Emma Steinman
CS 271 Project 1
December 16, 2017

1.b Before iteration i of the for loop, if v is in the set, the value v is in A[i..n].

1.c Initialization
Before the first iteration, $i = 1$, of the for loop, if v is in the set, v is in A[i..n]. A[i..n] is the entire set so it makes sense that if v is in the set, it is in A[i..n] so the loop invariant is true before the first iteration.
Maintenence
Assume the loop invariant is true before some iteration i. The loop invariant states that if v is in the set, v is in A[i..n]. If the function has not returned a value yet, it means it has not reached v in the set. There are two cases in the loop. The first is that $A[i] = v$. In this case, the function would return and the loop invariant would say the value v is in A[i..n] and it was at A[i] which holds true. The second case is that A[i] is not equal to v. In this case, i is incremented and the loop invariant will say, if v is in the set, v is in A[i+1..n]. Since v was not in A[i] or any previous places (since the function has not returned yet) then if it is in the set the value of v must be in the remainder of the set (A[i+1..n]) so the loop invariant holds true.

1.d Termination
The loop can end for two reasons.
Case 1 (return i)
If $A[i] = v$, search will return i and therefore the loop will end. In this case, the loop invariant says: if v is in A, it will be in A[i..n]. Since the function returned because $A[i] = v$, we know this is true because A[i] is in A[i..n].
Case 2 (v is not in A)
If search completes the entire for loop it is because A[i] was never equal to v; v is not in the set A. At the end of the loop, i increments one more time so $i = n+1$. The loop invariant will then say, After the for loop terminates, v will be in A[n+1..n]. Since this set doesn't exist, the statement is not false, therefore the loop invariant still holds.

1.e The best case time complexity is $\Theta(1)$. The best case would be the first item in the set is the value v, so the loop takes a constant amount of time to find it and return its index. The worst case time complexity is $\Theta(n)$. The worst case would be value v is not in the set, so the function has to complete the loop n times to check each item. The average time complexity is $\frac{\sum_{i=1}^{n+1} \Theta(i)}{n+1} = \frac{\Theta((n+1)*(n+2)/2)}{(n+1)} = \Theta(n)$

2.a Before each iteration j of the inner for loop, A[j..j+1] contains the elements previously in A[j..j+1] but in sorted order.
Initialization
Before the first iteration, $j = n$, A[n..n+1] contains the elements previously in A[n..n+1] but in sorted order. This set does not exist so it is not false, therefore it holds.
Maintenance
Assume the loop invariant is true before some iteration j. The loop invariant then states A[j..j+1] contains the elements previously in A[j..j+1] in sorted order. During iteration j, if A[j-1] is greater than A[j], they are swapped. Thus at the end of the iteration, A[j..j+1] will be the same elements but in sorted order, as the loop invariant states. At the beginning of the next iteration, the elements in A[j..j+1] which are the elements previously in A[j-1..j] are in sorted order because they have not been moved since the end of the last iteration. Thus the loop invariant maintenance step holds.

2.b Before iteration each iteration i of the outer for loop, A[1..i] contains elements from A in sorted order.

Initialization

Before the first iteration, $i = 1$, A[1..1] contains elements from A in sorted order. A[1..1] is just A[1] and a single element is always sorted, thus the loop invariant holds.

Maintenance

Assume the loop invariant is true before some iteration i. That says that all elements in A[1..i] are elements from A in sorted order. During that loop, the next smallest item will be swapped into place A[i]. At the end of the loop, the statement is true because A[1..i] still contains the same elements, but before the next iteration A[1..i] will be equivalent to A[1..i+1] which is in sorted order.

Termination (of inner loop)

At the end of the inner for loop, $j = i$, the termination condition states A[i..i+1] are elements from A in sorted order. Those are the next two smallest elements to be sorted so this proves the outer for loop, because the outer loop invariant claims that A[1..i] will be sorted, so this will 'sort' the next two items.

2.d The best case time is $\Theta(n^2)$which occurs when all of the items are already in sorted order.This is because the algorithm has to go through each item in the list for both loops, even if the items are already sorted. The worst case time is also $\Theta(n^2)$ because the algorithm does not behave differently if the items are already sorted.

CORRECTION: The best case time is actually $\Theta(n)$ if the algorithm is optimized to jump out of the loop early if no swaps were made.