solutions for a Suggested Exercise

This exercise considered the following computational problem.

Sum of Array Elements

Precondition: An integer array `A` with some positive length n is given as input.

Postcondition: The value

$$\sum_{i=0}^{n-1} \mathtt{A}[i]$$

is returned as output.

It also concerned the following algorithm.

```
integer arraySum ( integer[] A ) {
1. integer sum := A[0]
2. integer i := 0
3. while (i < A.length − 1) {
4.   i := i + 1
5.   sum := sum + A[i]
   }
6. return sum
}
```

1. You were first asked to state a ***bound function*** for the `while` loop in the above algorithm and to prove that your answer is correct.

   Solution: The function $\mathtt{A.length} - \mathtt{i} - 1 = n - \mathtt{i} - 1$ is a bound function for the `while` loop in this algorithm.

To see that this is the case, it suffices to show that this function satisfies all the required properties included in the definition of a "bound function for a `while` loop":

- Since `A` is an input integer array, and `i` is an integer variable whose value has defined before the array has been reached, this is a well-defined, total, integer-valued function of the algorithm's inputs and local variables (which have all been defined with values given before the array has been reached, if the function depends on them).
- The array `A` and its length, n, are not changed when the body of this `while` loop is executed. On the other hand, the value of `i` is increased by one when the step at line 4 is executed — so that the value of this function is decreased by at least one (indeed, by exactly one) every time the body of this `while` loop is executed.
- Finally, if the value of this function is less than or equal to zero then $\texttt{A.length} - \texttt{i} - 1 \leq 0$ — so that $\texttt{i} \geq \texttt{A.length} - 1$, and loop test (at line 3) would fail if it was checked.

Thus this function satisfies all the properties of a "bound function for a `while` loop", as required.

2. You were then asked to prove that this algorithm terminates whenever it is executed when the precondition for the "Sum of Array Elements" problem is initially satisfied. Recall that it now follows (since its partial correctness has also been established) that this algorithm correctly solves this problem.

   Solution: To begin, consider any execution of the `while` loop of this algorithm that is part of an execution of the algorithm starting with the precondition for the "Sum of Array Problems" satisfied.

   - Since the body of the `while` loop consists of only two simple assignment statements — at lines 4 and 5 — every execution of the body of this `while` loop ends.
   - As shown above, this `while` loop has a bound function.

   It now follows by "Loop Theorem #2" that an execution of this `while` loop, that is part of an execution of this algorithm with its problem's precondition satisfied, does eventually terminate.

   Now consider any execution of the algorithm that begins with the precondition for the "Sum of Array Elements" problem satisfied.

   This execution of the algorithm includes the execution of two simple assignment statements (each of which will halt when executed) before the `while` loop is reached and executed. As noted above, this execution of the `while` loop eventually ends — and the execution of the algorithm then terminates after the execution of the step at line 6.

Thus every execution of this algorithm, beginning with the precondition for the "Sum of Array Elements" satisfied, terminates, as claimed.

3. You were also asked to use this to state the number of steps executed by this algorithm, when it is executed with an input array `A` with positive length n.

   Solution: Since `i` $= 0$ when the `while` loop is reached, the initial value of the bound function is `A.length`$-1 = n-1$, and this is an upper bound for the number of executions of the body of the `while` loop included in an execution of the loop.

   However, something more can be said for this particular `while` loop: Note that an execution of the body of the `while` loop always decreases the value of the bound function by *exactly* one. Furthermore, a comparison of the loop test at line 3 and the bound function suffices to confirm that the loop test fails *if and only if* the value of the bound function is less than or equal to zero. Furthermore, the body of the loop does not contain any statements that could cause the loop to terminate before the bottom of the loop body is reached and the loop test is checked again. Under these circumstances[1], it can be argued that the number of executions of the loop body is *equal* to the initial value of the bound function — in this case, $n - 1$.

   Since there *must* be one more execution of the loop test — that fails — in order for an execution of this `while` loop to terminate, it also follows that there are exactly n executions of the loop test, at line 3, included in an execution of this `while` loop.

   Since the loop test is the simple test "`i` $<$ `A.length` $- 1$", — which does not call subroutines involving the execution of other algorithms — $T_{\text{test}}(\, j\,) = 1$ for every integer j such that $1 \leq j \leq n$.[2] It follows that the total number of steps included in all executions of the loop test is

   $$\sum_{j=1}^{n} T_{\textit{test}}(\, j\,) = \sum_{j=1}^{n} 1 = n.$$

   Similarly, the loop body consists of a pair of simple assignment statements, at lines 4 and 5,so that $T_{\textit{body}}(\, j\,) = 2$ for every integer j such that $1 \leq j \leq n - 1$. It follows that the total number of steps, included in all executions of the loop body as part of an execution of this `while` loop (again, if the precondition of the problem was satisfied when the execution of the algorithm began) is

   $$\sum_{j=1}^{n-1} T_{\textit{body}}(\, j\,) = \sum_{j=1}^{n} 2 = 2n - 2.$$

   The total number of steps included in an execution of this loop is therefore

   $$n + (2n - 2) = 3n - 2.$$

[1] and *only* under these circumstances...

[2] The terminology and notation introduced in the notes for this reading are now being used here.

Now consider an execution of the algorithm when the precondition for the "Sum of Array Elements" problem is satisfied. There are *two* statements (at lines 1 and 2) that are executed before the `while` loop is reached and executed, and *one* statement (at line 6) that is reached and executed after the execution of the loop ends, before the execution of the algorithm ends.

It follows that the number of steps used by this algorithm, if the precondition for its problem is satisfied when execution begins and the input array `A` has positive length n, is

$$2 + (3n - 2) + 1 = 3n + 1.$$

Note: It will generally ***not*** be possible to compute the number of steps executed *exactly* when more complicated algorithms are considered. Instead, the techniques introduced in these readings will be used to find reasonably accurate *upper bounds* for this.