EnochianHendersonWaters – HW4 Analysis

After running the program 135 times (5 times for 3 different list sizes for each threshold value), we have determined the following number of comparisons for each sorting method:

Sort type	***BUBBLE***		
List size	***5***	***T***	***19***
Minimum	1	5	N/A
Maximum	5	14	N/A
Average	3.666666667	9.377777778	N/A
Overall average	**6.522222222 → 7**		
Sort type	**INSERTION**		
List size	***5***	***T***	***19***
Minimum	0	7	N/A
Maximum	9	83	N/A
Average	5.133333333	35.91111111	N/A
Overall average	**20.52222222 → 21**		
Sort type	**MERGE**		
List size	***5***	***T***	***19***
Minimum	N/A	N/A	82
Maximum	N/A	N/A	82
Average	N/A	N/A	82
Overall Average	**82**		
	Note: We determined that this is accurate because mergeSort will always have to do the same number of comparisons as long as the size of each list remains the same.		
Sort type	**QUICK**		
List size	***5***	***T***	***19***
Min	N/A	N/A	23
Max	N/A	N/A	31
Avg	N/A	N/A	26.91111111
Overall avg	**26.91111111 → 27**		

After running our program 135 times with various threshold values for a randomly generated list of various sizes, we have determined that the most efficient method for threshold values less than or equal to the threshold value is bubbleSort, since it takes the least amount of comparisons (on average). The most inefficient sorting method for threshold values greater than 12 was mergeSort, as it always took 82 comparisons to completely sort the lists of size 19. Our sorting methods, from most efficient to least efficient, are:

1. bubbleSort (for threshold values less than or equal to the size of the list)

2. quickSort (for threshold values greater than the size of the list)
3. insertionSort
4. mergeSort

As a result, threshold values that are smaller than the size of the list are preferred for the HybridSort algorithm as it will determine which sorting methods to use depending on the list size and the T value (which will determine what the Large and Small algorithms are).