# Insertion Sort

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8] Start

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8] 1 comparison, 0 elements move.

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8] 1 comparison, 0 elements move.

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8] 1 comparison, 1 element moves.

[2, 7, 4, 9, 1, 5, 3, 6, 0, 8] 1 comparison, 1 element moves.

[2, 4, 7, 9, 1, 5, 3, 6, 0, 8] 1 comparison, 0 elements move.

[2, 4, 7, 9, 1, 5, 3, 6, 0, 8] 1 comparison, 1 element moves.

[2, 4, 7, 1, 9, 5, 3, 6, 0, 8] 1 comparison, 1 element moves.

[2, 4, 1, 7, 9, 5, 3, 6, 0, 8] 1 comparison, 1 element moves.

[2, 1, 4, 7, 9, 5, 3, 6, 0, 8] 1 comparison, 1 element moves.

[1, 2, 4, 7, 9, 5, 3, 6, 0, 8] 1 comparison, 0 element moves.

[1, 2, 4, 7, 9, 5, 3, 6, 0, 8] 1 comparison, 1 element moves.

[1, 2, 4, 7, 5, 9, 3, 6, 0, 8] 1 comparison, 1 element moves.

[1, 2, 4, 5, 7, 9, 3, 6, 0, 8] 1 comparison, 0 element moves.

[1, 2, 4, 5, 7, 9, 3, 6, 0, 8] 1 comparison, 1 element moves.

[1, 2, 4, 5, 7, 3, 9, 6, 0, 8] 1 comparison, 1 element moves.

[1, 2, 4, 5, 3, 7, 9, 6, 0, 8] 1 comparison, 1 element moves.

[1, 2, 4, 3, 5, 7, 9, 6, 0, 8] 1 comparison, 1 element moves.

[1, 2, 3, 4, 5, 7, 9, 6, 0, 8] 1 comparison, 0 element moves.

[1, 2, 3, 4, 5, 7, 9, 6, 0, 8] 1 comparison, 1 element moves.

[1, 2, 3, 4, 5, 7, 6, 9, 0, 8] 1 comparison, 1 element moves.

[1, 2, 3, 4, 5, 6, 7, 9, 0, 8] 1 comparison, 0 element moves.

[1, 2, 3, 4, 5, 6, 7, 9, 0, 8] 1 comparison, 1 element moves.

[1, 2, 3, 4, 5, 6, 7, 0, 9, 8] 1 comparison, 1 element moves.

[1, 2, 3, 4, 5, 6, 0, 7, 9, 8] 1 comparison, 1 element moves.

[1, 2, 3, 4, 5, 0, 6, 7, 9, 8] 1 comparison, 1 element moves.

[1, 2, 3, 4, 0, 5, 6, 7, 9, 8] 1 comparison, 1 element moves.

[1, 2, 3, 0, 4, 5, 6, 7, 9, 8] 1 comparison, 1 element moves.

[1, 2, 0, 3, 4, 5, 6, 7, 9, 8] 1 comparison, 1 element moves.

[1, 0, 2, 3, 4, 5, 6, 7, 9, 8] 1 comparison, 1 element moves.

[0, 1, 2, 3, 4, 5, 6, 7, 9, 8] 1 comparison, 0 element moves.

[0, 1, 2, 3, 4, 5, 6, 7, 9, 8] 1 comparison, 1 element moves.

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] 1 comparison, 0 element moves.

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] 1 comparison, 0 element moves.

33 Comparisons, 23 elements moved.

Insertion sort is a simple algorithm that maintains a sorted list in the lower half of the array, as shown above, as it moves along the array it sorts the numbers out, with the smallest being at the front and the largest being at the back. As it sorts through the list it checks the next item in the list and then inserts it back such as the next item next to it is larger in the sorted list. It has an average case of O(n2), which means it is very inefficient algorithm and is very rarely used.

# Bubble-sort

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8]

Key:

2,7 means that a comparison has occurred, but nothing has swap.

9,4 means that a comparison has occurred, and they have swapped

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8]

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8]

[2, 7, 4, 9, 1, 5, 3, 6, 0, 8]

[2, 7, 4, 1, 9, 5, 3, 6, 0, 8]

[2, 7, 4, 1, 5, 9, 3, 6, 0, 8]

[2, 7, 4, 1, 5, 3, 9, 6, 0, 8]

[2, 7, 4, 1, 5, 3, 6, 9, 0, 8]

[2, 7, 4, 1, 5, 3, 6, 0, 9, 8]

[2, 7, 4, 1, 5, 3, 6, 0, 8, 9]

[2, 7, 4, 1, 5, 3, 6, 0, 8, 9]

[2, 4, 7, 1, 5, 3, 6, 0, 8, 9]

[2, 4, 1, 7, 5, 3, 6, 0, 8, 9]

[2, 4, 1, 5, 7, 3, 6, 0, 8, 9]

[2, 4, 1, 5, 3, 7, 6, 0, 8, 9]

[2, 4, 1, 5, 3, 6, 7, 0, 8, 9]

[2, 4, 1, 5, 3, 6, 0, 7, 8, 9]

[2, 4, 1, 5, 3, 6, 0, 7, 8, 9]

[2, 4, 1, 5, 3, 6, 0, 7, 8, 9]

[2, 4, 1, 5, 3, 6, 0, 7, 8, 9]

[2, 1, 4, 5, 3, 6, 0, 7, 8, 9]

[2, 1, 4, 5, 3, 6, 0, 7, 8, 9]

[2, 1, 4, 3, 5, 6, 0, 7, 8, 9]

[2, 1, 4, 3, 5, 6, 0, 7, 8, 9]

[2, 1, 4, 3, 5, 0, 6, 7, 8, 9]

[2, 1, 4, 3, 5, 0, 6, 7, 8, 9]

[2, 1, 4, 3, 5, 0, 6, 7, 8, 9]

[2, 1, 4, 3, 5, 0, 6, 7, 8, 9]

[1, 2, 4, 3, 5, 0, 6, 7, 8, 9]

[1, 2, 4, 3, 5, 0, 6, 7, 8, 9]

[1, 2, 3, 4, 5, 0, 6, 7, 8, 9]

[1, 2, 3, 4, 5, 0, 6, 7, 8, 9]

[1, 2, 3, 4, 0, 5, 6, 7, 8, 9]

[1, 2, 3, 4, 0, 5, 6, 7, 8, 9]

[1, 2, 3, 4, 0, 5, 6, 7, 8, 9]

[1, 2, 3, 4, 0, 5, 6, 7, 8, 9]

[1, 2, 3, 4, 0, 5, 6, 7, 8, 9]

[1, 2, 3, 4, 0, 5, 6, 7, 8, 9]

[1, 2, 3, 4, 0, 5, 6, 7, 8, 9]

[1, 2, 3, 4, 0, 5, 6, 7, 8, 9]

[1, 2, 3, 0, 4, 5, 6, 7, 8, 9]

[1, 2, 3, 0, 4, 5, 6, 7, 8, 9]

[1, 2, 3, 0, 4, 5, 6, 7, 8, 9]

[1, 2, 3, 0, 4, 5, 6, 7, 8, 9]

[1, 2, 3, 0, 4, 5, 6, 7, 8, 9]

[1, 2, 3, 0, 4, 5, 6, 7, 8, 9]

[1, 2, 3, 0, 4, 5, 6, 7, 8, 9]

[1, 2, 3, 0, 4, 5, 6, 7, 8, 9]

[1, 2, 0, 3, 4, 5, 6, 7, 8, 9]

[1, 2, 0, 3, 4, 5, 6, 7, 8, 9]

[1, 2, 0, 3, 4, 5, 6, 7, 8, 9]

[1, 2, 0, 3, 4, 5, 6, 7, 8, 9]

[1, 2, 0, 3, 4, 5, 6, 7, 8, 9]

[1, 2, 0, 3, 4, 5, 6, 7, 8, 9]

[1, 2, 0, 3, 4, 5, 6, 7, 8, 9]

[1, 2, 0, 3, 4, 5, 6, 7, 8, 9]

[1, 0, 2, 3, 4, 5, 6, 7, 8, 9]

[1, 0, 2, 3, 4, 5, 6, 7, 8, 9]

[1, 0, 2, 3, 4, 5, 6, 7, 8, 9]

[1, 0, 2, 3, 4, 5, 6, 7, 8, 9]

[1, 0, 2, 3, 4, 5, 6, 7, 8, 9]

[1, 0, 2, 3, 4, 5, 6, 7, 8, 9]

[1, 0, 2, 3, 4, 5, 6, 7, 8, 9]

[1, 0, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

There are 81 compares and the elements are moved 23 times.

Bubble sort is a type of sorting algorithm where the largest number “bubbles” its way to the top of the array. As shown in the process above, when the algorithm gets to the number 9, it slowly moves itself to the top of the array, it then does this to the rest of the numbers, the smallest number, 0, is the last number to find its place in the array. Even after it’s sorted the array it must go through once again to make sure. You can make bubble sort a lot more efficient by making the for loop that runs the algorithm smaller for each iteration, because each after each iteration, the largest number will always be at the top, so you don’t need to check if that is the largest because it is.

# Selection Sort

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8] Start

Key:

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8] used when the program is looking for the largest number, it does every time it moves the numbers about, I’ve left the other moves out.

0 Indicates that it has been moved to the correct position

2 Indicates that it has be moved to but not to the right position, it has been moved because the place where it was is now occupied by the correct number, this number has been moved to the olds’ number place.

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8]

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8]

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8]

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8]

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8]

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8]

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8]

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8]

[2, 7, 9, 4, 1, 5, 3, 6, 0, 8]

[0, 7, 9, 4, 1, 5, 3, 6, 2, 8]

[0, 1, 9, 4, 7, 5, 3, 6, 2, 8]

[0, 1, 2, 4, 7, 5, 3, 6, 9, 8]

[0, 1, 2, 3, 7, 5, 4, 6, 9, 8]

[0, 1, 2, 3, 4, 5, 7, 6, 9, 8]

[0, 1, 2, 3, 4, 5, 7, 6, 9, 8]

[0, 1, 2, 3, 4, 5, 6, 7, 9, 8]

[0, 1, 2, 3, 4, 5, 6, 7, 9, 8]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

It does 9 swaps but does 45 compares.

Select sort makes one exchange for every pass of the list, meaning that every time the array is passed it will move one element. The select sort algorithm searches for the smallest or largest element in a pass and then puts it in the correct location. After the second pass it will be the second largest or smallest that will be moved to its correct place, it has an average O(n) for swaps because it only swaps it once, but O(n2) for comparisons because it compares every element through every pass.