Insertion sort is one of the easier algorithms to implement. The summary description of how this algorithm works is this: Start iterating over each item in the data structure (most of the time is an array or vector), starting with the second element in the data structure, check the current element’s value to the element prior to it. For every element’s value in the data set that comes before the current element, if it’s value is less than the current element’s value, move that element up a position in the set and keep cycling through until there is an element whose value is not less than the current element’s value. And place the current element’s value in that spot.

Best times to use this algorithm:

When data set is relatively small

When items in the data set are semi-sorted already

Running time complexity:

Worst case performance: O(n²) — So if you have 100 elements in your array, at worst, this algorithm will do 10,000 comparisons. This is when your data set is sorted completely opposite of how you wish to sort the data.

Average case performance: O(n²) — The average case won’t quite be 10,000 comparisons, but it will be higher than O(n) or O(nlogn) time.

Best case performance: O(n) — The best case performance is when the data set is already sorted and it only needs to iterate each item in the array and do one comparison for each item in the data set.