**Alpha2011-**Merge

**Beta-**Selection

**Gamma**-Bubble

**Delta**-Quick

**Epsilon-**Insertion

**Zeta-**Heap

**Analysis**

We first set the list size to 16, in ascending order. Using this list, we deduced that Beta and Gamma were selection and bubble sorts, because they require no movement to sort a list that is already in order. However, when we reversed the order of the array, Gamma required N2comparisons, identifying it as a bubble sort. This means that Beta must be selection sort.

We also observed that when using the size 16, ascending order array, Epsilon completed the sort with only 15 comparisons, which clearly indicates an insertion sort. This is because in a sorted array, each value after the first is compared to the previous value and since the array is sorted, the previous value will be less and thus only N-1 comparisons are required.

Continuing to use the same array, we discovered that Delta made considerably more comparisons, yet sorted with N-1 movements. This led us to believe that Delta was a quicksort, because the pivot had been selected as a max or min value, leading to an inefficient amount of comparisons. However, since the array was already sorted, only N-1 movements were needed to be made. We decided that Delta was a quicksort.

With Alpha2011 and Zeta remaining, we were sure that only merge sort and heap sort remained. We decided to investigate the number of comparisons made by each sort. By changing the list size, we repeated noticed that Alpha2011 completed the sort with less comparisons than Zeta. Knowing that merge sort is more stable than heap sort, we decided to test the reliability of each sort at 60000 values. Alpha2011 completed consecutive sorts mostly within 30-32. However, Zeta completed consecutive searches with fluctuating times ranging from 0-55. Finally, we repeatedly noticed that Zeta required fewer movements than Alpha2011. We identified Alpha2011 as a merge sort because it required fewer comparisons (it only needs comparisons when merging), performed more reliably with high volume lists, and required more movements because of all the auxiliary space the size-one arrays need. We decided Zeta was a heap sort because it takes more comparisons than merge sort to heap down every other step, is less reliable with high volume lists, and requires less movements than merge sort because it does not necessarily have to traverse the entire heap array to heap down, and only uses one movement to swap the max value to the end.

Graphs