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 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 allow to count product of 2 single digit numbers faster and thus boost algorithm (proved by tests).

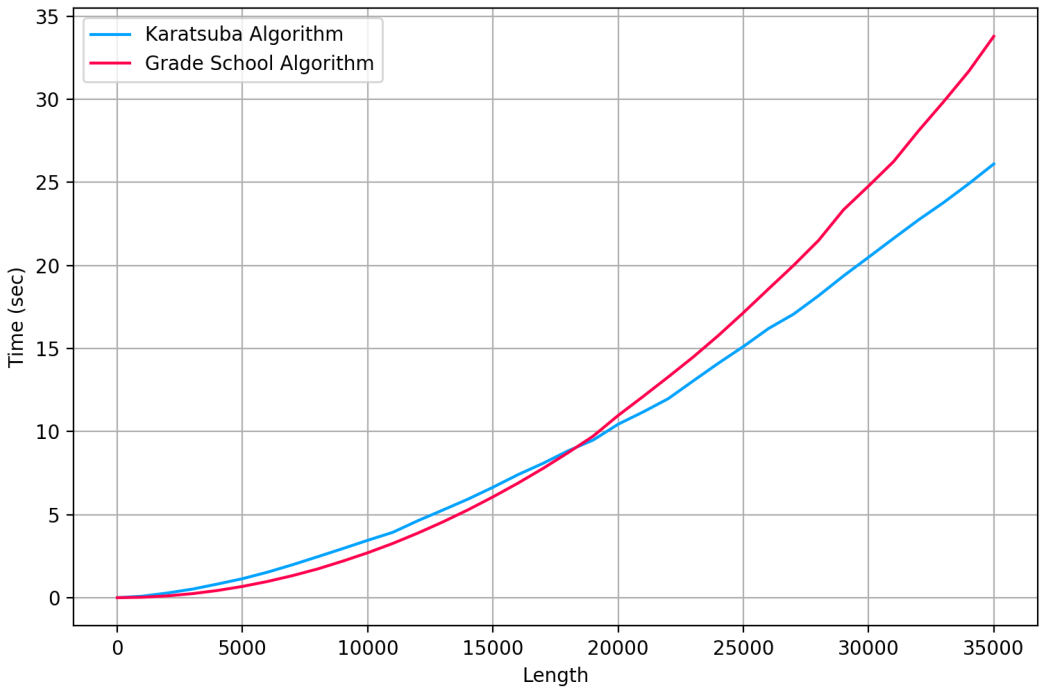
Each multiplication algorithm class, which all stored in *Multiplicator,h* is named by short name of algorithms (DAC for Divide and Conquer, KM for Karatsuba multiplication and GSM for Grade school multiplication) for convinience. Each of these classes has method multiply, which take 2 parameters - numbers, represented by LargeInteger and return product of them also as LargeInteger. Each has method time, which take length of numbers as a parameter 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. In this header file there is also a function, called *quick_test.* I has 3 parameters: range, step and attempts. This function test 3 algorithms on given range with given steps and count average mean from given number of attempts. It writes all received results in *output.csv* immediately.

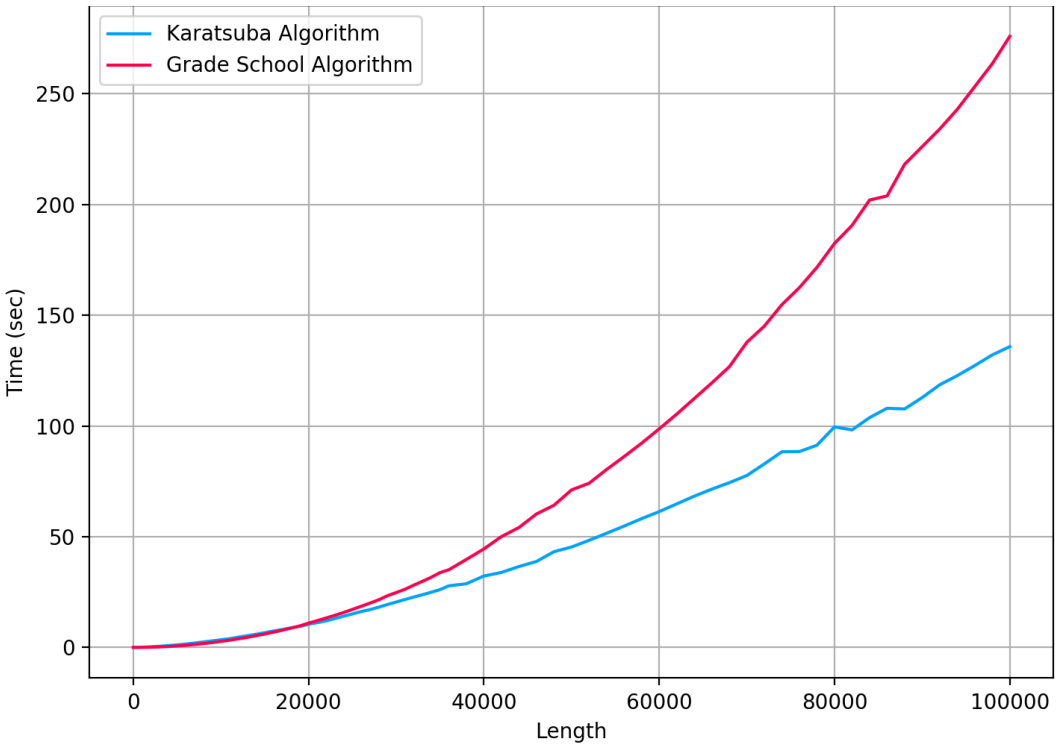
The project is located at git repository: https://github.com/a063mg/dsba-ads2020-hw1

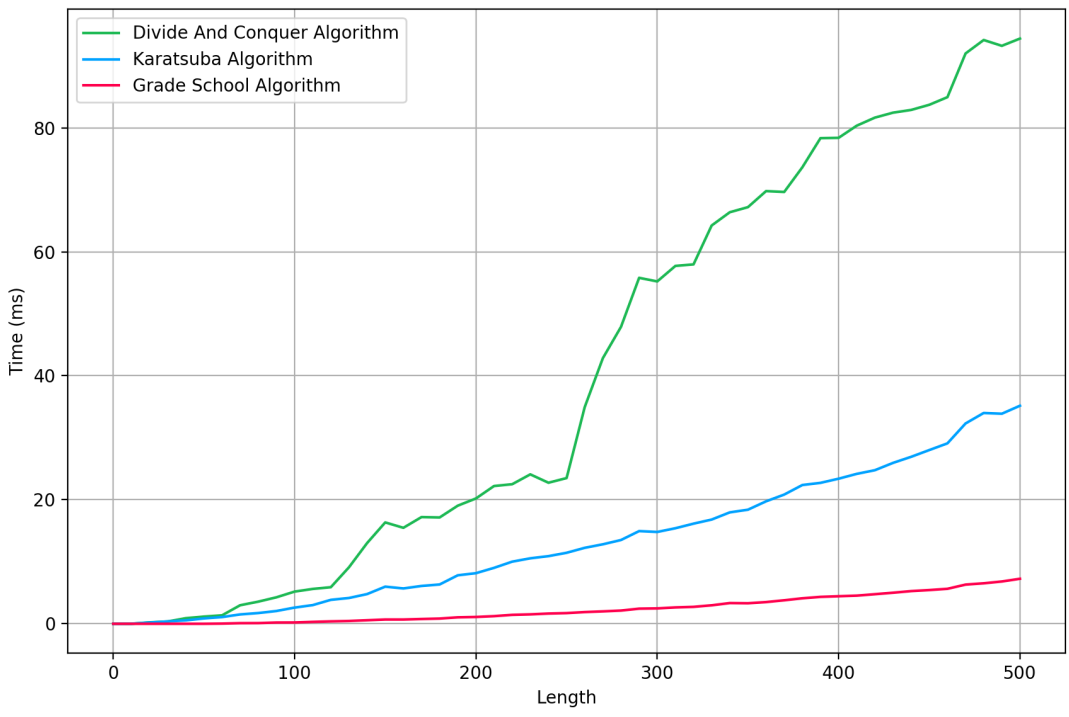
RESULTS:

First 2 pictures represent Karatsuba algorithm, compared to Grade school algorithm time results on length range [0-35000] and [0-100000] and step 1000 and 2000 respectively: (This data stored in csv files: «data/100000.csv» and «data/35000.csv»)

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


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


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


Divide And Conquer Algorithm
Karatsuba Algorithm
Grade School Algorithm
Time (ms)
Length
0
20
40
60
80
0
100
200
300
400
500

As it can be seen in 1st and 2nd pictures, on small numbers (with length less than 20000) Grade school multiplication algorithm is the fastest (There is no Divide and Conquer as it works too slow on such numbers), 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 algorithms, as require more recursive calls. His graph looks like n^2, but with some inaccuracies.

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 so-called greedy algorithm will reduce time, required for counting product. Another possible option for optimization Karatsuba and DaC is to realize multiplication algorithm as *= operator. Letting algorithm to resize one of the argument (which are both const now) will reduce number of variables, created in each recursive call which save memory and might increase speed.

As time tests shows, Divide and conquer is always the slowest algorithm, so there is no need to use it real life. Karatsuba should be used on numbers with length greater than 20.000 and Grade school for lesser numbers.

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

Work done by Alekseev Artem 191-1. Supervisor: Piatski George. Uploaded date: 26.04.2020