# Sorting and Searching

The example file contains two types of search techniques, each as its own class, Insertion sort and Merge sort.

The insertion sort technique treats an array as being composed of two parts, a sorted list followed by an unsorted list. The sort moves elements from the unsorted list to the sorted list one element at a time and to the correct position in the sorted list.

The merge sort works recursively. It breaks an array into two halves and continues to separately break each half into subarrays until they are each of length one. Then the adjacent pairs of lists are recursively merged into sorted arrays until the full array is created again.

1. **For 1000 samples, which sort method performed faster?**
2. **Re-test the program at 100 samples. How does the time difference vary compared to 1000 samples?**
3. **Re-test the program at 10 samples. How does the time difference vary compared to 100 samples?**
4. **From the number of samples testing between the two sort methods, what can you conclude would be a more efficient method as the number of samples increases?**

The example file showed acceleration data being sampled and then sorted.

1. **How does the total sample time relate to the number of samples in the program?**

A sequential search looks to find a key value in a list of n elements, where n is a positive real number. The search starts at the first element of an array and compares a key value to each element until it finds it or there are no more elements in the list.

1. **Create a new class in the example file called SequentialSearch. Implement a method called contains() that has logic for a sequential search through an array given a key value input argument.**

Suppose you wanted to create a sequential search method that counts the number of instances of a value in an array.

1. **Add a new method, the SequentialSearch class that is called containCount() that returns the count value of a key in an array.**