Randomized algorithms
HW3 – Matrix multiplication optimization

Ildar Nurglaiev

General Task:

1. Optimize matrix multiplication

2. Show that you can identify shortest paths in $O(n^w \log(n))$ time using matrix multiplication

Source code:

https://github.com/ILDAR9/Randomized-Algorithms/blob/master/HugeMatrixMultiple/hw3.ipynb

The optimization methods used (or attempted)

1. Naively square matrix multiplication takes $O(n^3)$ time.

2. Strassen's algorithm can perform it in $O(n^{2.807})$ (deterministic)

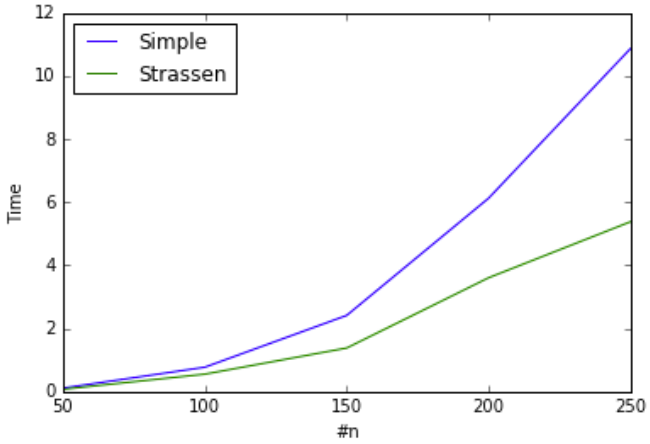
We can think of an n x n matrix as a 2 x 2 matrix whose elements are $\frac{n}{2}$ x $\frac{n}{2}$ matrices and apply the 2 x 2 algorithm recursively.

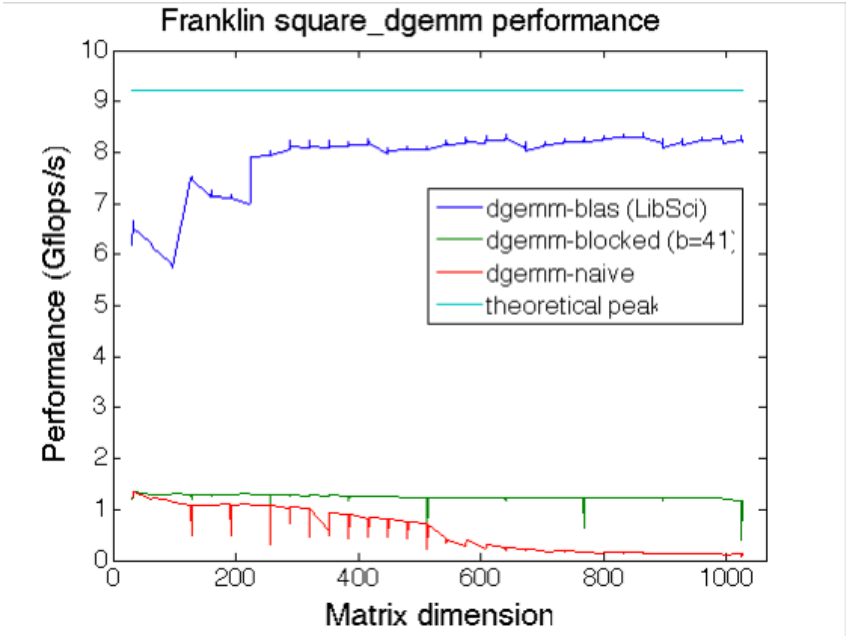
3. Winograd algorithm works in $O(n^{2.373})$, the implementation is on library.

The results of those optimizations

In general Strassen's algorithm works twice faster than naïve.

Recurrence Relation: $T(n) = 7T(n/2) + O(n^2)$
Runtime: $O(n^{\log 7/\log 2}) = O(n^{2.807})$





dgemm-naive - A naive implementation of matrix multiply using three loops
dgemm-blocked - A simple blocked implementation of matrix multiply
dgemm-blas - A wrapper for vendor's optimized implementation of matrix multiply

The reason for any odd behavior in performance in randomized approach

In modern computers the amount of external memory (e.g., disk storage or tape storage) has increased enormously, while RAM and computing speeds have increased, but at a substantially slower pace. Thus, we have the ability to store large amounts of data, but not in RAM, and we do not have the computational ability to process these data with algorithms that require super-linear time. A related motivation is that input-output rates have not increased proportionally. Thus, the size of the data inputs (as limited, e.g., by the size of disks) has increased substantially faster than the rate at which we can access the data randomly.

Show that you can identify shortest paths in $O(n^w \log(n))$ time using matrix multiplication

Input:
Let A be the adjacency matrix, an n x n matrix.

Output:
Let D be the distance matrix, an n x n integer matrix with d_{ij} representing the length of the shortest path from vertex i to vertex j in the graph G.

For x from 1 to $\log_2 n$:
$D \leftarrow D * D$

Thus O(M(n)logn)

The fastest version runs up to $O(n^{2.373} log n)$