1. First, we give the computer a list of unsorted numbers and store them in an array of memory cells.
2. To begin the sort, the computer divides the sorted and unsorted sections of the list by placing a marker after the first number. To sort the numbers, it will repeatedly compare the first unsorted number with the numbers in the sorted section. If the unsorted number is smaller than its sorted neighbor, the computer will swap them.
3. The first number in the unsorted section is 8, so the computer compares it with the number to the left. Since 8 is greater than 7, these numbers do not need to swapped and the computer simply advances the marker one position. Notice that only one comparison was needed to sort the 8.
4. Now the first number in the unsorted section is 5. 5 is less than 8, so the computer swaps these numbers.

5 is also less than 7, so the computer swaps these numbers as well.

Now 5 is in the correct order, so the computer advances the marker one position. This time two comparisons and two swaps were needed to sort the number.

1. Now the first number in the unsorted section is 2. 2 is less than 8, 7, and 5, so after three comparisons and three swaps, 2 arrives at the correct sorted position, and the computer advances the sort marker.
2. Now the first number in the unsorted section is 4. 4 is less than 8, 7, and 5 but it is not less than 2. This time the computer performs four comparisons and three swaps to put the 4 in the correct order. Only three swaps were needed since the 2 and the 4 did not need to be switched. After these comparisons and swaps, the computer advances the sort marker.
3. Now 6 is the first number in the unsorted section. After three comparisons and two swaps, the computer places the 6 in the correct position between 5 and 7. Notice that the computer did not need to compare the 6 with the 2 or the 4 since it already knows these numbers are less than 5. Once the computer finds a number in the sorted section less than 6, it knows it has found the correct position for 6 and it can advance the sort marker.
4. The final unsorted number is 3. To find the correct position for 3, the computer must compare it with every number in the unsorted section. However, only five swaps are required since the first number (2) is less than 3. After moving 3 to the correct position and advancing the sort marker, the Insertion Sort is complete since the unsorted section is empty.