COL-216
Intelligent Coding

Orin Pao | 2022CS11130

Matrix Multiplication:

Matrix multiplication, inherently a cubic order process requiring three nested loops, yields a result independent of the loop execution order. However, the computation time varies based on the loop order due to modern computer caches relying on "spatial locality" and memory storage patterns for efficiency.

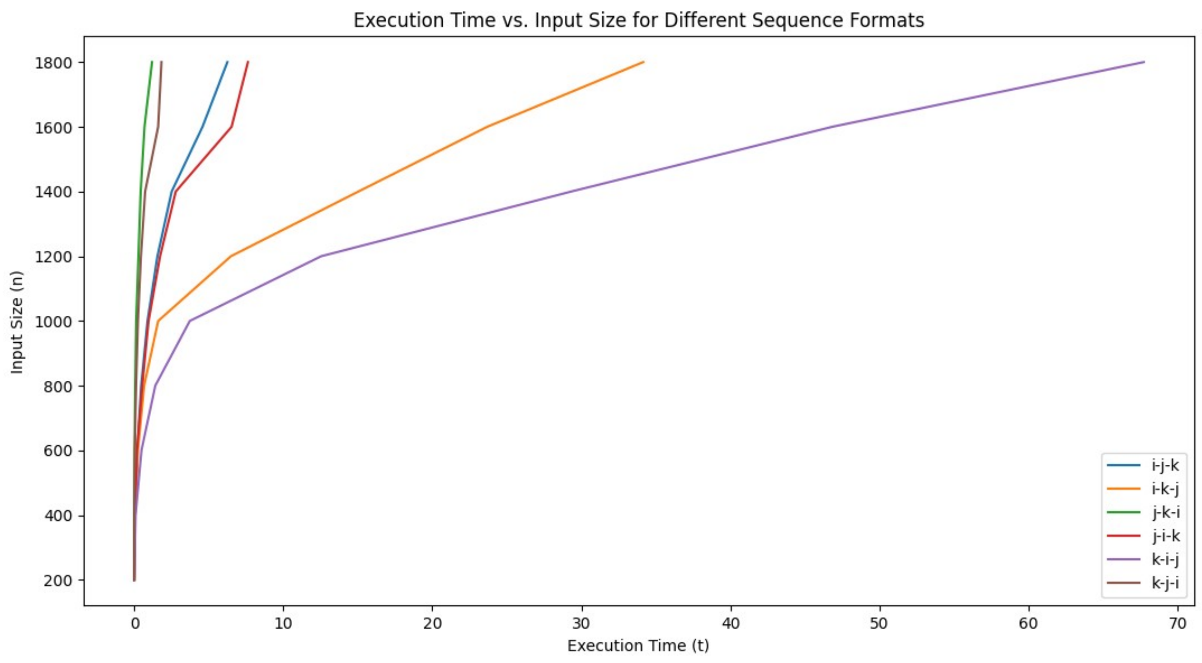
Spatial Locality:

Assuming the matrix is stored row-wise in memory, performing matrix multiplication in a row-major manner, where elements are accessed row by row, improves the hit rate and consequently reduces execution time. Conversely, performing multiplication in a column-major manner results in accessing elements that are further apart in memory, leading to decreased hit rates due to spatial locality and thus longer execution times.

Analysis:

In conducting tests, the execution times for various loop orders were measured, and the discrepancies were plotted against the matrix size. At a specific value of n, the cache memory reaches its limit, resulting in a noticeable abrupt change in the discrepancies. As n increases further, the differences become more pronounced.

The observed increase in the difference is attributed to the exhaustion of the cache as n grows. When cache is depleted, data retrieval from memory becomes necessary. Due to the advantages of spatial locality in row-major operations, the number of memory loads is relatively lower compared to column-major operations. Consequently, the difference between the two methods continues to widen.



Matrix Transposition:

Transposing a nxn matrix can be achieved through two methods: (i) In-place transposition and (ii) Using an additional array. While both methods are expected to produce similar outcomes, the key distinction lies in the extra cache space utilized by the latter approach. Consequently, the value of n at which cache exhaustion occurs is reduced. Moreover, aligning with the concept of spatial locality, performing row-wise swapping and storing data row-wise enhances the hit rate, resulting in reduced execution time. Thus, a larger time difference between the two methods is anticipated.

It can be seen that the time difference is fairly constant to nearly zero (till the cache is exhausted) and then it starts increasing. This is at par with the theory.

Conclusion:

The cache limit in the testing machine was determined using the provided graphs. It was observed that the number of integers stored in the cache is quadratic in relation to the matrix size (number of columns/rows). When the cache limit is reached, the graph begins to distort and deviate from 0. This critical value of n can be considered as the matrix size corresponding to the cache limit. Consequently, the cache limit is approximately (4 * 85 * 85) = 30,000 bytes or 30kB. This aligns closely with the original L1 cache size of the machine, which is 160kB, indicating that the cache may have been optimized to store other crucial data such as instructions. Furthermore, the choice of programming strategies does indeed impact the computational speed, despite having the same time complexity for different strategies

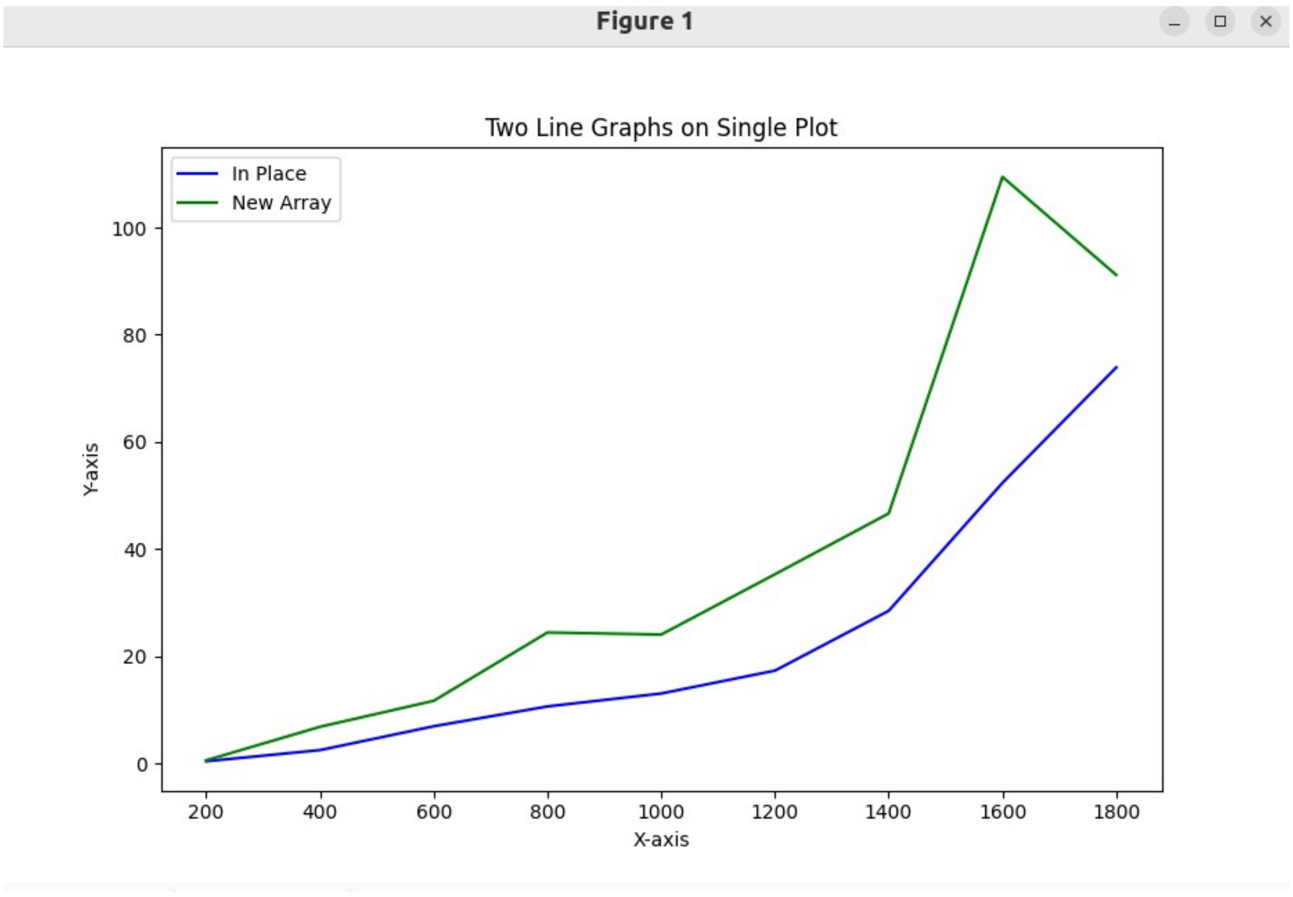


Figure 1
Two Line Graphs on Single Plot
In Place
New Array
Y-axis
X-axis