|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n | S(n) | S(n)/n2 | Q(n) | Q(n)/(nlog2n) | M(n) | M(n)/(nlog2n) |
| 1000 | 3.811 | 3.811e-006 | 0.178 | 1.78611e-005 | 0.199 | 1.99683e-005 |
| 2000 | 18.606 | 4.6515e-006 | 0.32 | 1.45908e-005 | 0.478 | 2.17951e-005 |
| 4000 | 66.55 | 4.15937e-006 | 0.703 | 1.46877e-005 | 0.953 | 1.99109e-005 |
| 8000 | 182.55 | 2.85234e-006 | 1.638 | 1.57916e-005 | 2.197 | 2.11807e-005 |
| 16000 | 778.34 | 3.04e-006 | 3.528 | 1.57886e-005 | 4.154 | 1.85901e-005 |
| 32000 | 3342.08 | 3.263e-006 | 7.444 | 1.55438e-005 | 9.009 | 1.88117e-005 |

The values in the table correspond to the expected output.

For selection sort, you can see that the algorithm is running an O(n2) from the way the numbers grow exponentially with a larger n.

For quick sort and merge sort, you can also see that the algorithm is running an O(nlog2n) time, and works much faster than selection sort. As their results are comparable, we see that both are running approximately the same time.

Quick sort works a little faster than Merge sort, most likely because the worst case for quick sort is uncommon and so Quick sort worked on a better runtime than Merge Sort.