**Sorting**

-Arranging things into ascending or descending order is called sorting.

-You can sort any objects that are Comparable (that is, objects of any class that implements the interface Comparable)

Selection Sort (aka Bubble Sort)

– find the smallest number and exchange it with starting position.

-elements trade places with other elements

-O(n^2) regardless of the initial order of the elements in an array

Insertion Sort

-compare elements as you traverse through the array

-if element is larger than previous element then swap them.

-you are comparing unsorted elements with sorted elements

-an insertion sort **partitions** (divides) array into two parts: one is sorted and the other contains remaining elements)

-O(n^2); at best this is O(n). The closer the array is to sorted order the less work an insertion sort does

Shell Sort

-capitalizes on Insertion Sort being O(n) if most elements are almost sorted

-sorts subarray of elements at equally spaced indices; instead of moving to an adjacent location, an element moves several locations away

-example: if array of size 13 🡪 13/2 = 6

Start with position 0 go to 6 then go to 12

Start with 1 go to 7

Start with 2 go to 8

Start with 3 go to 9

Start with 4 go to 10

Start with 5 go to 11

Use insertion sort to sort these subarrays

Class Note: Gap = array.length / 2 = n -🡪 n/2 = nNew 🡪 continue until n = 1. At this point do normal insertion sort.

-O(n) best

-O(n^1.5) avg

-O(n^2) worst or O(n^1.5)

**Faster Sorting Methods**

Merge Sort

-divides an array into halves, sorts the two halves, and then merges them into one sorted array

-often done recursively

-divide and conquer algorithm; divide problem into pieces and conquer each piece to reach a solution

-start with two distinct arrays that are sorted

-obtain a third array that will contain two merged arrays

-process both arrays from beginning to end comparing an element in one array with an element in the other and copying the smaller element to a new third array.

-after reaching the end of one array you simply copy the remaining elements into the third array

-O(nlogn)

Disadvantage of merge sort is that we require a temp array during the merge step

Quick Sort

-divides an array into two pieces, but unlike merge sort, these pieces are not necessarily halves of the array.

-Quick sort chooses one element in the array called **pivot** and rearranges the array elements so that:

a) the pivot is in the position that it will occupy in the final sorted array

b) elements in positions before the pivot are less than or equal to the pivot

c) elements in positions before the pivot are greater than or equal to the pivot

-This arrangement is called **partition** of the array.

-creating a partition divides the array into two pieces which we will called Smaller and Larger (separated by the pivot)

O(nlogn) best

O(n^2) worst (choice of pivots affects its behavior)