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 Stein Introduction to Algorithms Third Edition, 2009 (or other editions).

*Sections:*

*I Foundations*

1. *Divide-and-Conquer*
2. *Probabilistic Analysis and Randomized Algorithms VI Graph Algorithms*

*23 Minimum Spanning Trees 25 All-Pairs Shortest Paths 26 Maximum Flow*

*IV Advanced Design and Analysis Techniques 15 Dynamic Programming*

16 Greedy Algorithms VII 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 = (cij) size of n×n in which

# What’s the problem?

# Classical matrix multiplication algorithm evaluates C = AB within the formula behind and complexity of the algorithm is O(n3) = O(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

# Where

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

# 

# Which will be used in

# 

# 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(n2.81) = O(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

# 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 ()

# and complexity of algorithm will . But for all 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(n2.38) 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

# 

# The elements of С matrix evaluates as:

# 

# 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 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: . 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 all 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 . 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.