# Assignment 6

**Billy Ross**
Rene German
CPSC350-3
12/13/19
wross@chapman.edu

## Abstract

For this project we where asked to go over different sorting algorithms and use empirical analysis to experiment and see the difference of times between each method. Furthermore we will look into the mathematical analysis of each sorting method and determining the big O run time.

## 1 Empirical Analysis

Through building this program and testing the different sorting algorithms I found that in the time I recorded merge sort was the fastest. Although I did not have time to test these algorithms with large amounts of numbers, the amount I did test with showed a large difference in time. From my own research I have found that the programming language of choice should not effect the rum times of these. Although the users ability could effect him when constructing these methods in a different programming language. Some of the short comings from the empirical analysis performed was the amount of time it takes in order to build the surrounding methods for testing.

## 2 Mathematical Analysis

In this section we will look into each big o run time of the sorting methods.

- Quick Sort: O(nlog(n)) worst case- O(n squared)
- Merge Sort: O(nlog(n))
- Insertion Sort: O(n squared)
- Selection Sort: O(n squared)
- Bubble Sort: O(n squared)

It can be concurred from these run times that merge sort is the best as far as run times. This beats quick sort because in its worst case quick sort is at O(n squared) which is much slower than O(nlog(n)).

## 3 Trade offs

Some of the trade offs of these sorting methods is the different amount in CPU or memory usage. With merge sort the amount of memory that is being used is a lot due to arrays being made constantly. With quick sort however it used more CPU and less ram due to the amount of processing it needs to do.