Homework 2 Solutions

February 21, 2010

Problem 1a (3 points)

Precondition:

$N > 1$ is a natural number. A is an array of N elements. Each element is either red or blue.

Postcondition:

A contains the same number of red and blue elements, and all the red elements come before all blue elements.

Problem 1b (4 points)

Denote by i_k and j_k the values of i and j after the k^{th} iteration. The loop invariant is

1. If $j_k > 1$, then $A[1 \ldots j_k - 1]$ contains only red elements
2. If $i_k > j_k$, then $A[j_k \ldots i_k - 1]$ contains only blue elements.

We prove this by induction. The base case is the first iteration, or $k = 1$. In the case that $A[1]$ is red, then $i_1 = 2$ and $j_1 = 2$, so to prove the LI we just need to show that $A[1]$ is red. The swap operation does not affect the array since $i_0 = j_0$. Therefore, the value of $A[1]$ does not change and remains red. In the case that $A[1]$ is blue, then $i_1 = 2$ and $j_1 = 1$, so to prove the LI we need to show that $A[1]$ is blue. Since the if condition fails, A is not affected and $A[1]$ remains blue.

For the general case, we will show that the loop invariants holds after $k + 1$ iterations. First consider the case that $A[i_k] = red$. The only changes to A in iteration $k + 1$ are that the values of $A[i_k]$ and $A[j_k]$ are swapped. Moreover, $j_{k+1} = j_k + 1$ and $i_{k+1} = i_k + 1$. We prove the two parts of the LI separately:

1. By the IH, $A[1 \ldots j_k - 1]$ contains only red elements. Also, $A[j_k]$ contains a red value after the swap. Therefore, $A[1 \ldots j_{k+1} - 1] = A[1 \ldots j_k]$ contains only red values.

2. Since this part is only applicable if $i_{k+1} > j_{k+1}$, we can assume that $i_k > j_k$, and apply the IH to get that $A[j_k \dots i_k - 1]$ contains blue values prior to this iteration. After the swap, $A[i_k]$ gets a blue value, while $A[j_k + 1 \dots i_k - 1]$ does not change. Therefore, the elements in $A[j_{k+1} \dots i_{k+1} - 1] = A[j_k + 1 \dots i_k]$ are all blue.

Now we consider the case that $A[i_k]$ is blue. Then the array is not changed during this iteration, and $j_{k+1} = j_k$ and $i_{k+1} = i_k + 1$. Again, we prove the two cases separately:

1. Since this part is only applicable if $j_{k+1} > 1$, we can assume that $j_k > 1$ and apply the IH to get that $A[1 \dots j_k - 1]$ contains only red elements. Since A is not changed in this iteration and $j_{k+1} = j_k$, the elements of $A[1 \dots j_{k+1} - 1]$ are all red as needed.

2. If $i_k > j_k$, we apply the IH to get that $A[j_k \dots i_k - 1]$ contains only blue. Since we have assumed in this case that $A[i_k]$ is also blue, it follows that $A[j_{k+1} \dots i_{k+1} - 1]$ contains only blue. If $i_k = j_k$, then we need to only show that $A[i_k]$ is blue, we just follows from our assumption in this case.

Problem 1c (3 points)

Consider the sequence of values $N - i_k$ for $k > 0$. It is a decreasing sequence of natural numbers because N does not change and $i_{k+1} > i_k$. Hence, it must become less than zero at some point. Then the condition $i \leq N$ will fail and the loop will terminate.

Problem 2 (5 points)

Let z_k be the value of z after the k^{th} iteration. The value of z is only changed in one line, when it is decremented by one, and hence $z_{k+1} < z_k$. Since z is initially a natural number, and the sequence $\{z_k\}$ is decreasing, by Theorem 2.5 it is finite. Therefore the loop will terminate.

(Note to students: the variables x and y are only there to confuse you, since they do not have any role in the termination of the loop).

Problem 3a (3 points)

Precondition:

A is a non-empty array of integers, with N elements. Also, f and l are integers and $1 \leq f \leq l \leq N$.

Postcondition:

The return value is $\min\{A[i] | f \leq i \leq l\}$.

Problem 3b (7 points)

The proof is by strong induction. The induction hypothesis $P(k)$ is if $k = l-f+1$ and the pre-conditions to $MIN(A, f, l)$ are satisfied, then the call terminates and the return value is trivially the minimum of $A[f \ldots l]$.

Since the pre-conditions stipulate that $f \leq l$, the base case is the smallest value $l - f + 1$ can take on, which is 1. In this case, $f = l$, and the program terminates at line 3 with a return value $A[f]$ which satisfies the post-conditions.

For the general case, we will show that the IH holds for general k given that it holds for all values smaller than k. In order to apply the IH to the first recursive call, we have to show that $m - f + 1 < l - f + 1$ and that the pre-conditions are satisfied. The pre-conditions are satisfied as long as $1 \leq f \leq m \leq N$. Similarly, in order to apply the IH to the second call, we have to show $l - m < l - f + 1$ and that $1 \leq m + 1 \leq l \leq N$. All these facts follow algebraically from the fact that, if $f \neq l$, then $f \leq (f + l) \text{ div } 2 < l$. We can therefore apply the IH to the two recursive calls to conclude that they terminate, a contains the smallest value in $A[f \ldots m]$, and b contains the smallest value in $A[m + 1 \ldots l]$.

Because there are no loops in the program and the recursive calls terminate, the program must terminate. Moreover, the minimum value of $A[f \ldots l]$ must lie in either the sub-array $A[f \ldots m]$ or in $A[m + 1 \ldots l]$. The program returns the minimum of the minimums in these two sub-arrays, therefore the returned value is the minimum of $A[f \ldots m]$ and the post-conditions are satisfied.