Isaac Rosenberg, Eric Myzelev, and Joshua Rovner
April 4, 2019

Insertion Sort

Insertion sort is a sorting algorithm that passes through each element in an array, immediately putting each one into its correct, sorted position before moving on to the next element. This is done by comparing the current index's value with each previous one, going as far back towards the beginning of the array as necessary, and placing the current element into its proper position. Insertion sort is most effective with smaller lists, and is more effective than selection sort (the other small-list sorting algorithm) when a list is already mostly sorted, performing in O (n) time.

Average Time Complexity: O (n^2)

Example:

This array contains the starting values before the sort. The final array will contain the same values in ascending order. For example purposes, the current index will be highlighted in red, and the value it is compared to will be in green.

4	**7**	**3**	**11**	**9**

Step 1: Begin by comparing the second value with the first value.

4	**7**	**3**	**11**	**9**

Since 4 is less than 7, nothing needs to be shifted.

Step 2: Compare the third value with the one before.

4	**7**	**3**	**11**	**9**

Since 3 is less than 7, their positions will be switched.

Step 3: The 3, in the second index, will be compared with the 4, in the first index.

4	**3**	**7**	**11**	**9**

Since 3 is less than 4, their positions will be switched.

Step 4: Compare the fourth value with the third value.

3	**4**	**7**	**11**	**9**

Since 7 is less than 11, nothing needs to be shifted.

Step 5: Compare the fifth value with the fourth value.

3	**4**	**7**	**11**	**9**

Since 9 is less than 11, their positions will be switched.

Step 6: The 9, in the fourth position, will be compared with the 7, in the third position.

3	**4**	**7**	**9**	**11**

Since 9 is greater than 7, nothing needs to be shifted.

Sorted Array:

3	**4**	**7**	**9**	**11**