Problem

Input: A sequence of n numbers $\langle a_0, a_1, \ldots, a_{n-1} \rangle$.
Output: A reordering $\langle a'_0, a'_1, \ldots, a'_{n-1} \rangle$ of the input sequence such that $a'_0 \leq a'_1 \leq \cdots \leq a'_{n-1}$.

Algorithm

```
1 insertion-sort(A[0 .. n-1])
2     for i = 1 to n - 1
3         key = A[i]
4         j = i
5         while j > 0 and A[j-1] > key
6             j = j - 1
7         if i != j
8             A[j+1 .. i] = A[j .. i-1]
9             A[j] = key
```

Correctness

Let's define a ***loop invariant*** $\mathcal{L}$ to help us prove the correctness of the algorithm:

$\mathcal{L}(i)$: At the start of the iteration i of the for loop, the subarray $A[0 : i-1]$ contains the same elements originally in the input, but is now sorted.

1. **Initialization:** The loop invariant is true before the first iteration of the loop.

   Trivially, $A[0 : 0]$ is just a single element, thus is sorted and the same as the original input. $\mathcal{L}(1)$ is true.

2. **Maintenance:** If the loop invariant is true before an iteration of the loop, then it remains true before the next iteration.

   Suppose that $\mathcal{L}(i)$ is true, that is, at the start of the iteration i, the subarray $A[0 : i-1]$ contains the same elements originally in the input and is sorted:

   $$A[0] \leq A[1] \leq \cdots \leq A[i-1].$$

   In the while loop, we find the correct place to insert $A[i]$. Specifically, there are two possible termination on line 5.

   (a) Terminates on $j = 0$. In this case, we have that all elements from the subarray $A[0 : i-1]$ is greater than *key*:

   $$key < A[0] \leq A[1] \leq \cdots \leq A[i-1]$$

   (b) Terminates on $A[j-1] \leq key$. This means that we found j such that

$$A[0] \leq A[1] \leq \cdots \leq A[j-1] \leq key < A[j] \leq \cdots \leq A[i-1].$$

Now we consider two cases at line 7.

   (a) If $i = j$, then we have

$$A[0] \leq A[1] \leq \cdots \leq A[i-1] \leq A[i].$$

   The subarray $A[0:i]$ is sorted and unmodified in this iteration.

   (b) If $i \neq j$, then we move all the element from the subarray $A[j:i-1]$ by one position to the right, and assign the original value of $A[i]$ to $A[j]$ via key. Now we have

$$A[0] \leq A[1] \leq \cdots \leq A[j-1] \leq A[j] < A[j+1] \leq \cdots \leq A[i].$$

   The subarray $A[0:i]$ is sorted. And it still contains the original elements since

   - $A[0:j-1]$ is untouched;
   - the new $A[j+1:i]$ is one-to-one to the original $A[j:i-1]$;
   - the new $A[j]$ is the original $A[i]$.

In both cases, after the iteration completes, the subarray $A[0:i]$ is sorted and contains the same elements as the original input $A[0:i]$. Therefore, at the start of the next iteration $i+1$, $\mathcal{L}(i+1)$ holds.

3. **Termination:** When the loop terminates, the invariant gives us a useful property that helps show that that algorithm is correct.

   After the last iteration of the loop $i = n-1$ completes, the maintenance property ensures that $\mathcal{L}(n)$ is true: the subarray $A[0:n-1]$ contains the same elements originally in the input, but is now sorted. This is exactly the desired output.

Therefore, the algorithm is correct and thus solve the sorting problem.