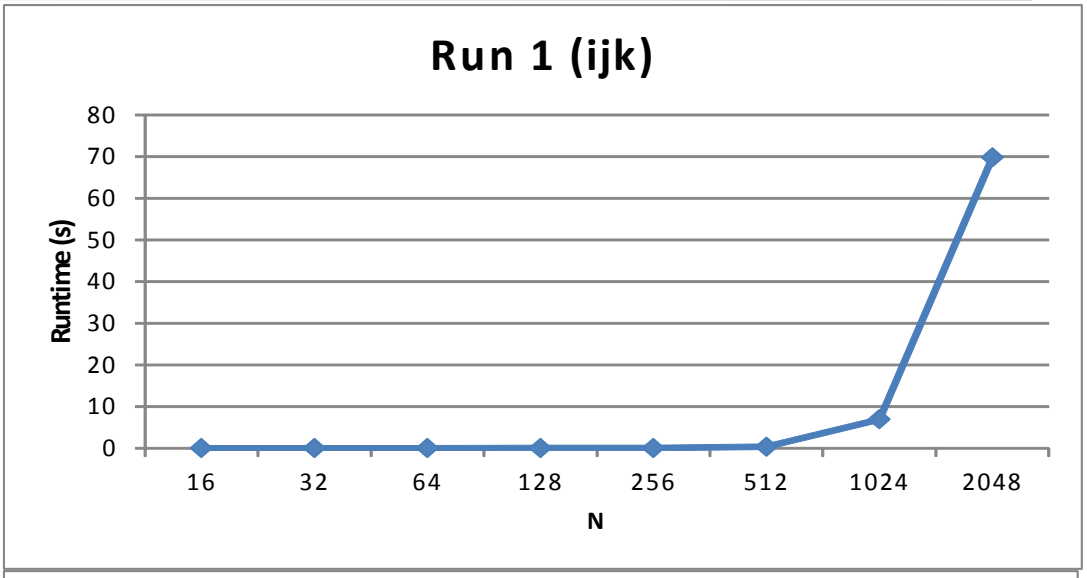
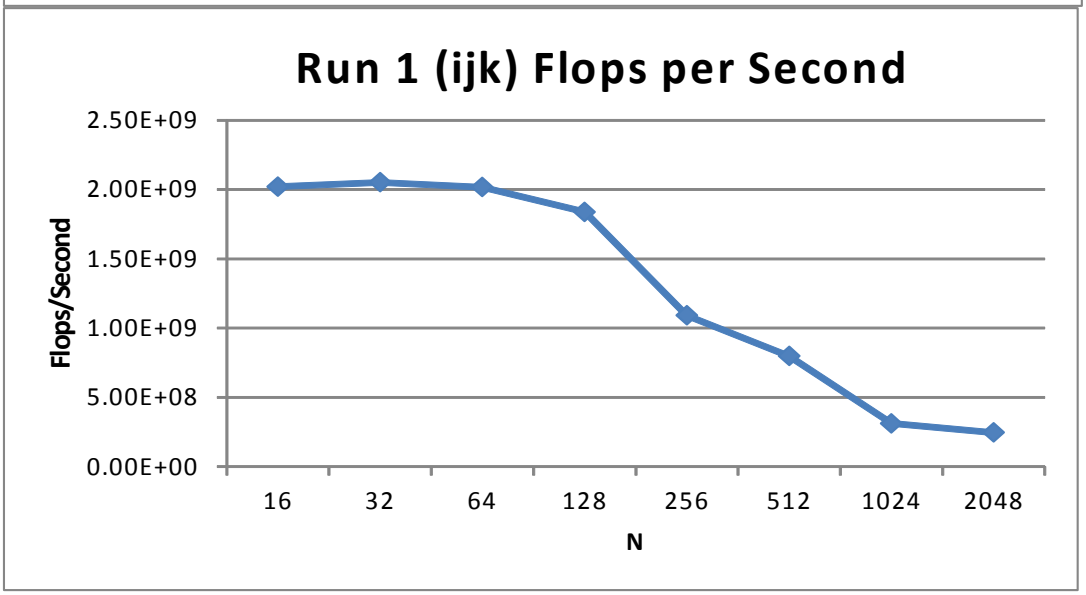
CS140 Homework 1

Steven Cabral
scabral@csil
F 1PM Section

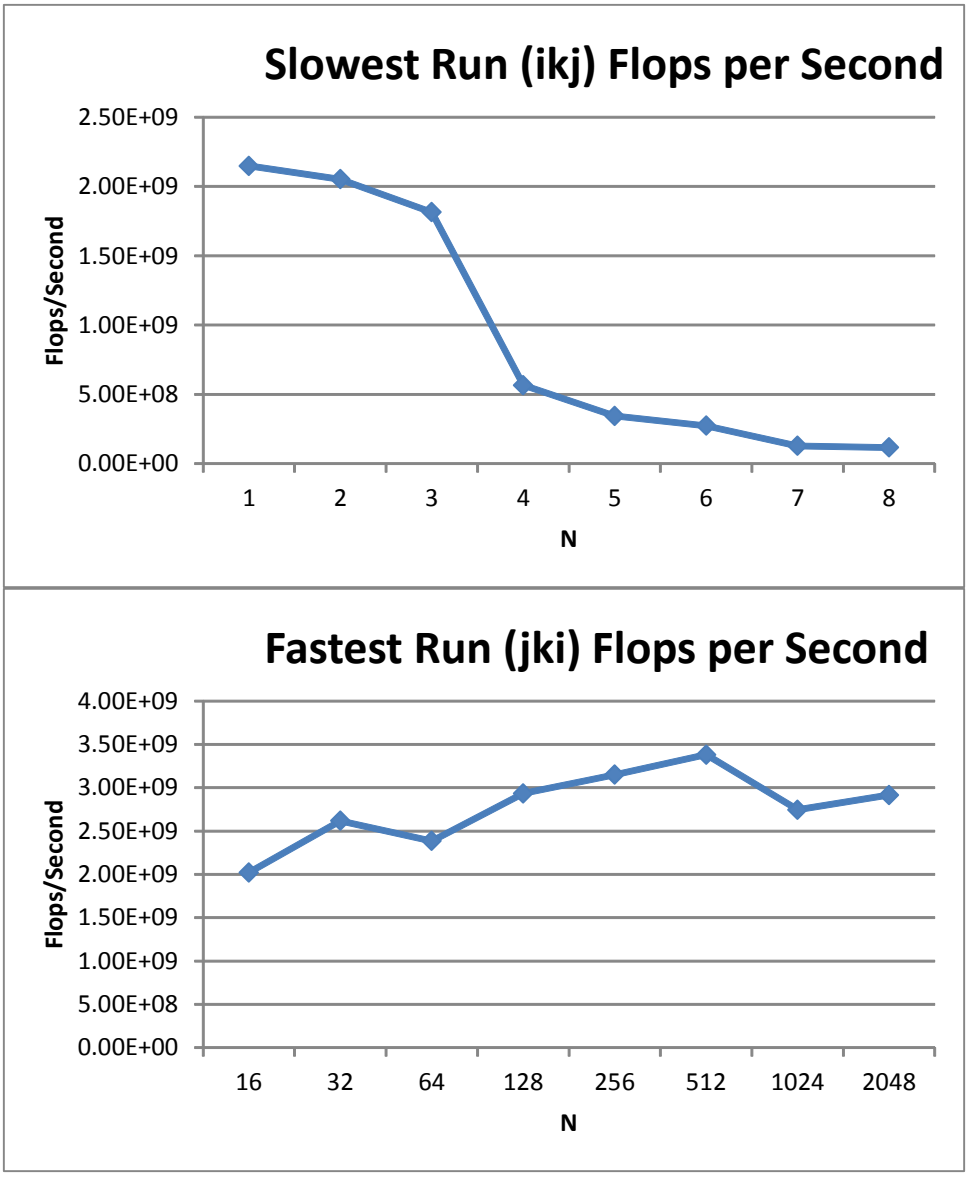
Run 1: rows structures in sequence ijk

N	16	32	64	128	256	512	1024	2048
Time (s)	.000004	.000032	.00026	.002282	.030767	.336614	6.89608	69.77695





As the data shows for the initial run, leaving the rows unchanged, the time appears to scale exponentially with the size of the matrix (N), though the exact power of the scaling term is hard to determine with the graph alone. However, the interesting phenomena occur on the flops per second graph. As the size becomes significantly larger, the flops per second become notably smaller. This is likely due to more of the runtime being dedicated to fetching specific pieces of the memory stored farther away than the level 1 cache. Therefore, by organizing the program to utilize the cache more fully, the program will likely benefit from a performance boost.



The same phenomenon evident in the first run is far more exaggerated in the slowest run. However, the fastest run seems not only to fail to show this phenomenon, but may even be benefiting from an increase in flops per second as the size becomes large, though the plot does demonstrate a lot of noise. The noise is likely because the computer used (csil) was also running different processes, giving a higher variance in the run time of the program. However, the difference in the trends between the two runs is likely due to the fact that in the jki run, entire rows of matrix A are run at once. Since the rows use contiguous memory chunks, the data works better with the cache than doing non-contiguous columns at a time as in the ikj case.

The program can likely benefit from larger performance boosts if the calculations are done in blocks of C at a time, so that only a small, more-contiguous portion of A and B are used at a time. However, this was not implemented in the code. Instead, the only modification of the code is the production of functions matrix_multiply_run_n (where n is 2 to 6), which vary only based on the ordering of the nested loops.