Insertion Sort

Algorithm 1 Insertion-Sort(A)

```
for j = 2 to A.Length do
    key = A[j]
                                      ▷ Insert A[j] into the sorta sequence A[1...j − 1].
    i = j − 1
    while i > 0 and A[i] > key do
        A[i + 1] = A[i]
        i = i − 1
    end while
    A[i + 1] = key
end for
```

Loop Invariant: At the start of each iteration of the **for** loop, the subarray $A[1 \dots j-1]$ consists of the elements original in $A[1 \dots j-1]$, but in sorted order.

Initialization: We start by showing that the loop invariant holds before the first loop iteration, when $j = 2$. The subarray $A[1 \dots j-1]$, therefore, consists of just the single element $A[1]$, which is in fact the original element in $A[1]$. Moreover, this subarray is sorted (trivially), which shows that the loop invariant holds prior to the first iteration of the loop.

Maintenance: Informally, the body of the **for** loop works by moving $A[j-1], A[j-2], A[j-3], \dots$ by one position to the right until it ifnds the proper position for $A[j]$. The subarray $A[1ldotsj-1]$ then consists of the elements originally in $A[1 \dots j]$, but in sorted order. Incrementing j for the next iteration of the **for** loop then preserves the loop iteration. [1]

Termination: The condition causing the **for** loop to terminate is that $j > A.length = n$. Because each loop iteration increases j by 1, we must have $j = n + 1$ at that time. Substituting $n + 1$ for j in the wording of loop invariant, we hae that the subarray $A[1 \dots n]$ consists of the elements originally in $A[1 \dots n]$, but in sorted order. Observing that the subarray $A[1 \dots n]$ is the entire array, we conclude that the entire array is sorted.

[1] A more formal treatment of the Maintenance property would require us to state and show a loop invariant for the **while** loop.