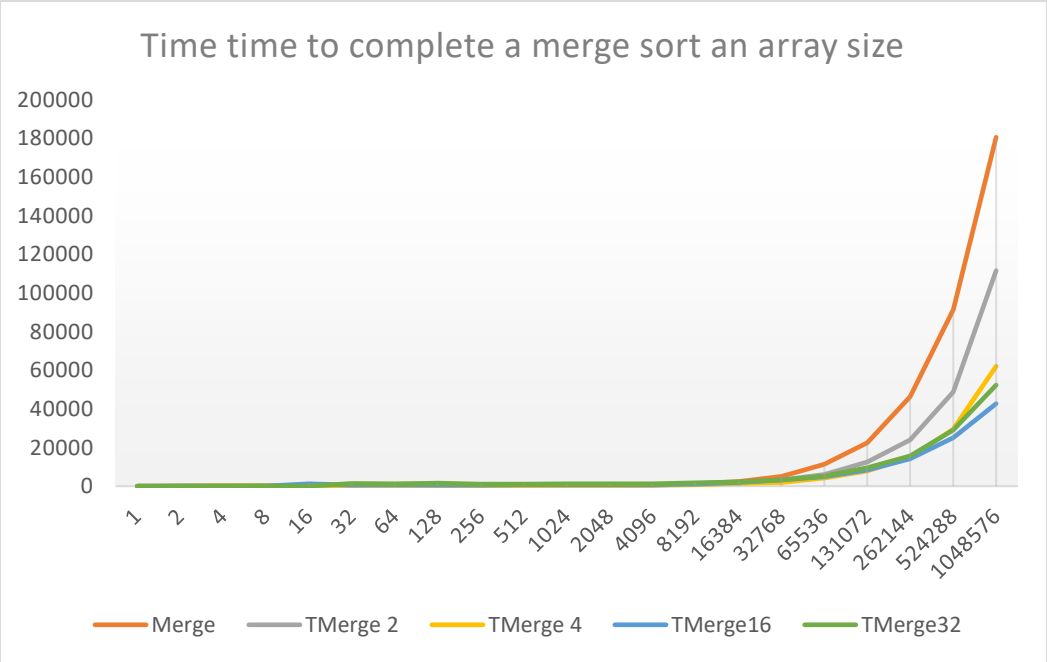
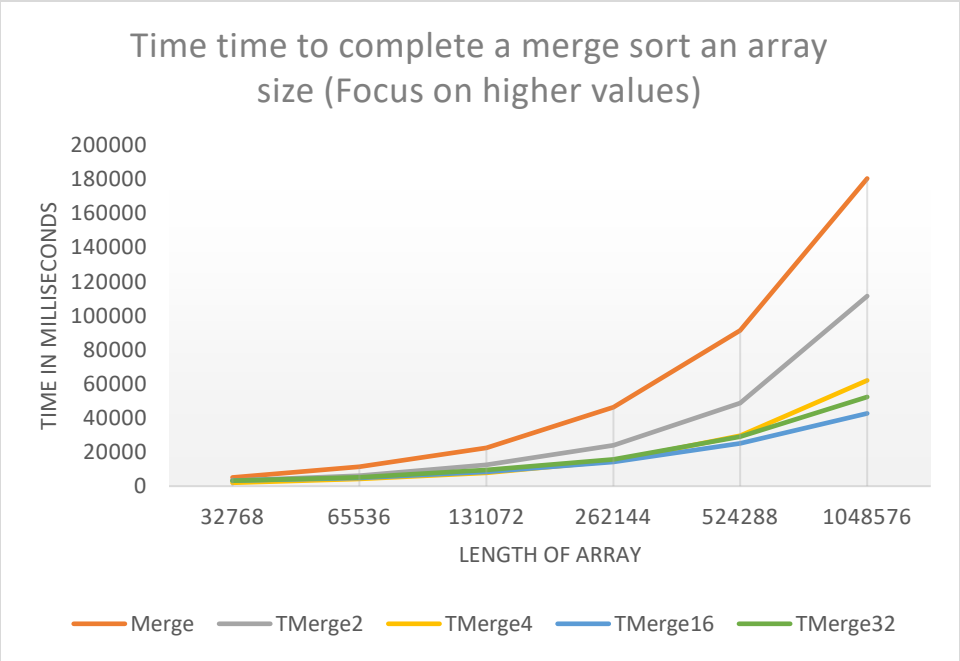
# Homework 4 - CS361 Threaded Merge Sort

**Freddie Patterson**

## Table Data and Plots:

| Array Size | Merge | TMerge 2 | TMerge 4 | TMerge16 | TMerge32 |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 252 | 0 | 0 | 0 |
| 4 | 0 | 357 | 322 | 0 | 0 |
| 8 | 1 | 350 | 264 | 1 | 0 |
| 16 | 2 | 226 | 389 | 1202 | 1 |
| 32 | 4 | 229 | 307 | 625 | 1469 |
| 64 | 8 | 153 | 375 | 758 | 1174 |
| 128 | 12 | 169 | 233 | 605 | 1613 |
| 256 | 26 | 188 | 281 | 639 | 1039 |
| 512 | 76 | 262 | 232 | 715 | 1121 |
| 1024 | 108 | 220 | 243 | 664 | 1248 |
| 2048 | 319 | 350 | 311 | 704 | 1239 |
| 4096 | 599 | 634 | 451 | 736 | 1241 |
| 8192 | 1174 | 1052 | 750 | 1034 | 1803 |
| 16384 | 2386 | 1609 | 1358 | 1865 | 2153 |
| 32768 | 5098 | 3229 | 1979 | 3251 | 3194 |
| 65536 | 11292 | 6087 | 4181 | 4870 | 5243 |
| 131072 | 22497 | 12562 | 7800 | 8422 | 9441 |
| 262144 | 46267 | 23917 | 14812 | 14157 | 15650 |
| 524288 | 91362 | 48723 | 29517 | 25143 | 29092 |
| 1048576 | 180591 | 111573 | 62070 | 42700 | 52344 |






**Conclusion on data:**

**The data clearly shows that as thread count increases, the overall time it takes to complete the merge sort decreases. This is not the case necessarily at the low end of the size of the array. I would postulate that this is to do with the fact that the time it takes to even assign the threads, takes longer than it actually takes to sort the array. However, as the value of n items in the array increases, it becomes very clear that the time taken greatly decreases with the number of threads. This evidence would show that the time complexity of the regular merge sort of is O(nlog(n)), and although it keeps generally to the improvement that would be expected in O(n/tLog(n/t)) + O(n), the data clearly shows inconsistency between the improvement by threads that does not follow it exactly, even if it follows the same trend generally e.g. The differences between 131072 and 262144 items.**

**What I have found easiest has to have been getting the overall structure in of the program, not just for the standard single threaded merge sort, but getting the structure for running the threads simultaneously and making sure that things that need to be locked and performed differently from the single threaded version of the merge sort, were surprisingly easy to set up the structure of the program.**

**The hardest has definitely been getting the rejoining of all the threads into the correct merged list. Although not too difficult to do with the lesser number of threads, when running at 16 or 32 threads, the merging of the whole list became very difficult and something I struggled quite a lot with. Also just merge itself. I made the mistake of trying to just directly implement from the written version of the merge function, of which I made quite a lot of syntax mistakes that took me a decent amount of time to debug especially with so many threads running. E.g., declaring an int* list without giving it a size value, and then accidentally making it just an int of that value instead of a size.**