CS240: Homework 5

Sahit Mandala

October 23, 2015

Problem 1

0th: $[5, 8, 2, 6, 1, 4, 7, 3]$,
1st: $[5, 2, 6, 1, 4, 7, 3, 8]$
2st: $[2, 5, 1, 4, 6, 3, 7, 8]$
3st: $[2, 1, 4, 5, 3, 6, 7, 8]$
4st: $[1, 2, 4, 3, 5, 6, 7, 8]$
5st: $[1, 2, 3, 4, 5, 6, 7, 8]$
6st: $[1, 2, 3, 4, 5, 6, 7, 8]$

Problem 2

Here, we define $P(x) :=$ on the mth iteration of the inner loop, $(i), (ii), (iii), (iv), (v)$ of the inner loop invariants all hold.

Say we are given $A, n \in \mathbb{N}$. We will use induction to show $P(x)$ holds on every iteration of the loop:

Base case P(0):

On the 0th iteration, the loop has not executed once, so we would like to check that m, i, k, A satisfy the loop invariant prior to loop execution. First, on line 1, $m = n$; furthermore, on line (2), we see that $m > 1$ (otherwise, the loop on line (2) would never be executed, but was assumed that we are executing line (5)) Notice that on line 3 and 4 respectively, $k = 0$ and $i = 0$. Clearly, $0 \leq i \leq 0 \leq m - 1$ and $0 \leq k \leq i$. Furthermore, the statement (iii) is also true since $A[0, ..., 0] = A[0]$ and $A[0, ..., (k-1) = -1] = []$; trivially, every element in $A[0]$ is $\geq$ every element in the empty array []. Also, the singleton array $A[0]$ is sorted, so (iv) holds. Finally, since A is currently unchanged, it is clearly a permutation of the original array input (namely, the identity permutation).

Inductive Step: Assume that $P(x)$ for some $x \in \mathbb{N}$. Then:

$0 \leq i_x \leq m_x - 1$;$0 \leq k \leq i_x$; All elements from $A[k_x, .., i_x]$ are $\geq$ than any element from $A_x[0, .., (k_x - 1)]$; All elements from $A_x[k_x, .., i_x]$ are sorted; A_x is a permutation of the original input array. We use the subscript x to indicate these are values after the xth iteration of the loop.

We want to show $P(x + 1)$ holds, specifically that (i)-(v) holds for $i_{x+1}, m_{x+1}, k_{x+1}, A_{x+1}$. If the $x + 1th$ does not occur, then we are done, so assume this iteration occurs.

At the beginning of the loop, $i_{x+1} = i_x, m_{x+1} = m_x, k_{x+1} = k_x, A_{x+1} = A_x$, since those were the results after the last iteration. Since we assume the loop is being executed, the conditional $i_{x+1} = i_x < m_{x+1} - 1$ on line (5) holds. Consider 2 cases (noting that, at line (6), $i_{x+1} = i_x$):

Case 1: If $A_{x+1}[i_x] > A_{x+1}[i_x + 1]$, then we swap these entries. So $A_{x+1}[i_x] = A_x[i_x + 1]$ and $A_{x+1}[i_x + 1] = A_x[i_x]$ after (7) and then $k_{x+1} = i_{x+1} + 1 = i_x + 1$ after (8). At (9), we increment i, so $i_{x+1} = i_x + 1$, completing the iteration.

Since we know $0 \leq i_x \ i_{x+1} = i_x < m_{x+1} - 1$ on line 5, we see that $0 \leq i_{x+1} = i_x + 1 \leq m - 1$ after the iteration. Furthermore, since $k_{x+1} = i_x + 1 = i_{x+1}$ after the iteration (note $i_x \geq 0$), then $0 \leq k_{x+1} \leq i_{x+1}$.

We now want to show (iii). We know that all elements $A_x[k_x, ..., i_x]$ are $\geq$ than elements in $A_x[0, ..., (k_x - 1)]$ by the inductive hypothesis. Consider $A_{x+1}[k_{x+1}, ..., i_{x+1}] = A_{x+1}[i_x + 1, ..., i_x + 1] = A_{x+1}[i_x + 1]$ and $A_{x+1}[0, ..., (k_{x+1} - 1)] = A_{x+1}[0, ..., i_x]$. We know that $A_x[i_x + 1] = A_{x+1}[i_x] < A_x[i_x] = A_{x+1}[i_x + 1]$ from the assumption and swap step. Since $A_{x+1}[i_x + 1] = A_x[i_x]$ is $\geq$ all elements in $A_x[0, ..., (k_x - 1)] = A_{x+1}[0, ..., i_x - 1]$ (since these $k_x - 1$ entries were unchanged in this iteration), it follows that $A_{x+1}[i_x + 1] = A_{x+1}[k_{x+1}, ..., i_{x+1}]$ is $\geq$ all elements in $A_{x+1}[0, ..., i_x - 1, i_x] = A_{x+1}[0, .., k_{x+1} - 1]$, implying (iii).

To show (iv), note that $A_{x+1}[k_{x+1}, ..., i_{x+1}] = A_{x+1}[i_x + 1, ..., i_x + 1] = A_{x+1}[i_x + 1]$, so we see that this singleton list is trivially sorted.

To show (v), recall that A_x was a permutation of the original input array. Also note that A_{x+1} has the same entries at every index as A_x except at i_x, i_{x+1}. At these indices, we swapped the values of $A_x$at $i_{x+1} = i_x + 1, i_x + 1$. That is, we permuted these 2 entries from A_x to A_{x+1}. So A_{x+1} is a permutation of A_x, and because the composition of permutations are permutations, A_{x+1} must be a permutation of the original input array, as expected.

Case 2: If $A_{x+1}[i_x] \leq A_{x+1}[i_x + 1]$, then the if statement is entirely skipped, so there are no changes to A, k between lines (6)-(8). At (9), we increment i, so $i_{x+1} = i_x + 1$, completing the iteration.

Since we know $0 \leq i_x \ i_{x+1} = i_x < m_{x+1} - 1$ on line 5, we see that $0 \leq i_{x+1} = i_x + 1 \leq m - 1$ after the iteration. Since $k_{x+1} = k_x \leq i_x$, then $k_{x+1} = k_x \leq i_x + 1 = i_{x+1}$.

Since $A_{x+1}[i_x] \leq A_{x+1}[i_x + 1]$

We know that all elements $A_x[k_x, ..., i_x]$ are $\geq$ than elements in $A_x[0, ..., (k_x - 1)]$ by the inductive hypothesis. Since $A_{x+1} = A_x$, note that $A_{x+1}[i_{x+1}] = A_x[i_x + 1]$ is $\geq$ all elements in $A_x[0, ..., (k_x - 1)] = A_{x+1}[0, ..., (k_{x+1} - 1)]$. So all elements in $A_{x+1}[k_{x+1}, ..., i_{x+1}]$ are $\geq$ then all elements in $A_{x+1}[0, ..., (k_{x+1} - 1)]$

For (iv), we also know that $A_x[k_x, ..., i_x] = A_{x+1}[k_{x+1}, ..., i_{x+1} - 1]$ is sorted by the inductive hypothesis. We also know that $A_{x+1}[i_x] = A_{x+1}[i_{x+1} - 1] \leq A_{x+1}[i_x + 1] = A_{x+1}[i_{x+1}]$. So clearly, the array $A_{x+1}[k_{x+1}, ..., i_{x+1}]$ is sorted since $A_{x+1}[k_{x+1}, ..., i_{x+1} - 1]$ is sorted and $A_{x+1}[i_{x+1}]$ is the larger than or equal to the largest element of $A_{x+1}[k_{x+1}, ..., i_{x+1} - 1]$, making the whole subarray sorted.

Finally, since $A_x = A_{x+1}$ and (v) holds for A_x, we know that (v) holds for A_{x+1}.

Overall, we have shown the inner loop invariant conditions hold.

Problem 3

Let some $A, n \in \mathbb{N}$

P(x) := After the xth iteration of the loop at (2), (i),(ii),(iii) of the outer loop invariants hold.

Base Case P(0):

On this 0th iteration, we see that at line 1, $m_0 = n$ is set. Since this is the only line executed, we readily see that $0 \leq m_0 \leq n$. Further more, the subarray $A[m_0, ..., n-1] = A[n, n-1]$ is the empty array. The empty array satisfies both (ii) and (iii) since there are no elements in the array to consider. Hence, we have shown P(0).

Inductive Step: Assume that $P(x)$ for some $x \in \mathbb{N}$. Then:

$0 \leq m_x \leq n$

All elements from $A_x[m_x, .., n - 1]$ are in sorted order

All elements from $A_x[m_x, .., n - 1]$ are $\geq$ elements in $A_x[0, ..., m_x - 1]$

We want to show $P(x + 1)$ holds, specifically that (i)-(v) hold for m_{x+1}, A_{x+1}. If the $x + 1th$ does not occur, then we are done, so assume this iteration occurs and completes. Note that $m_{x+1} = m_x, A_{x+1} = A_x$ at the beginning of the loop.

During this iteration, $i_{x+1} = 0, k_{x+1} = 0$ are set at lines 3,4. Then the while loop at (5) iterates. We assumed the whole program and thus this inner while loop should complete, say in y iterations. By the inner loop invariant, after the while loop completes, $0 \leq i_{x+1} \leq m_{x+1} - 1 = m_x - 1$ (noting that m_{x+1} is unchanged in the inner loop), $0 \leq k_{x+1} \leq i_{x+1}$, all elements from $A_{x+1}[k_{x+1}, .., i_{x+1}]$ are $\geq$ than any element from $A_{x+1}[0, .., (k_{x+1} - 1)]$, and all elements from $A_{x+1}[k_{x+1}, .., i_{x+1}]$ are sorted. Since we assumed the inner while loop terminated, we know the conditional at (5) must have been false, so $i_{x+1} \geq m_{x+1} - 1 = m_x - 1$. I would also note that the subarrays $A_x[m_x, ..., (n-1)] = A_{x+1}[m_{x+1}, ..., (n-1)]$ since $m_x = m_{x+1}$ and $i_{x+1} < m_{x+1} - 1$ on the inner loop, so the entries $A_{x+1}[m_{x+1}, ..., (n-1)]$ are never accessed and thus never changed. Finally, at (10), $m_{x+1} \leftarrow k_{x+1}$ is set.

To show (i), first of all, notice that $i_{x+1} \leq m_x - 1$ and $i_{x+1} \geq m_x - 1$, which implies $i_{x+1} = m_x - 1$. We will also note that $m_x \leq n$ by the inductive hypothesis, so $i_{x+1} = m_x - 1 < n$, and furthermore, $0 \leq k_{x+1} \leq i_{x+1} < n$. Since $m_{x+1} = k_{x+1}$, we see that $0 \leq m_{x+1} \leq n$.

To show (iii), first recall $A_x[m_x, ..., (n-1)] = A_{x+1}[m_x, ..., (n-1)]$ and all elements of $A_x[m_x, ..., (n-1)]$ are $\geq$ elements of $A_x[0, ..., m_x - 1]$. The elements of A_{x+1} and A_x are a permutation of the original A, and since $A_x[m_x, ..., (n-1)] = A_{x+1}[m_x, ..., (n-1)]$, we can infer that all the elements in $A_x[0, ..., m_x - 1]$ are also in $A_{x+1}[0, ..., m_x - 1] = A_{x+1}[0, ..., m_x - 1]$ (though potentially permuted. This means that all elements of $A_{x+1}[m_x, ..., (n-1)] = A_x[m_x, ..., (n-1)]$ are $\geq$ elements of $A_{x+1}[0, ..., m_x - 1]$. Now consider the subarray $A_{x+1}[k_{x+1}, ..., i_{x+1}]$, which we know is $\geq$ the elements of $A_{x+1}[0, ..., k_{x+1}]$ by the inner loop invariant. We also noted that $m_{x+1} = k_{x+1}$ and $i_{x+1} = m_x - 1$, so $A_{x+1}[k_{x+1}, ..., i_{x+1}] = A_{x+1}[m_{x+1}, ..., m_x - 1]$. Clearly, all elements of $A_{x+1}[m_{x+1}, ..., (n-1)]$ are $\geq$ elements of $A_{x+1}[0, ..., k_{x+1}] = A_{x+1}[0, ..., m_{x+1}] \subseteq A_{x+1}[0, ..., m_{x+1} - 1]$, and we just showed that this is true for $A_{x+1}[m_{x+1}, ..., m_x - 1]$ as well. So the elements of $A_{x+1}[m_{x+1}, ..., (n-1)]$ must be $\geq$ the elements of $A_{x+1}[0, ..., m_{x+1}]$, thus proving (iii)

To show (ii), first note that $A_x[m_x, ..., (n-1)] = A_{x+1}[m_x, ..., (n-1)]$ and $A_x[m_x, ..., (n-1)]$ is sorted by the inductive hypothesis, $A_{x+1}[m_x, ..., (n-1)]$ is also sorted. We also know that $A_{x+1}[k_{x+1}, ..., i_{x+1}] = A_{x+1}[m_{x+1}, ..., m_x - 1]$ is sorted. Furthermore, from (iii) which we just proved, we know that all elements of $A_{x+1}[m_x, ..., (n-1)]$ are greater than all the elements of $A_{x+1}[m_{x+1}, ..., m_x - 1]$, so $A_{x+1}[m_x - 1] \leq A_{x+1}[m_x]$ and thus all the elements in the array $A_{x+1}[m_{x+1}, ..., (n-1)]$ are in sorted order, thus proving (ii).

Overall, we have shown the outer loop invariant conditions hold.

Problem 4

Partial correctness:

Let some $A, n \in \mathbb{N}$ be given. Since n is the length of the array, $A[0, ..., n-1]$ is the entire array. Assume that the program terminates on this input. This implies that the while loop at (2) must have exited, which means that $m \leq 1$. Say that the outer loop terminated on the kth iteration, for some $k \in \mathbb{N}$. Then, by the outer loop invariant, $0 \leq m$. So $0 \leq m \leq 1$. Since m is an integer, there are only 2 possible cases:

Case 1: Suppose $m = 0$. Then by the outer loop invariant, all the elements from $A[m, ..., n-1]$=$A[0, ..., n-1]$ are in sorted order. But this is the entire array, which implies that the entire array A is now sorted.

Case 2: Suppose $m = 1$. Then by the outer loop invariant, all the elements from $A[m, ..., n-1]$=$A[1, ..., n-1]$ are in sorted order (if n=0 or n =1, then $A[1, ..., n-1]$ is just empty). Note that by the outer loop invariant, all the elements in $A[1, ..., n-1]$ are $\geq$ the elements in $A[m, ..., m-1] = A[0]$. But this implies that $A[0] \leq A[1]$, which means that the array $A[0, n-1]$ is also sorted.

Overall, in either case, the outer loop terminates with a sorted list, and thus the program does terminate on the correct output.

Termination:

Suppose we are given a valid input, say some $A, n \in \mathbb{N}$. To show termination, we need to show that both inner loops terminate across all inputs. First, consider the inner loop. On iteration 0, $i_0 = 0$, which is set on line (4). On every iteration of the loop at (5), i is incremented by 1, so on the kth iteration $i_k = 0 + 1 + 1... + 1 = k$. Also note that, throughout the loop, m is unchanged and by the inner loop invariant, that $0 \leq i \leq m - 1$. So on the $m - 1^{th}$

iteration, $i_{m-1} = m - 1$. The loop conditional $i < m - 1$ would be false, causing the inner loop to terminate. Now we consider the outer loop at (2). Notice that at the 0th iteration of this loop, $m_0 = n$. If $n = 0$ or $n = 1$, the loop conditional fails, so the loop would terminate and we are done. Otherwise, on some x^{th} iteration of the outer loop, we know that from the inner loop invariant, $0 \leq k_x \leq i_x \leq m_x - 1 = m_{x-1} - 1$ after the inner loop executes, noting that $m_x = m_{x-1}$ from the previous iteration at this point. Then m_x is set to k_x. so $0 \leq m_x \leq m_{x-1} - 1$, so the value of m_{x-1} is decremented by atleast 1 after every iteration. Because the outer loop terminates when $m \leq 1$, after a finite number of iteration (at most $n - 1$ iterations since m starts at $m_0 = n$), m will eventually reach the lower bound of 1. Thus, the outer loop will also terminate and thus the entire program terminates on any input A,n.