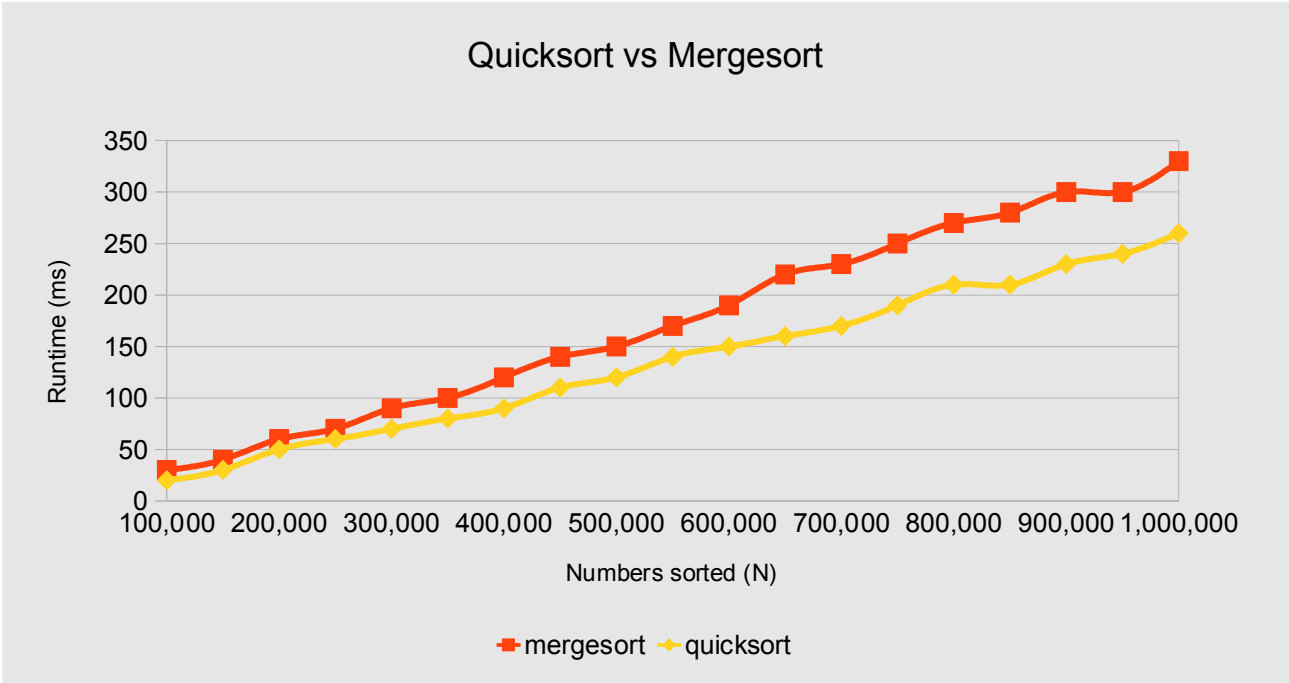
Zachary Jones
COP 3530, Fall 2011

numbers	mergesort	quicksort
100,000	30	20
150,000	40	30
200,000	60	50
250,000	70	60
300,000	90	70
350,000	100	80
400,000	120	90
450,000	140	110
500,000	150	120
550,000	170	140
600,000	190	150
650,000	220	160
700,000	230	170
750,000	250	190
800,000	270	210
850,000	280	210
900,000	300	230
950,000	300	240
1,000,000	330	260



Notes:
Quicksort was the faster of the algorithms. I suspect this is because it requires no memory allocation and copying. Mergesort splits the list in half, copying the halves into new malloc'd space. This causes overhead in the time it takes to allocate memory, copy the list parts into the new memory, sort, and then deallocate that memory. Furthermore every run of both algorithms had to read the numbers from a file which could have been optimized by keeping a copy of the raw number data in memory and reusing it between experiences, verses reading from disk every time.