Serial Matrix Product (ijk vs jki)
Hirokatsu (Hiro) Suzuki

Problem

Implement two algorithms (ijk-form, and jki-form) to multiply two matrices that are not necessarily square. For this you may follow lecture notes 4 and 5.

Your program must be able to manipulate matrices with up to 40,000 integer values, which still are very small matrices. You must use the `gettimeofday(…)` routine to **measure the execution time of A*B in seconds**. Take into account that `gettimeofday(…)` provides you the elapsed time with an accuracy of microseconds.

Solution

I implemented both ijk and jki form of matrix multiplication in C++. The pseudo-code was used as a template to the code constructions. For ijk form, the program selects the element of the matrix consecutively to multiply through each column in the second matrix. For jki form, the program selects each row of the matrix to multiply to each element in the second matrix. Below is the table for the runtime with different matrix size and -O0/-O3 optimizations.

Size of Matrices	Algorithm	Runtime (sec) -O0 optimization (No Optimization)	Runtime (sec) -O3 optimization (Full Optimization)	Speed-up
100x100 * 100x100	ijk	0.004562	0.001025	4.6
	jki	0.004624	0.001076	4.3
150x150 * 200x200	ijk	0.020458	0.003572	5.7
	jki	0.021363	0.005342	4.0
400x400 * 400x400	ijk	0.407801	0.131682	3.1
	jki	0.563867	0.344465	1.6

Table.1 Runtime based on an average of 5 runs.

The ijk form has faster runtime because the program loop through each element (row-wise) in the first matrix first which is the way they're stored in the memory. However, the jki form loop through the column of the matrix first which ask the program to jump around the memory as much as the size of the matrix. To sum-up, the ijk form has faster runtime because it favors the memory allocation. In addition, the speed-up from -O0 optimization to -O3 optimization of the jki form is lesser as the size of the matrix increases compare to the ijk form.

The program showed no memory leak checked with valgrind tool.