FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION

ITMO UNIVERSITY

Report on the practical task No.8 (Extended). Practical analysis of advanced algorithms

Performed by

Alexander Yamoldin

J4134c

Accepted by

Dr Petr Chunaev

St. Petersburg

2021

Goal

Practical analysis of advanced algorithms

Book: *Thomas H. Cormen Charles E. Leiserson Ronald L. Rivest Clifford SteinIntroduction to Algorithms Third Edition, 2009 (or other editions).*

Sections:

I Foundations

4 Divide-and-Conquer

5 Probabilistic Analysis and Randomized

AlgorithmsVI Graph Algorithms

23 Minimum Spanning

Trees25 All-Pairs

Shortest Paths 26

Maximum Flow

IV Advanced Design and Analysis

Techniques15 Dynamic

Programming

16 Greedy AlgorithmsVII Selected Topics

Theoretical part

For completing the course I chose two algorithms.
The first one was from the Book: Thomas H. Cormen Charles E. Leiserson Ronald L. Rivest Clifford Stein, Introduction to Algorithms Third Edition, 2009 There is a **Strassen's algorithm** for matrix multiplication from the 4 section Devide-and-Conquer. The second one is **modified Vinograd-Strassen algorithm** for matrix multiplication [1]. In this laboratory work we will compare both algorithms between each other and classical matrix multiplication algorithm. Multiplication of two matrix (n×n size) A and B calling matrix C = AB that C = (c_{ij}) size of n×n in which $c_{ik} = \sum_{j=1}^{n} a_{ij} * b_{jk}$

What's the problem?
Classical matrix multiplication algorithm evaluates C = AB within the formula behind and complexity of the algorithm is $O(n^3) = O(\boldsymbol{n^{log_2 8}})$ and when n is big evaluation time becomes inadequate.

Solution
There is a **Strassen's algorithm** for matrix multiplication [2]. **Strassen's algorithm** is a recursive algorithm. The main idea is dividing each multipliable matrix on 4 matrix that the matrix multiplication C = AB represents as

$$\boldsymbol{A} = \begin{Bmatrix} \boldsymbol{A_{11}} & \boldsymbol{A_{12}} \\ \boldsymbol{A_{21}} & \boldsymbol{A_{22}} \end{Bmatrix}$$

$$\boldsymbol{B} = \begin{Bmatrix} \boldsymbol{B_{11}} & \boldsymbol{B_{12}} \\ \boldsymbol{B_{21}} & \boldsymbol{B_{22}} \end{Bmatrix}$$

$$\boldsymbol{C} = \begin{Bmatrix} \boldsymbol{C_{11}} & \boldsymbol{C_{12}} \\ \boldsymbol{C_{21}} & \boldsymbol{C_{22}} \end{Bmatrix}$$

Where

$$\begin{Bmatrix} \boldsymbol{C_{11}} & \boldsymbol{C_{12}} \\ \boldsymbol{C_{21}} & \boldsymbol{C_{22}} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{A_{11}B_{11} + A_{12}B_{21}} & \boldsymbol{A_{11}B_{12} + A_{12}B_{22}} \\ \boldsymbol{A_{21}B_{11} + A_{22}B_{21}} & \boldsymbol{A_{21}B_{12} + A_{22}B_{22}} \end{Bmatrix}$$

But we still have 8 multiplications. Do the **Strassen's trick**. Define a new variables

$$\begin{aligned} P_1 &= (A_{11} + A_{22})(B_{11} + B_{22}); \\ P_2 &= (A_{21} + A_{22})B_{11}; \\ P_3 &= A_{11}(B_{12} - B_{22}); \\ P_4 &= A_{22}(B_{21} - B_{11}); \\ P_5 &= (A_{11} + A_{12})B_{22}; \\ P_6 &= (A_{21} - A_{11})(B_{11} + B_{12}); \\ P_7 &= (A_{12} - A_{22})(B_{21} + B_{22}), \end{aligned}$$

Which will be used in

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} P_1 + P_4 - P_5 + P_7 & P_3 + P_5 \\ P_2 + P_4 & P_1 - P_2 + P_3 + P_6 \end{pmatrix}.$$

So in each step of the recursion algorithm does 7 multiplications matrix size of n/2 and 18 additions. Because of this trick the complexity of algorithm is $O(n^{2.81}) = O(\boldsymbol{n^{log_2 7}})$

This algorithm is useful for big no sparse matrix.
Algorithm works only with square matrix size of n×n where n is pow of 2

$$(\boldsymbol{n = 2^k, k = 0, 1, 2} \ldots .)$$

So in the first step we must check the size of multiplication matrix and if size n is not pow of 2 we must adding matrix by 1 in main diagonal and 0 in others position to the require size (n is pow of 2).
So this method is not effective for the small matrix, for example 5x5.
Because of to evaluate 5x5 we must adding matrix to size 8x8 ($\boldsymbol{n = 2^3}$)
and complexity of algorithm will $\boldsymbol{5^3 < 8^{2.81}}$ ($\boldsymbol{125 < 344.9}$). But for **all** $\boldsymbol{n >}$ $\boldsymbol{2^{\frac{log_8 7}{7}}} \approx \boldsymbol{30\_000}$ we have an advantage of using **Strassen's algorithm**.
I try to evaluate this size of matrix, but it takes a lot of time, so I don't have a time to deadline for evaluate it

The second algorithm is **modified Vinograd-Strassen algorithm** for matrix multiplication [1]. This algorithm is asymptotically faster than Strassen's algorithm. The time complexity of the algorithm is $O(n^{2.38})$ but in practice this algorithm more difficult for programming and require more resources than Strassen's algorithm. In this laboratory work we experimentally determine which algorithm is faster on sufficiently small sizes of multipliable matrices (n < 319)
The main idea of the **modified Vinograd-Strassen algorithm** is same as **Strassen's algorithm**, but we dividing each multiplier matrix by another method. We are evaluating submatrix $S_1, \ldots, S_8, P_1, \ldots, P_7, T_1, T_2$

$$
\begin{aligned}
S_1 &= (A_{21} + A_{22});\\
S_2 &= (S_1 - A_{11});\\
S_3 &= (A_{11} - A_{21});\\
S_4 &= (A_{12} - S_2);\\
S_5 &= (B_{12} - B_{11});\\
S_6 &= (B_{22} - S_5);\\
S_7 &= (B_{22} - B_{12});\\
S_8 &= (S_6 - B_{21});\\
P_1 &= S_2 S_6;\\
P_2 &= A_{11} B_{11};\\
P_3 &= A_{12} B_{21};\\
P_4 &= S_3 S_7;\\
P_5 &= S_1 S_5;\\
P_6 &= S_4 B_{22};\\
P_7 &= A_{22} S_8;\\
T_1 &= P_1 + P_2;\\
T_2 &= T_1 + P_4.
\end{aligned}
$$

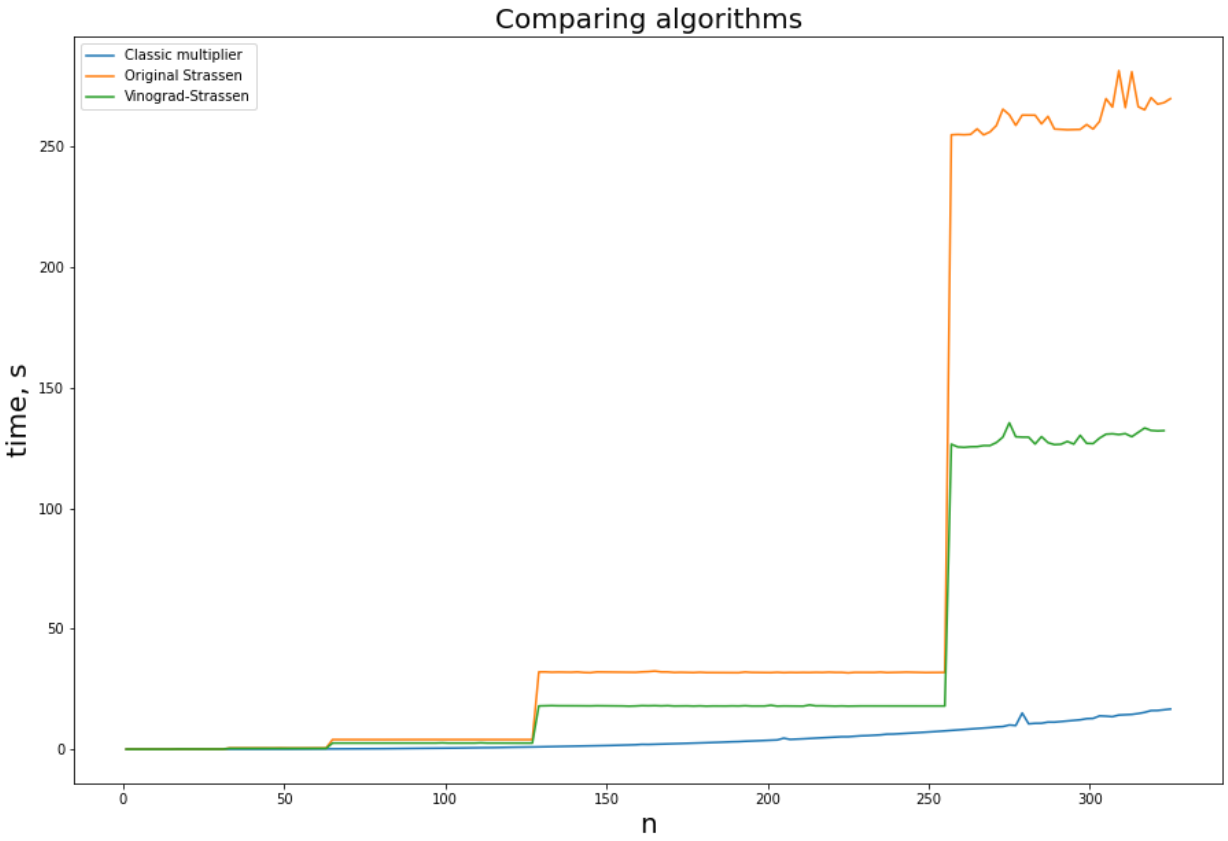
The elements of C matrix evaluates as:

$$
\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} P_2 + P_3 & T_1 + P_5 + P_6 \\ T_2 - P_7 & T_2 + P_5 \end{pmatrix}.
$$

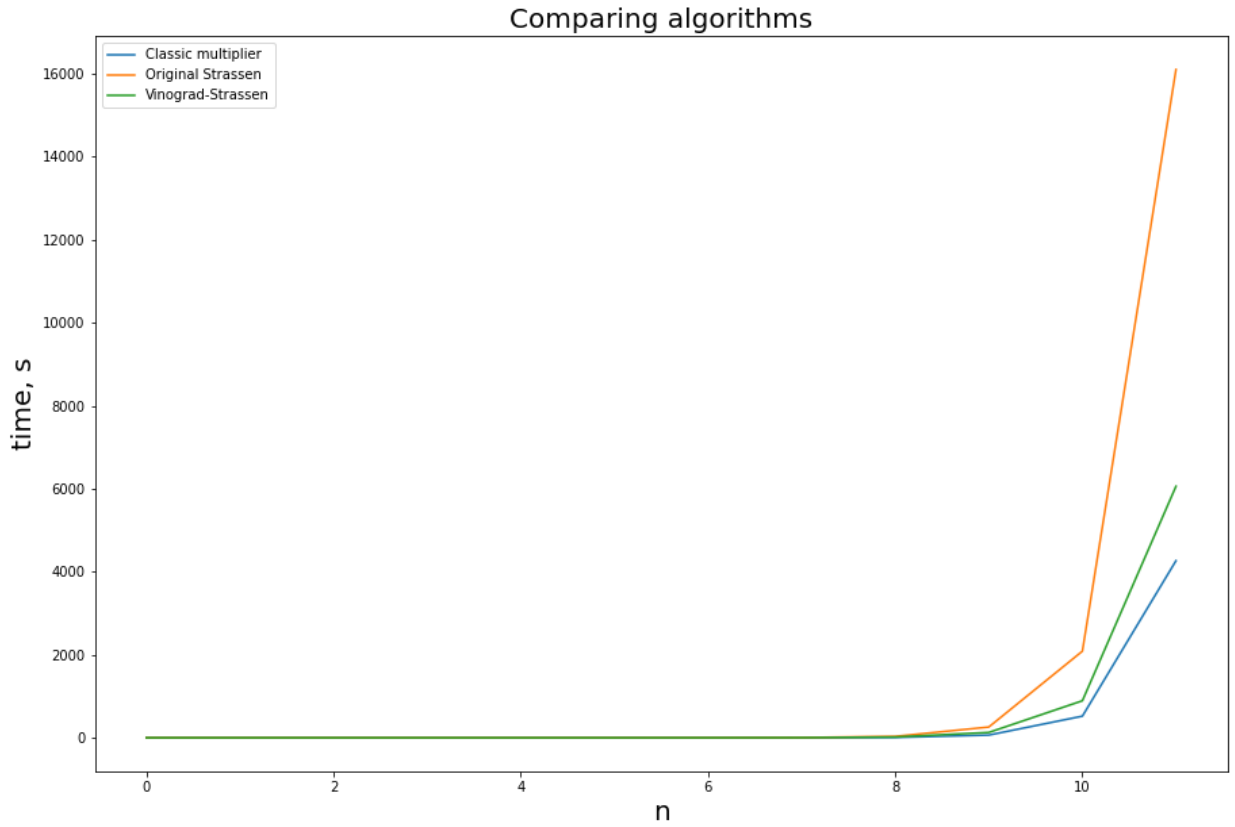
So by this suboperations we have 7 multiplier and 15 additions (in **Strassen's algorithm** was 18). Is that we are have performance gains.

RESULTS

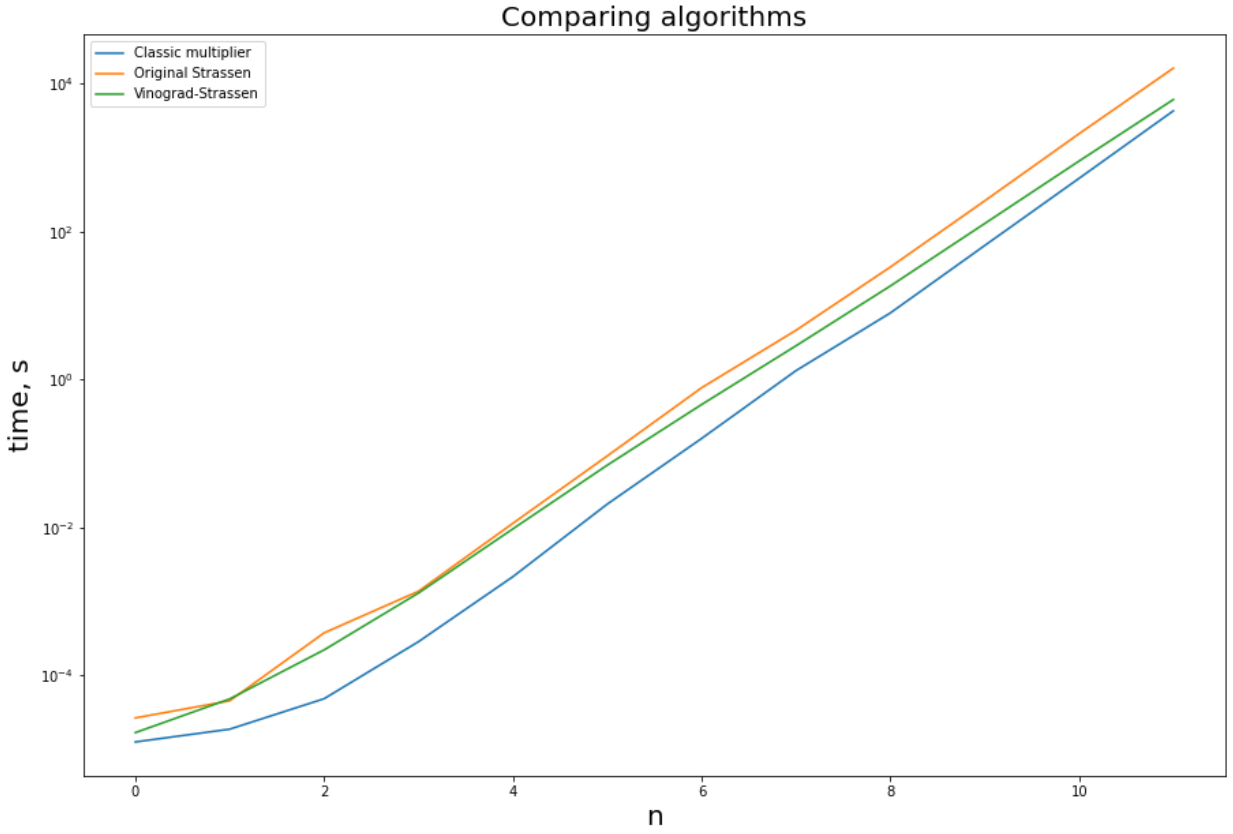
As I defined in theoretical part: *for* ***all*** $n > 2^{\frac{log_8 7}{7}} \approx 30\_000$ *we have an advantage of using* ***Strassen's algorithms.*** I experimentally **approved** this statement. In the picture below you can see that for the small size of the multiplied matrices < 350 which n is not pow of 2 (so algorithm need to adding matrix to the requirement size = n pow of 2) classical matrix multiplication algorithm works better than other. Notice, that steps increase in the size of the multiplied matrices = pow 2. This can be explained by the fact that a lot of time algorithm waste for adding matrix to require size and lose the performance on this step.



The idea of experiment was to compare complexity of algorithms the size of the multiplied matrices = pow of 2. My hypothesis was that Strassen's algorithm does not add matrix to require size, so we will se the performance gain of the algorithm on relatively small values of n: $n \in [n^0; n^{11}]$. The general view of complexity of algorithms you can see in picture below



Let's try to see on this graph on log scale. We can see, that asymptotically even on this small size of matrix Vinograd-Strassen algorithm (green line) has smaller slope angle tangent than classical algorithm (blue line). Thus, we can conclude that on large size of matrix the Vinograd-Strassen algorithm will be less time consuming than to the classical algorithm.



Conclusion

In this laboratory work I compare 3 algorithms of matrix multiplicate. There are **Classical**, **Strassen** and new modification **Vinograd-Strassen** Algorithm of matrix multiplication.

I did an experiment which duration was 18 hours. I experimentally defined theoretical statement: *for* $\boldsymbol{all\ n > 2^{log_8 7}} \approx \mathbf{30\_000}$ *we have an advantage of using* ***Strassen's algorithms.*** For better clarity and opportunity to see the results on the small size of the multiplied matrices I tried to multiplicate matrix with size only of pow of 2. This trick brought a result. First of all, we stopped lose the time on adding matrix to size pow of 2 and so we don't have steps. The second result is that even on small the size of the multiplied matrices we can see that asymptotically Vinograd-Strassen algorithm has smaller slope angle tangent than classical algorithm. Thus, we can conclude that on large size of matrix the Vinograd-Strassen algorithm will be less time consuming than to the classical algorithm.

Unfortunately, we can't show more expressive results of performance *using* ***Strassen's algorithms*** because I don't have enough resources to evaluating matrices of big size where $n \geq 30\ 000$. But we make a conclusion even on small size by observing the change in the tangent of the angle of inclination with the growth of the size of the multiplied matrices

Appendix

https://github.com/AAYamoldin/TrainingPrograms/blob/master/Python/ITMO_algorithms_lab/Task_8/Task8_Practical_analysis_of_advanced_algorithms(Extended).ipynb

1. Coppersmith D., Winograd S. Matrix multiplication via arithmetic progressions //Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, 1987, p. 1- 6.
2. Thomas H. Cormen Charles E. Leiserson Ronald L. Rivest Clifford Stein //Introduction to Algorithms Third Edition, 2009, p. 75-82.