Big Homework 1
Multiplication Algorithms Comparison

Conducted by: Kolobaev Dmitry, group 191-2
Supervised by: Shershakov Sergey

26.04.2020

Work objective: estimate and compare the complexity of three multiplication algorithms, namely "Grade School Multiplication" (GSM), "Divide and Conquer" (DaQ) and "Karatsuba Multiplication" by creating a custom implementation of these algorithms and analyzing the graphs, representing the time spent on calculations for different numbers size using different approaches.

The desired results: from the calculations, made during the seminars and provided in the CLRS and Coursera videos, we know, that the theoretical complexity of GSM, DaQ and Karatsuba are O(n^2), O(n^2) and O($n^{log_2(3)}$) ≈ O($n^{1.6}$), respectively. That is, GSM and DaQ are expected to have the same asymptotic complexity, which means that for some constant C,

$$C * T_{DaQ}(n) = T_{GSM}(n)$$

Meanwhile from our mathematical knowledge about logarithm, the logarithmic graph ($T_{Kar}(n)$) scales faster, than the quadratic for small n, and scales slower, than the quadratic for $n \to \infty$. Our experiment is held for $n \leq 10000$, and we expect it to be large enough for us to see that for some constants C_1, C_2

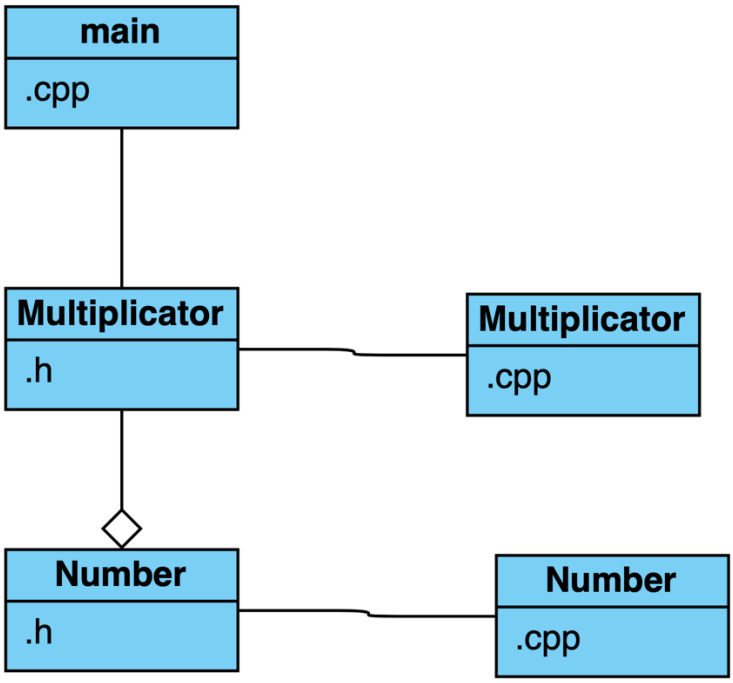
$$T_{Kar}(n) < C_1 * T_{DaQ}(n);\ T_{Kar}(n) < C_2 * T_{GSM}(n)$$

starting from some n.

Implementation details: in order to represent a big number I created a custom class, in which our number is stored as a reversed std::string object (e.g. a number 1234 is represented by std::string("4321")). Advantage of such representation is that std::container easily allows us to increase its size, so that we can store a bigger number after addition or multiplication. For example, if we are trying to sum 10 and 90, we create a container of size 2 ("00") and than extend it by 1 to receive ("001"). If we didn't use a reverse order string, we would have to transform ("00") into ("100"), which is a far more complexed procedure, requiring more time.

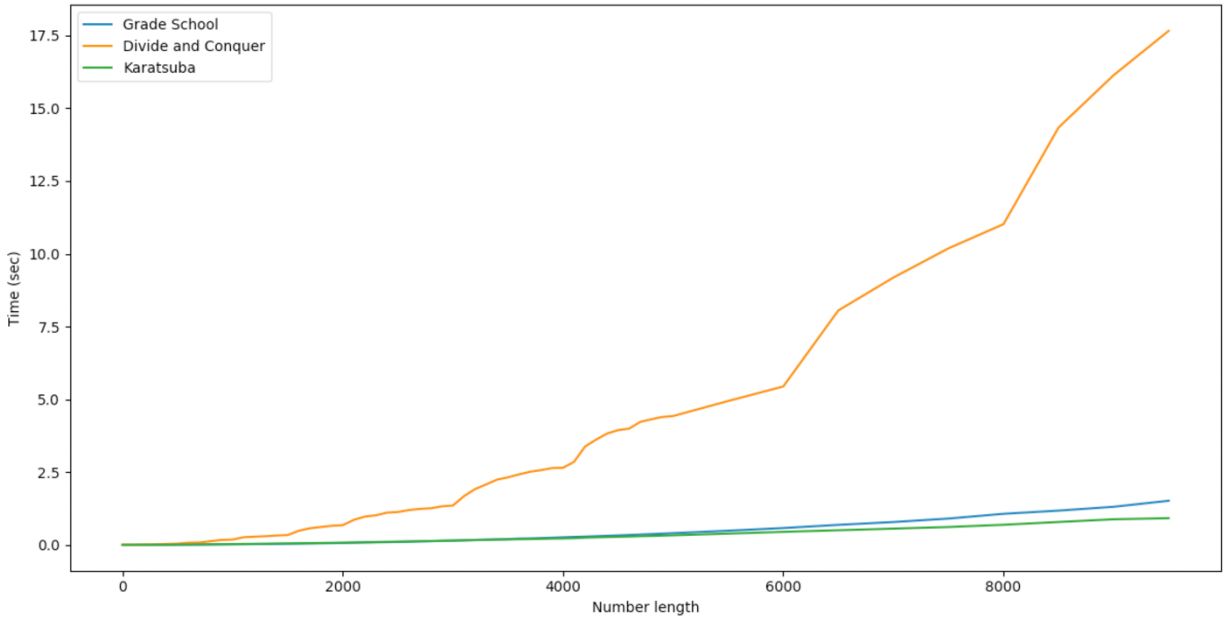
Another specific trait of my realization is that function for splitting one number into two doesn't result in numbers of exact size. This helps me avoid drastic changes in time when calculating, for example, product of numbers of size 1024 and 1025 or 4096 and 4097 by DaQ and Karatsuba algorithms.

Also, what should me mentioned about my realization is that for sake of decomposition classes Number and Multiplicator are each represented by a header file and a cpp file. The structure of my project is like this:

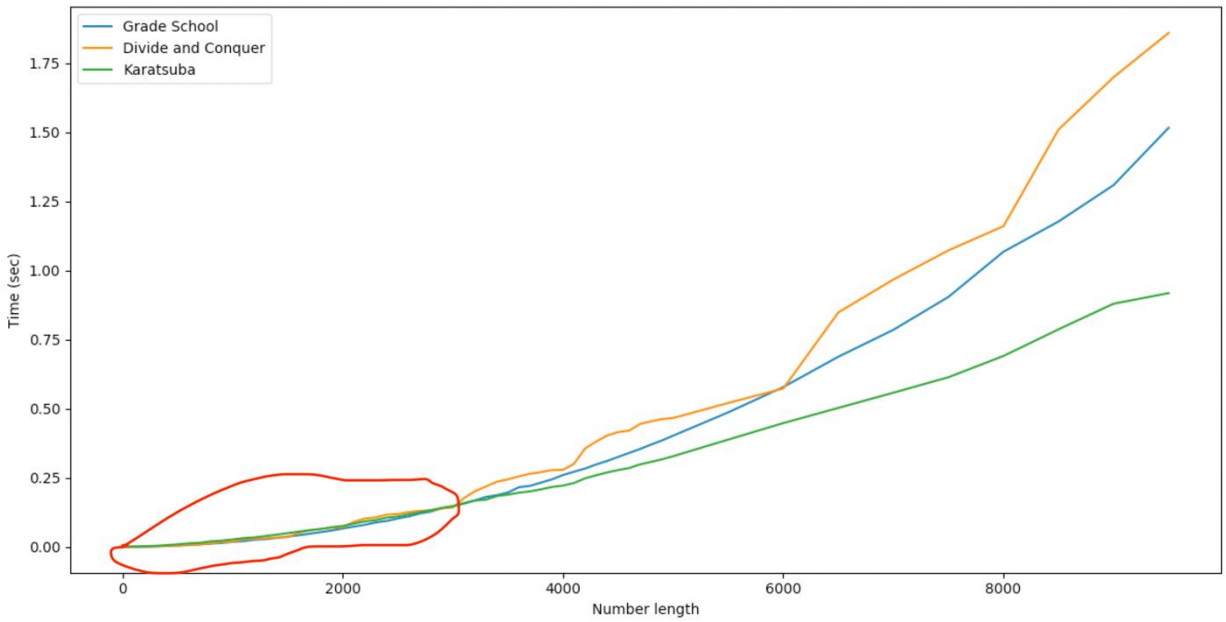


The code can be found by the following link: https://github.com/kol-di/dsba-ads2020-hw1.git

Results: below you can see the graph obtained



From this one it is not really evident whether our results satisfy our theoretical expectations. In order to examine it I divided all results of DaQ by an artificially selected constant ($\frac{1}{9.5}$). Obviously this doesn't affect the results of our asymptotic analysis. Here's the new graph:



Now we can see the following:

1) Graphs of GSM and DaQ algorithms are growing at approximately the same speeds, which means their complexity is the same O(n^2) according to calculations.
2) Speed of Karatsuba multiplication slower in in the beginning (circle by a red line), but faster afterwards, which also satisfies our theoretical expectations of its complexity being O($n^{1.6}$).

Conclusion: the theoretical results of complexity of different multiplication algorithms have been visualized and proved in practice. However, the speed of the multiplication is still to be improved. It is possible to be done by applying different Number representations, for example the ones based on a char or integer array or applying different tricks, like launching different algorithms for different problem sizes recursively.