Luvisutto Alberto, Algorithmic design exam

Homework: Matrix multiplication

Ex.1

Generalize the implementation to deal with non-square matrices

This implementation can be found in the `non_square_matrices` directory inside the github repository.

This exercise is solved re-conducting the "non-square matrices case" to the "base case" implemented. For this reason, in the *strassen.c* file, inside the `strassen_matrix_multiplication()` function, a padding is performed to modify the dimension of the two matrices A and B, transforming them into square matrices whose dimension is a power of two. Consequently, the `strassen_aux_mem_improved()` function is called, and the resulting matrix C is unpadded in order to obtain the actual result of the multiplication.

A specific section of code has been added to the *main.c* function in order to test the correct functioning of this method on non-square matrices.

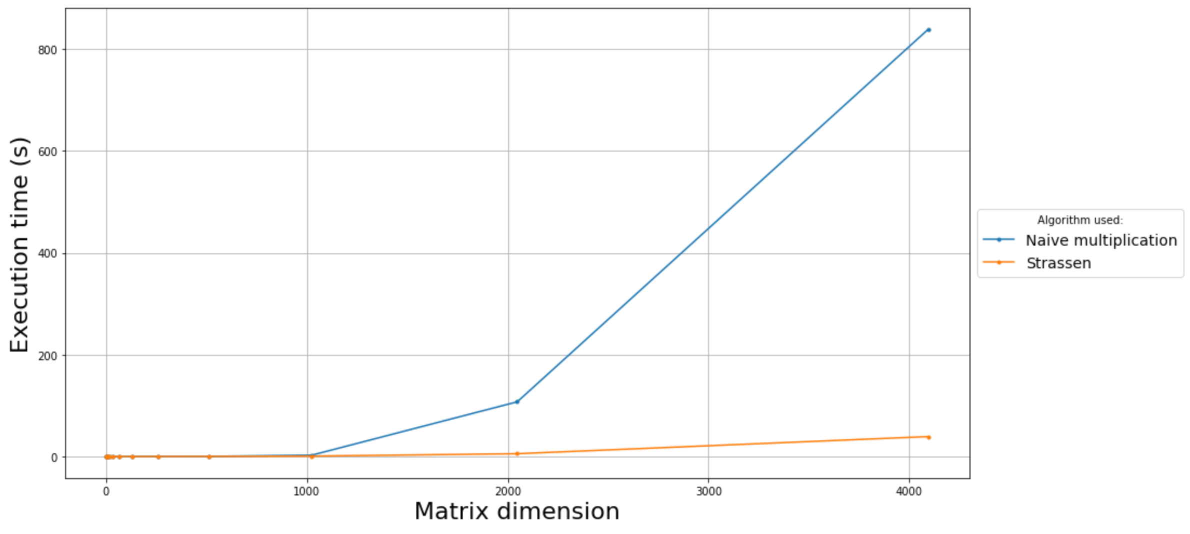
Ex.2

Improve the implementation of the Strassen's algorithm by reducing the memory allocations and test the effects on the execution time.

In order to reduce the unnecessary memory allocation, a smaller number of "intermediate" matrices S and P are generated and allocated; in particular, **two** matrices S and **three** matrices P are allocated. The deallocation of two of the three matrices P is performed as soon as they become unnecessary for the computation.

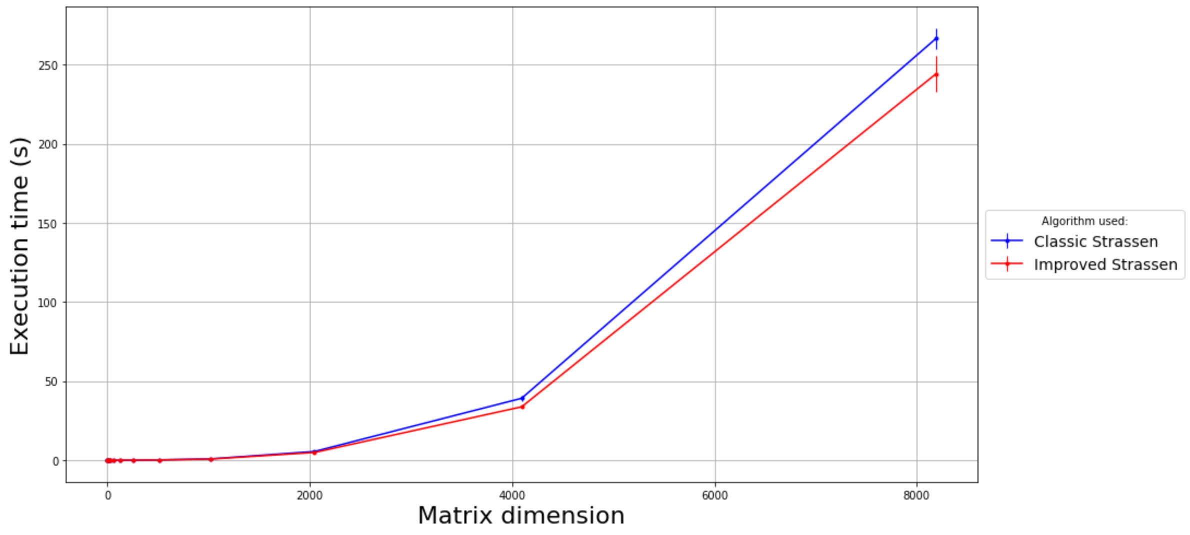
I report here some graphs showing the effects on the execution time; all the following graphs are obtained by applying the algorithms on matrices whose dimension is a power of two.

The following graph shows the difference in execution time between the naive implementation and the Strassen's algorithm:



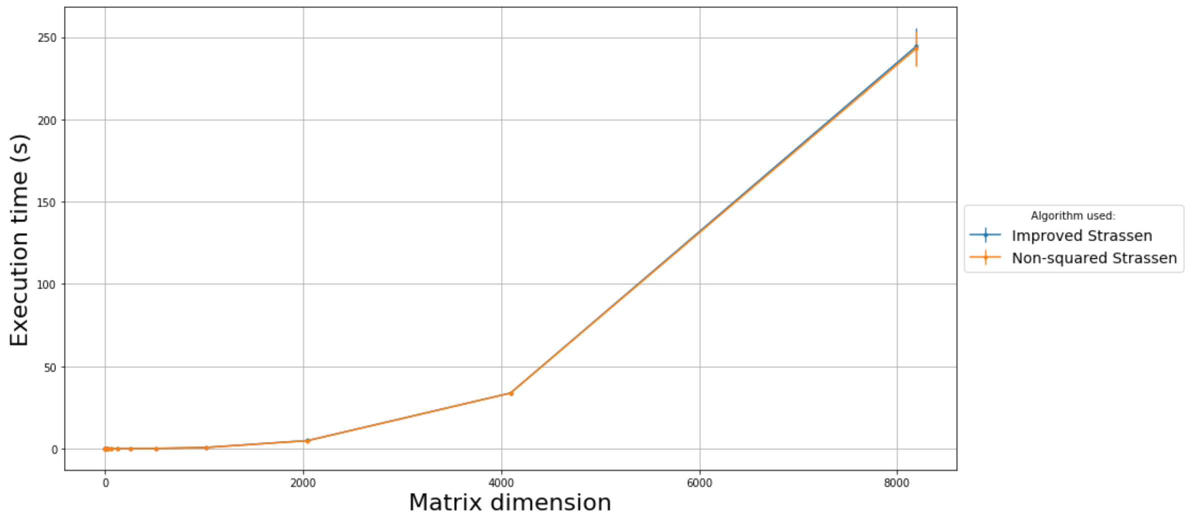
It is clearly visible that the Strassen's algorithm highly reduces the execution time.

The next graph shows the difference in execution time between two different implementations of the Strassen's algorithm: the "classic" one and the memory improved one.



It is again visible that the improved Strassen's algorithm clearly reduces the execution time. In this case the test is performed on one more instance of the problem (n = 8192) with respect to the previously considered ones, in order to show even better the positive effects of the reduced memory allocations. Observe also that error bars have been plotted in this graph, to show how much the execution time can vary with high values for *n* and randomly generated matrices, considering also that the computation is performed on a personal computer.

In this last graph I want to show that the execution time doesn't vary significantly when using the "non-squared" implementation of the improved Strassen's algorithm, even when the matrices considered are already squared and with dimension a power of two (so the padding of the matrices is actually not required).


Execution time (s)
Matrix dimension
Algorithm used:
Improved Strassen
Non-squared Strassen