Report

PROBLEM STATEMENT:

The problem of the current tase is to estimate time complexity of different multiplication algorithms (Grade school, Karatsuba and Divide and Conquer multiplication algorithms).

Theoretically, time complexity of Grade school algorithm is o(n^2) as well as Divide and Conquer. Meanwhile, time complexity of Karatsuba algorithm is o(n^$\log_2 3$). = o(n^1.59), which is faster than 2 other algorithms. This due to the fact that Karatsuba has only 3 recursive calls instead of 4 as in Divide and Conquer.

The research plan is to write c++ program, which multiply large integers using 3 algorithms, mentioned before. Then, measure working time of all algorithms, write it in csv file and build a plot, using it (visualize data). After that results can be easily analyzed and compared to theoretical.

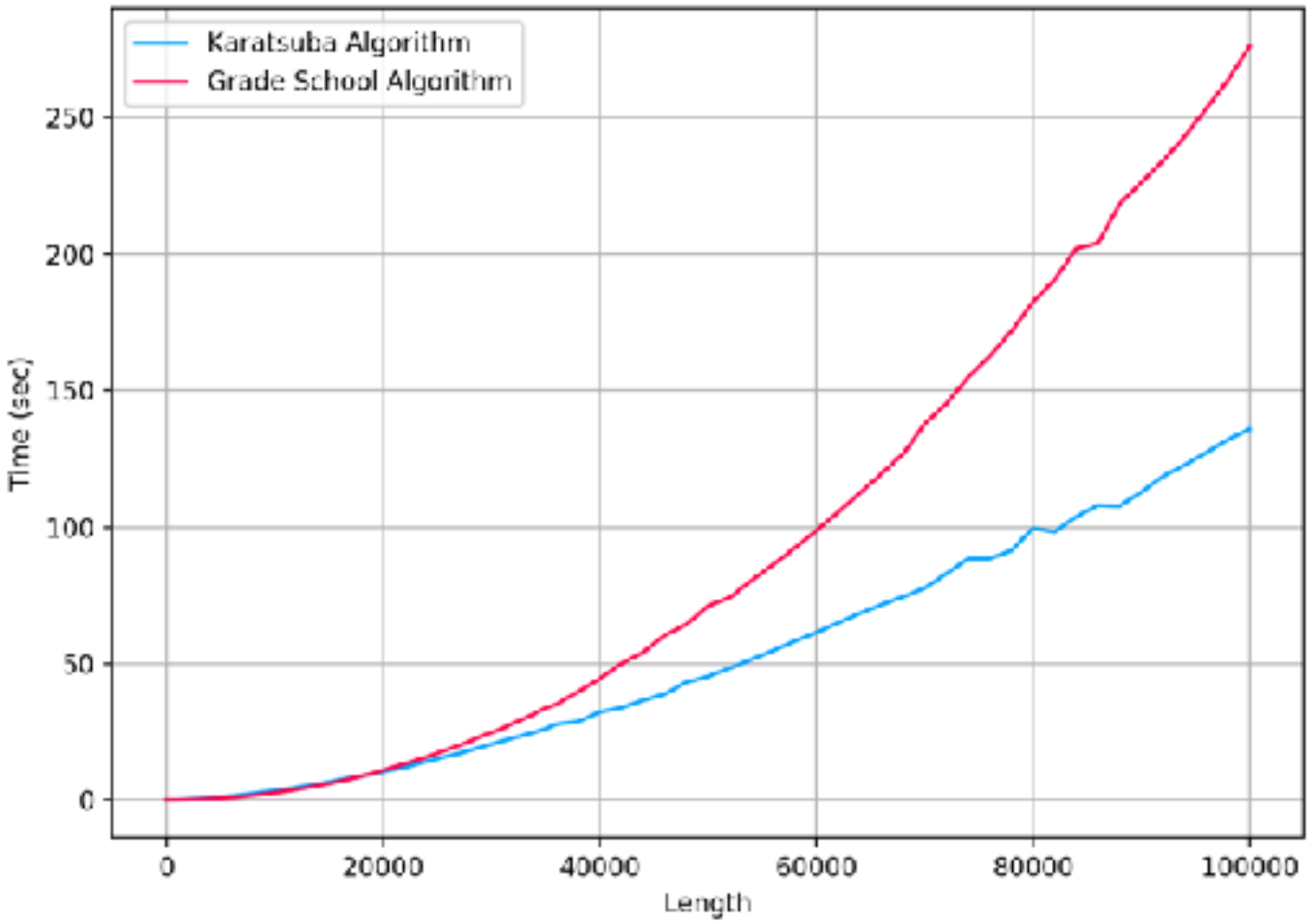
IMPLEMENTATION DETAILS:

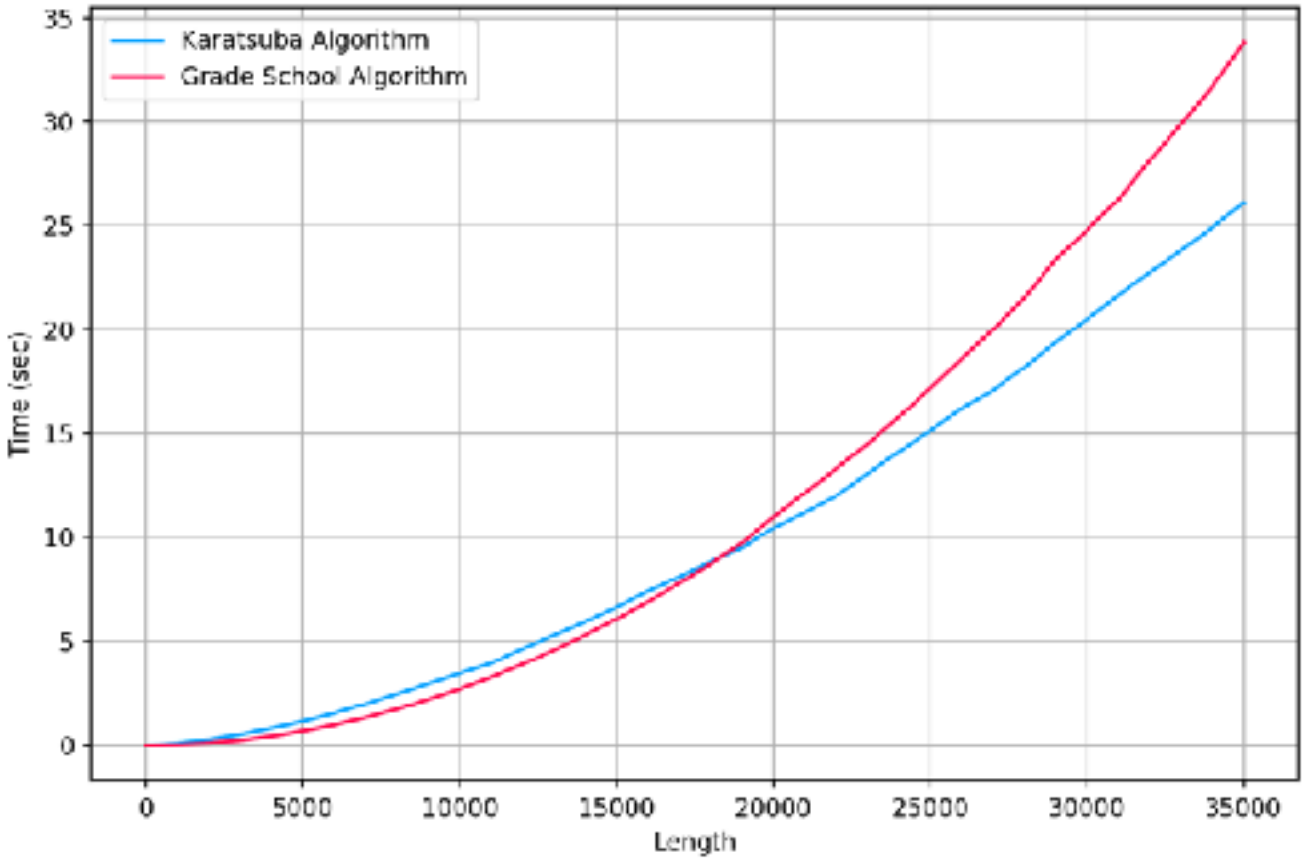
In program were created several classes: LargeInteger for storing large numbers and a class for each multiplication algorithm. LargeInteger stores numbers as a string and has several methods for simple usage of this class. Moreover, operators +=, -=, +, -, ==, !=, << were overloaded for convenient usage of the class. Some approaches were developed to improve working time of algorithms. For instance method split, which takes 6 arguments (2 numbers, which must be splitter and 4 variables for storing splitter subparts) in one cycle splits 2 numbersю It allow to avoid creating variables for storing same arguments of multiplication function, but as non-constant, to resize them. It is faster, than resize numbers and call 4 substring methods. Also, in the base case of Divide and conquer and Karatsuba algorithm I used special multiplication table, represented as 2d vector of LargeInteger (vector<vector<LargeInteger> > table), which stores product of the first and second index: table[7][3] = 7*3 = 21. It also boost algorithm proved by tests).

Each multiplication algorithm class is named by short name of algorithms (DAC for Divide and Conquer, KM for Karatsuba multiplication and GSM for Grade school multiplication). Each of these classes has method multiply, which take 2 parameters - numbers, represented by LargeInteger and return product of them also as LargeInteger. Each has method time, which take as a parameter, length of numbers, and return estimated time, required to multiply random numbers of this length. And also each class has method test, which test exact algorithm, and print errors if some occur.

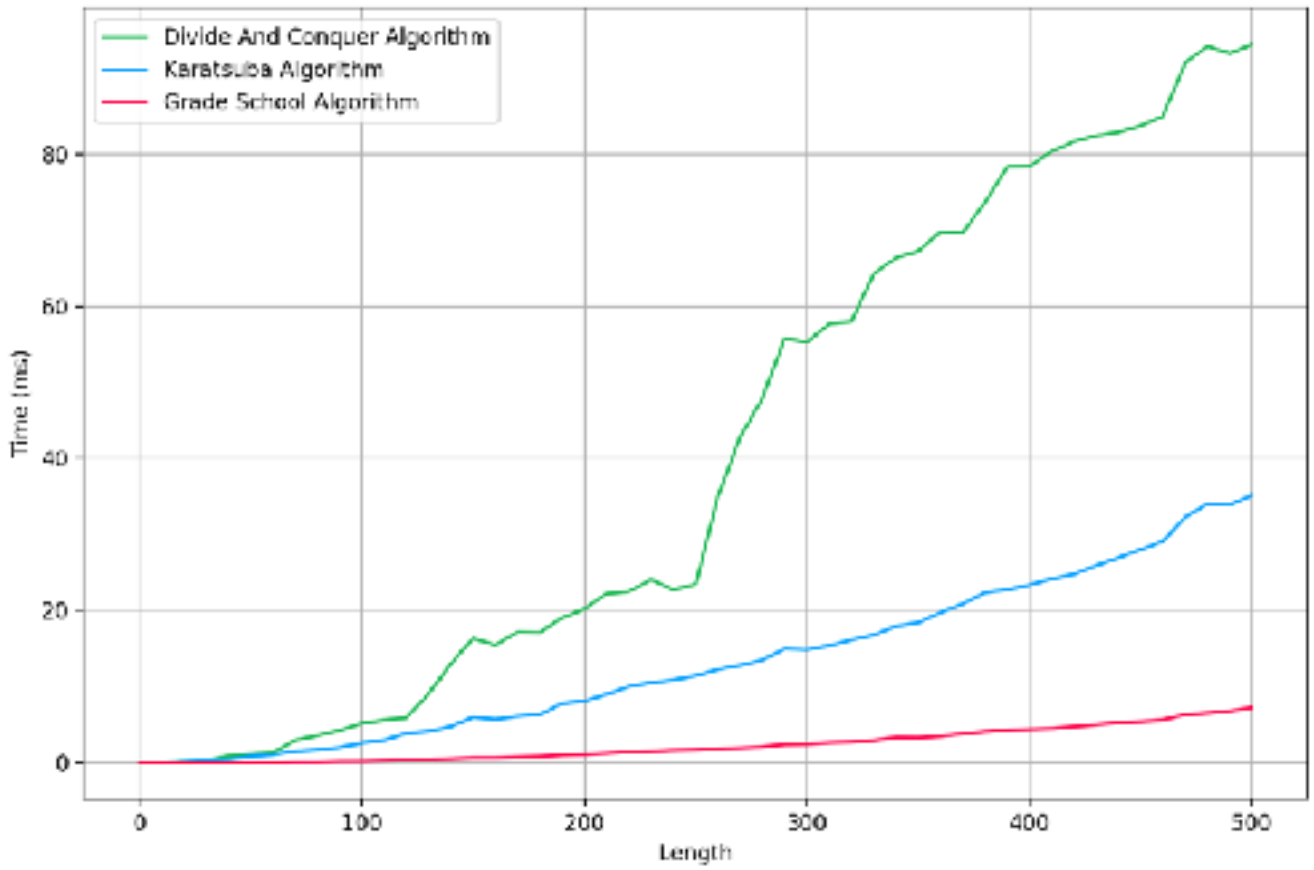
The project is located at git repository: https://github.com/a063mg/Multiplication.

RESULTS:
First 2 pictures represent Karatsuba algorithm, compared to Grade school algorithm on length range [0-35000] and [0-100000]: (This data stored in csv file: «100000.csv» and «35000.csv»)


Karatsuba Algorithm
Grade School Algorithm
Time (sec)
0
50
100
150
200
250
0
20000
40000
60000
80000
100000
Length


Karatsuba Algorithm
Grade School Algorithm
Time (sec)
0
5
10
15
20
25
30
35
0
5000
10000
15000
20000
25000
30000
35000
Length

Third picture represent Divide and conquer algorithm, compared to others on length range [0-1000]:Third picture represent Divide and conquer algorithm, compared to others on length range [0-1000]: (This data stored in csv file: «DAC.csv»)



As it can be seen on 1st and 2nd pictures, on small numbers (with length less than 20000) Grade school multiplication algorithm is the fastest, but starting from approximately 20000 length numbers (critical point), Karatsuba multiplication starts working faster. Graph of Karatsuba and Grade school algorithms are the same as theoretical one on large numbers. Divide and conquer Is the slowest algorithm, as it require more recursive calls. The graph of Divide and conquer is similar to n^2.

CONCLUSION:

Theoretical results differ from practical one. Grade school multiplication is the fastest, and Karatsuba works slower first time. On large numbers results are quite similar to theoretical, but not in case of Divide and conquer. Its complexity in practice is similar to n^2, according to the graph, but it is much slower than 2 others, even on large numbers. I still didn't found critical point for it, but maybe (I don't have much resources to verify it) on some very very big numbers it will be faster than Grade school algorithm, like Karatsuba.

Both Karatsuba and Divide and Conquer algorithms can be boost, by combining those algorithms with school one. Now, base case in DaC and Karatsuba is called for 1 digit numbers, instead of 30 for example. This will reduce time, required for counting product.

Recources
https://www.coursera.org/lecture/algorithms-divide-conquer/

https://sites.google.com/view/introprog19/materials?authuser=0#h.p_QxPQkqNM3wQ_
https://en.wikipedia.org/wiki/Divide_and_conquer
https://en.wikipedia.org/wiki/Multiplication_algorithm
https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9A%D0%B0%D1%80%D0%B0%D1%86%D1%83%D0%B1%D1%8B

Made by Alekseev Artem, group: 191-1. Supervisor: George Piatsky. Submitted at 26.04.2020.