National Research University
“Higher School of Economics”

Complexity of different multiplication algorithms

Research is conducted by Egorov A.
DSBA191-1
Supervisor: George Piatsky

26.04.20

Introduction

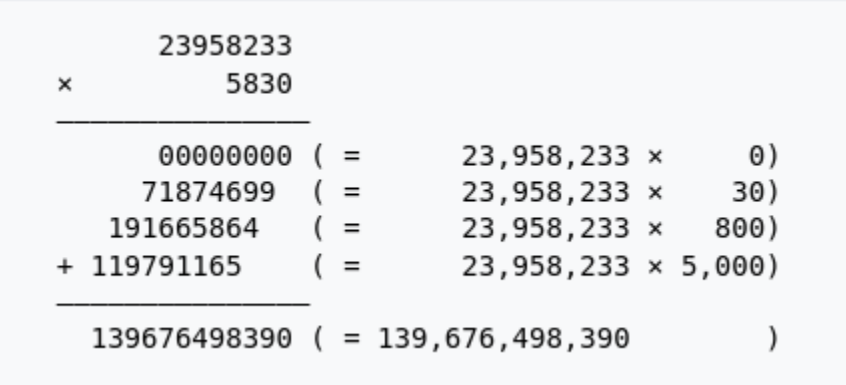
The aim of this research is to compare complexity of three approaches of positive integer multiplication: Grade School Multiplication, Divide and Conquer approach and Karatsuba algorithm. Let me provide a bit of data on each of them.

1) Grade School Multiplication

The most obvious way to multiply two numbers. An example is taken at https://en.wikipedia.org/wiki/Multiplication_algorithm:

The theoretical complexity of this algorithm is $O(n^2)$ [A. Karatsuba, 1995]

2) Divide and Conquer



It is a general approach of solving different types of problems recursively, which consists of three steps:

1. Divide problem into a number of subproblems that are smaller instances of the same problem.

2. Conquer the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner.

3. Combine the solutions to the subproblems into the solution for the original problem. [CLRS, p. 65]

The complexity of this algorithm is considered $Omega(n^2)$ [ADS, S. Shershakov, 19.03.20]

Multiplying arbitrary numbers x and y, one should apply the following formula: $xy = (a * 10^{n/2} + b)(c * 10^{n/2} + d) = ac * 10^n + (ad + bc) * 10^{n/2} + bd$, where ac, ad, bc and bd are calculated recursively.

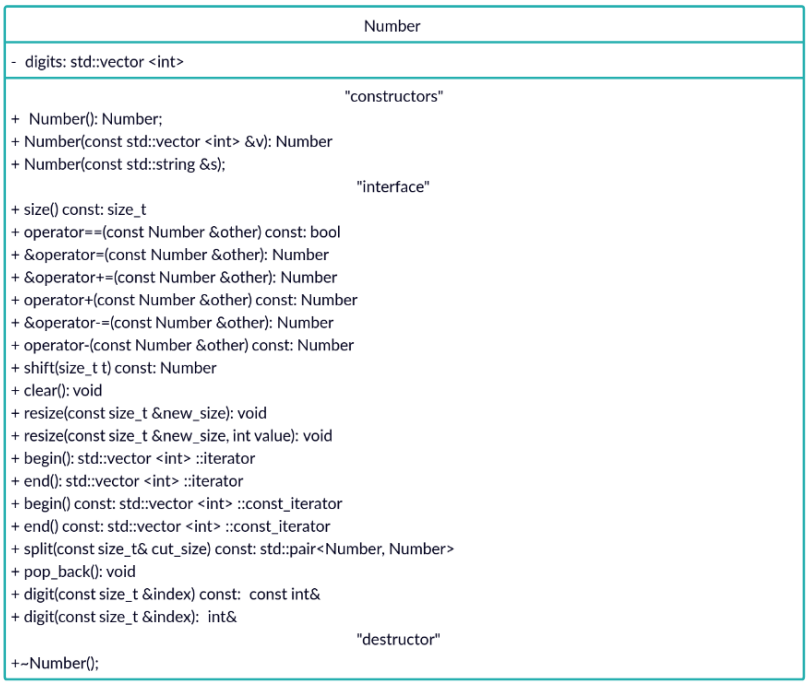
3) Karatsuba Algorithm

Based on Divide and Conquer approach, Karatsuba algorithm “is a source and a prototype of all fast multiplications.” [A. Karatsuba, 1995]. It was designed by Anatoly Karatsuba in 1960 and published in 1962. The author estimates its complexity as $O(n^{1.59})$, where the power of n is an approximation of $\log_2 3$. The scheme is the same as in the previous paragraph, but instead of four recursive calls, it makes three, combining the middle part into one multiplication, using that $ad + bc = (a + b)(c + d) – ac – bd$.

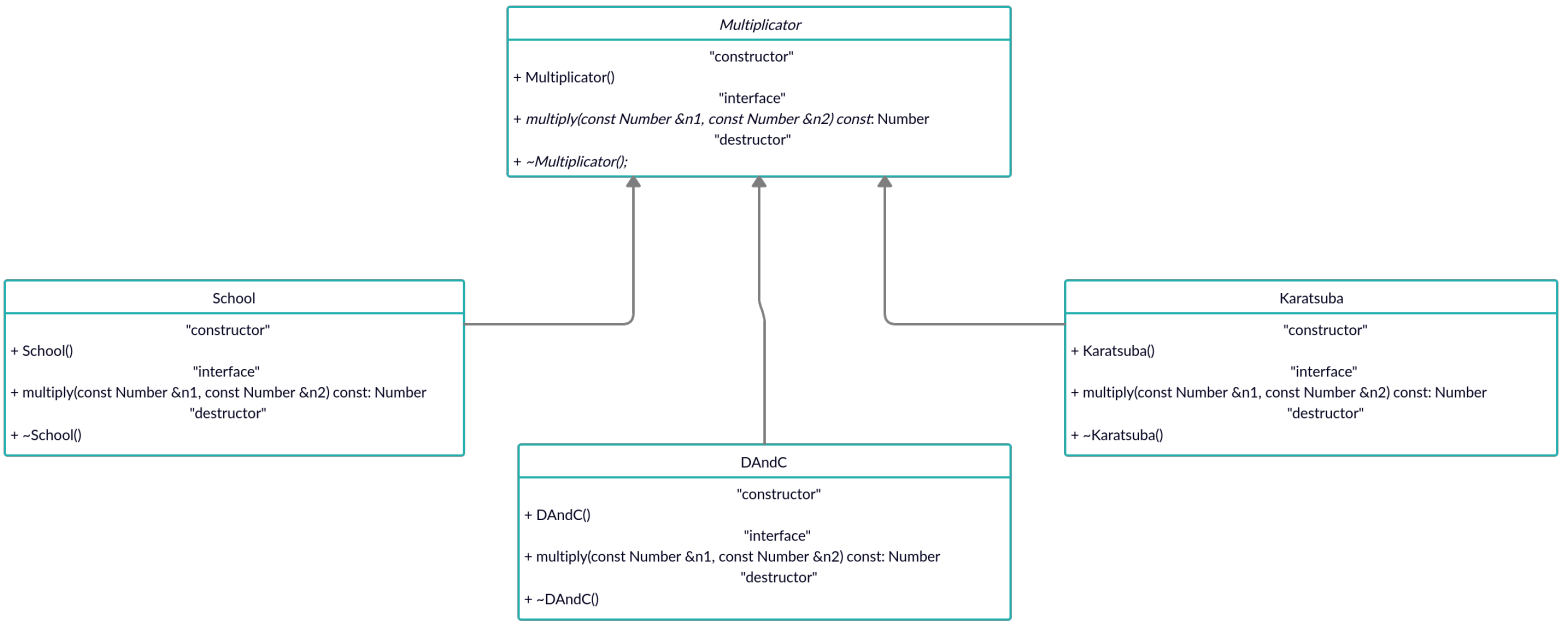
I am intended to point out, whether this theoretical knowledge correlates with data received during practical implementations of all of these algorithms. I design a C++ program which multiplies i-digit numbers for i from [1; 10000] using the three methods, measure how much it takes to perform each operation, and output the result to a file, after which I compare it to the theoretical estimation. I am going to perform calculations using my laptop with the following configuration: CPU – Intel Core i5-6300HQ @ 4x3.2GHz, 12 Gb RAM, OS - Manjaro 20.0 Lysia. The used IDE is QtCreator, the program is executed on Release build.

Implementation

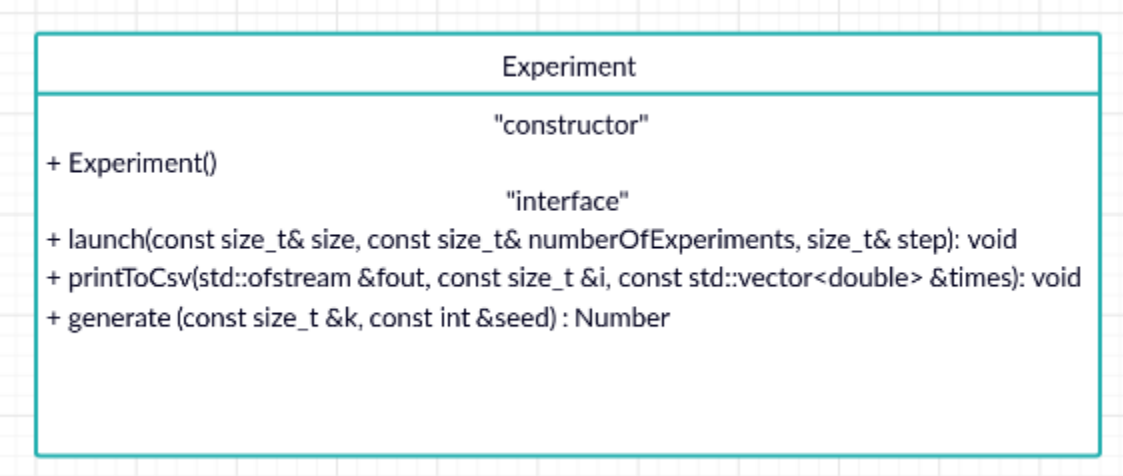
In this part of my report I am going to dedicate the reader to the process of creating a program which was used in order to perform this research.
First of all, I created class Number in order to store integers of big size and implemented basic operations on it.



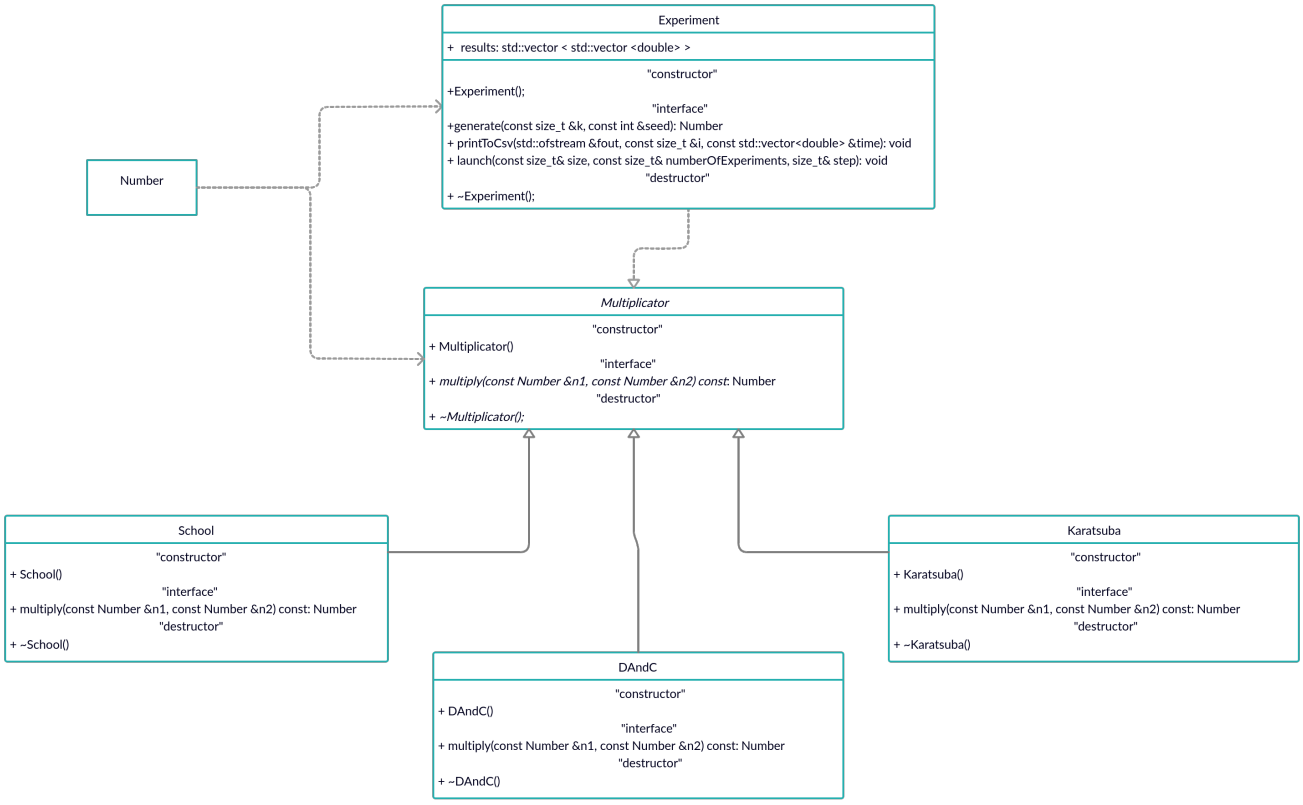
The following step is to create an abstract class Multiplicator, which would refer its only operation “multiply” to an implementation of that in one of the classes School, DAndC and Karatsuba, each named after the respective multiplication method.



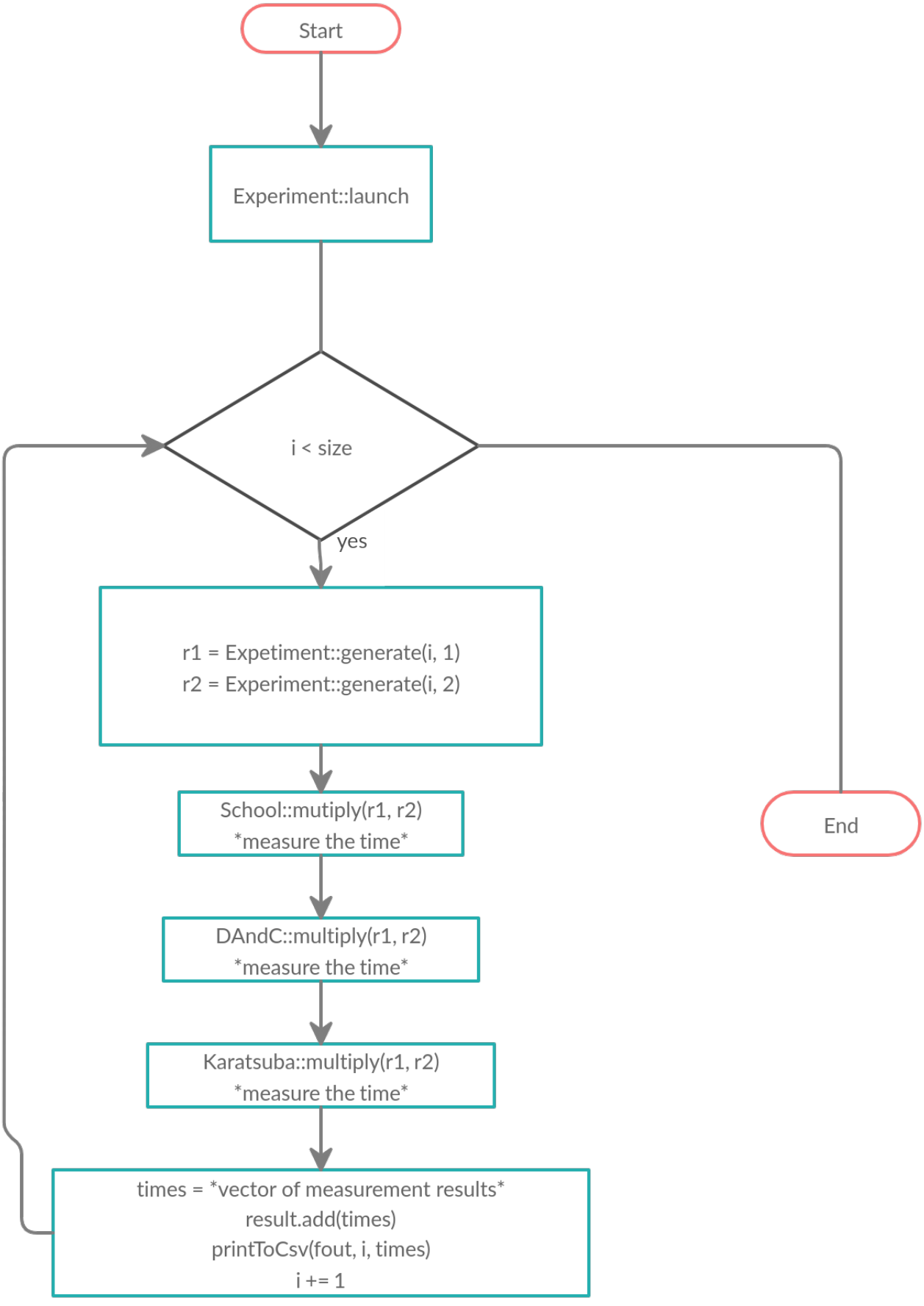
Next, I implement the class Experiment, which directly launches the calculations with its homonymous procedure. Parameter size – is the maximum size of digits, being the last multiplied, so the program multiplies 1-digit, 2-digit, …, size-digit numbers, numberOfExperiments is how many multiplications are performed at each step, step characterises the precision of graphs. Function “generate” generates a random k-digit integer.



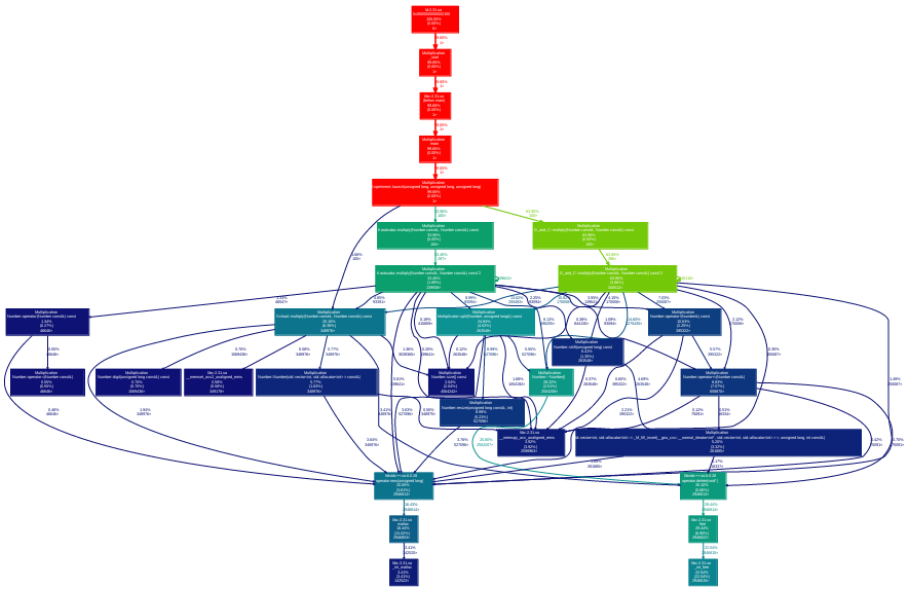
Overall, the interaction between classes looks like the following:



The function “launch” works in a following way: for all i from 1 to size it generates two i-digit numbers and then consequently multiply them using three algorithms, measuring time of each operation, printing them to a .csv file and storing the results. The approximate process is at the picture, for more precise version, please, address the code.


Start
Experiment::launch
i < size
yes
r1 = Expetiment::generate(i, 1)
r2 = Experiment::generate(i, 2)
School::mutiply(r1, r2)
measure the time
DAndC::multiply(r1, r2)
measure the time
Karatsuba::multiply(r1, r2)
measure the time
times = *vector of measurement results*
result.add(times)
printToCsv(fout, i, times)
i += 1
End

Here I would like to show how the program works with an experiment of size 1000 (one may address “utput.svg” for a more precise version)



Results

The main experiment is performed on up to 10000-digit integers with numberOfExperiments = 3 and a dynamic step = 1 for number sizes from 1 to 1000 and step = 10 further on. (figure one). It totally shows that on a set of small numbers the School Mutiplication algorithm is the most efficient, Karatsuba is slightly less effective, while D&C is counterproductive at all. (x = size of an integer, y = time in ms)

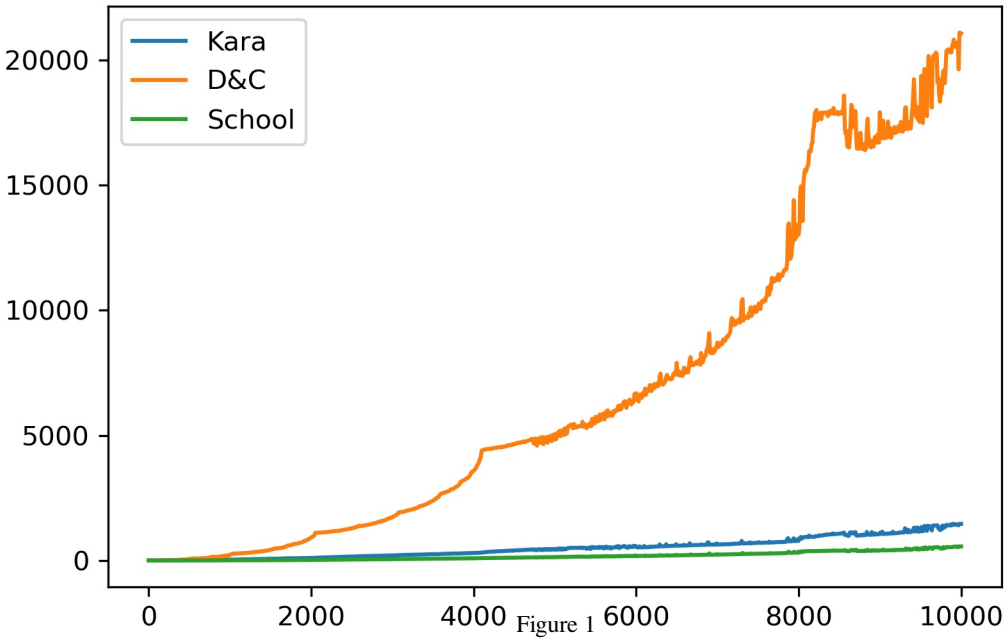


Figure 1

From the second graph we can clearly see, that D&C algorithm is indeed Omega(n^2) (x = size of an integer, y = time in ms):

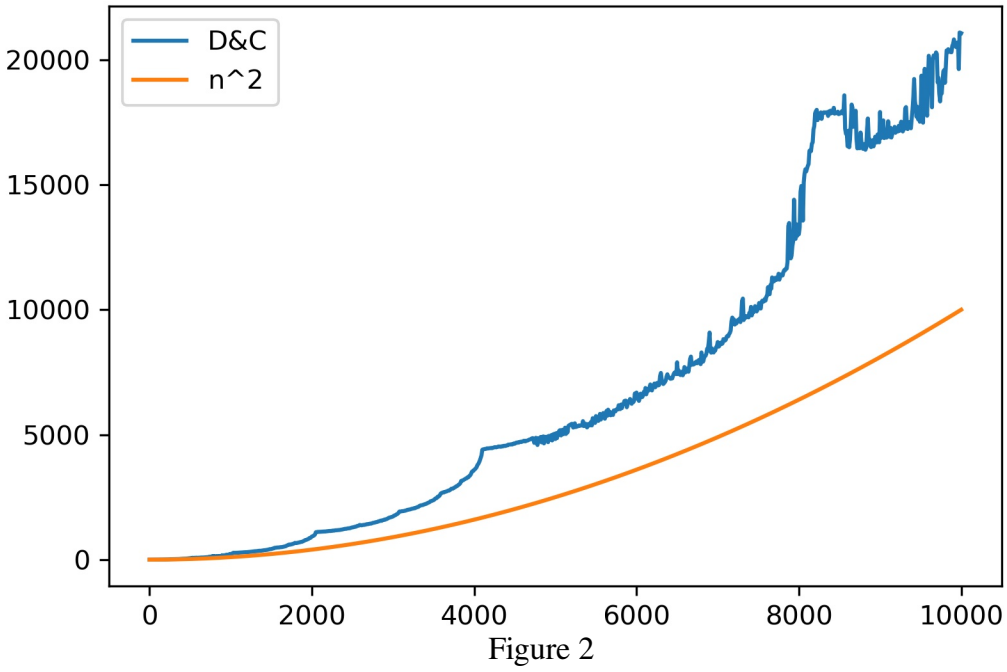


Figure 2

Moreover, one may make sure that Karatsuba algorithm is O($n^{1.59}$):

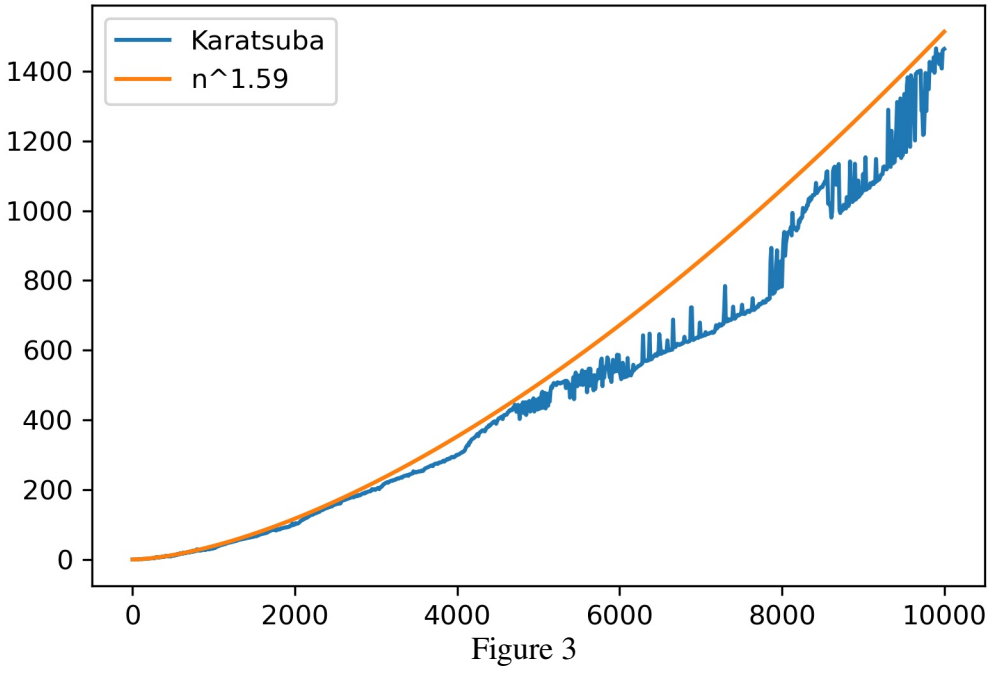
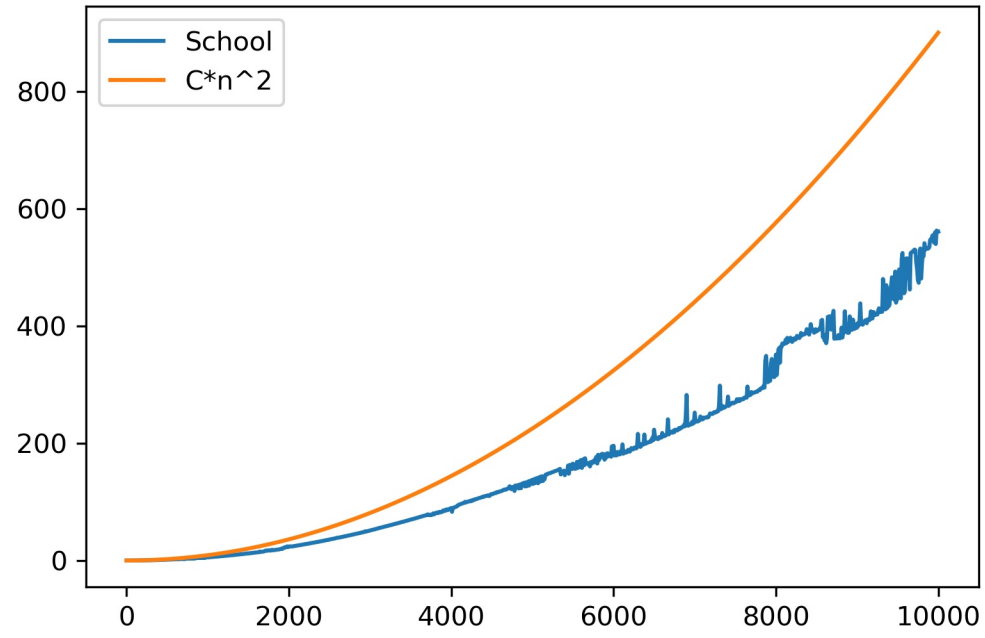


Figure 3

and School is $O(n^2)$:



I would like to address the attention to the third graph which is more precise comparison of School multiplication and Karatsuba algorithm:

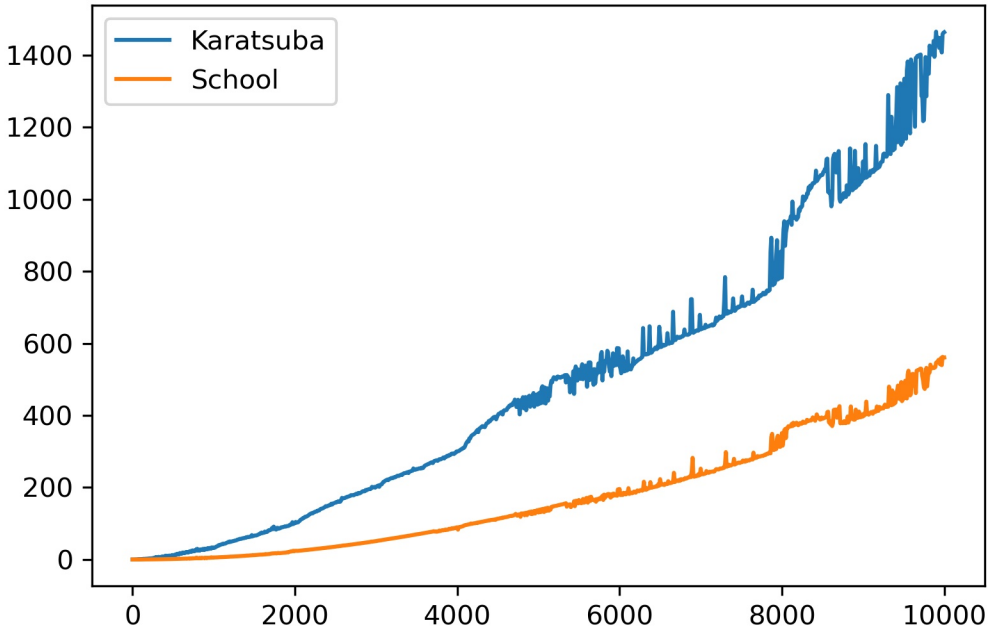
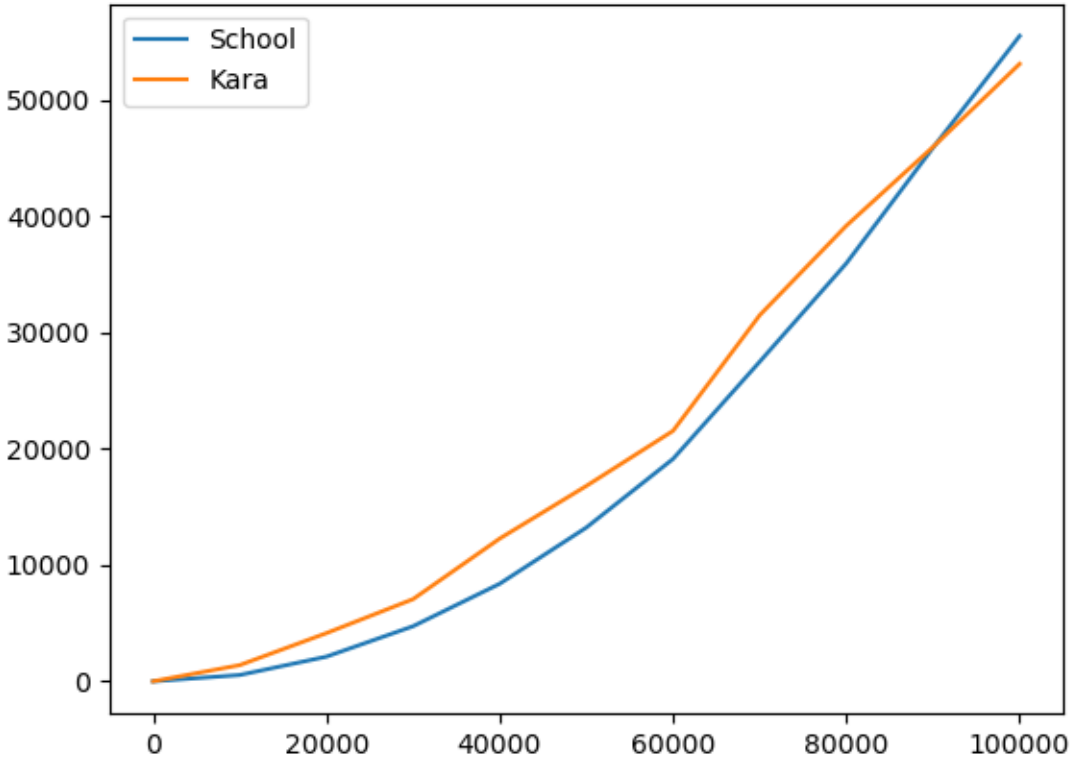


Figure 5

This graph shows that the first algorithm is much faster at small numbers (here I consider “small” to be 10000-digit), which contradicts with the theoretical proofs. That is why I decide to conduct another experiment (size = 100000, numberOfExperiments = 3, step = 10000) which would show that Karatsuba algorithm is faster at sufficiently large numbers.



As it is seen from the graph, for n ~ 87000, Karatsuba multiplication starts to overrun its rival, q.e.d.

Conclusion

This research has practically proved the theoretical claims about the complexity of mentioned algorithms. The work can be improved in the following way: making the time measurement a separate function, improving the implementation of Karatsuba algorithm and considering another approach for storing the digits of a big integer.

In this research I used the following theoretical materials:
https://en.wikipedia.org/wiki/Multiplication_algorithm
http://www.ccas.ru/personal/karatsuba/divcen.pdf
https://drive.google.com/file/d/1xzKmigw4mwtjbV-vnZEUB_bHgVDKY_mh/view (ADS Presentation)