DSBA Algorithms and Data Structures

Homework 1

Report

Lysanyuk Kseniia
Group 191-2

Supervisor - Sergey Shershakov
April 2020

1. Problem statement

Task

The goal of the current task was to estimate the running time of different multiplication algorithms experimentally and compare with theoretical calculations. Three different approaches were used:

- Grade School multiplication;
- Divide and Conquer multiplication;
- Karatsuba multiplication.

Theoretical background

A widely used, but not so effective method is Grade School multiplication, which requires $O(n^2)$ elementary operations (adding and multiplying one-digit numbers). Basic Divide and Conquer approach also has the same complexity. Karatsuba, although it is also referred to as "Divide and Conquer" algorithm requires only $O(n^{\log_2 3})$, as it uses three recursive multiplications instead of four (A. A. Karatsuba (1995). "The Complexity of Computations").

Research plan

My research plan for the task is as follows:

1. Create a C++ project, implement a class Number representing big numbers, equip it with all necessary semantics;
2. Develop a class Multiplicator and three inherited classes, for each algorithm;
3. Generate random numbers of length from 1 to 10000 and measure the mean running time of all algorithms for each input;
4. Using Python to create a graph plot, illustrate asymptotic run time of algorithms which is to analyze in this report, compare the results with theoretical.

2. Implementation details

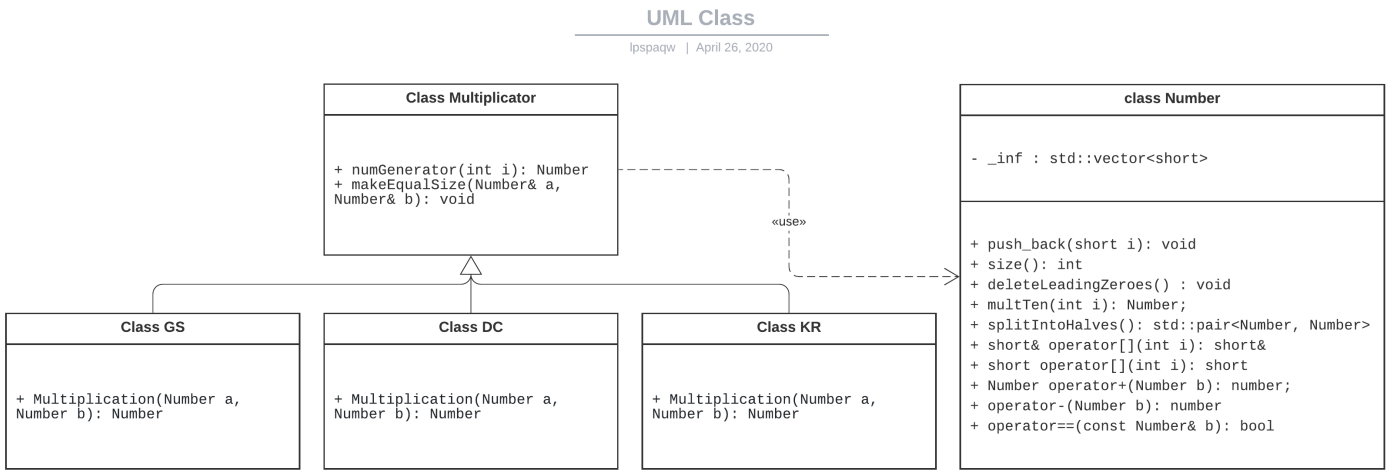

Figure 2.1: An UML Class diagram

Class Number

As C++ does not have in-built type for storing numbers with big number of digits, custom class Number was implemented. I have chosen std::vector of shorts for inner representation of those numbers to avoid conversions between types.

This class was equipped with such semantics as performing primitive operations:

- splitting into halves;
- adding and substrating two numbers using operator+ and operator- respectively;
- comparing using operator== (is not used in final version, was used for debugging only);
- referencing to individual components of a number using const and non-const overloads of operator[ ]

and some more (less important) ones.

Class Multiplicator

The parent class called Multiplicator provides auxiliary methods for calculating a product of two integer numbers:

- numGenerator generates a random Number of given length using rand();
- makeEqualSize is quite self-descriptive.

Sub-classes: SM, DC, KR

Classes SM, DC, KR are used to implement School Grade Multiplication, Divide and Conquer and Karatsuba respectively. They all have only one static method - multiplication itself, while auxiliary methods are inherited from the parent class Multiplicator.

Measurements

A function for measurements is importRes. In a loop it generates three pairs of random numbers. When the numbers are shorter than 100 the increment is 1, from 100 to 1000 it is 10, from 1000 to 4000 it is 50. Using chrono library it calculates time needed for each multiplication and puts mean results for three pairs to CSV file straightaway.

Repository URL

They say that picture is worth a thousand words, so for the whole program click here or go to the next url: `https://github.com/lysanyikkseniia/dsba-ads2020-hw1`.

3. Results and discussion

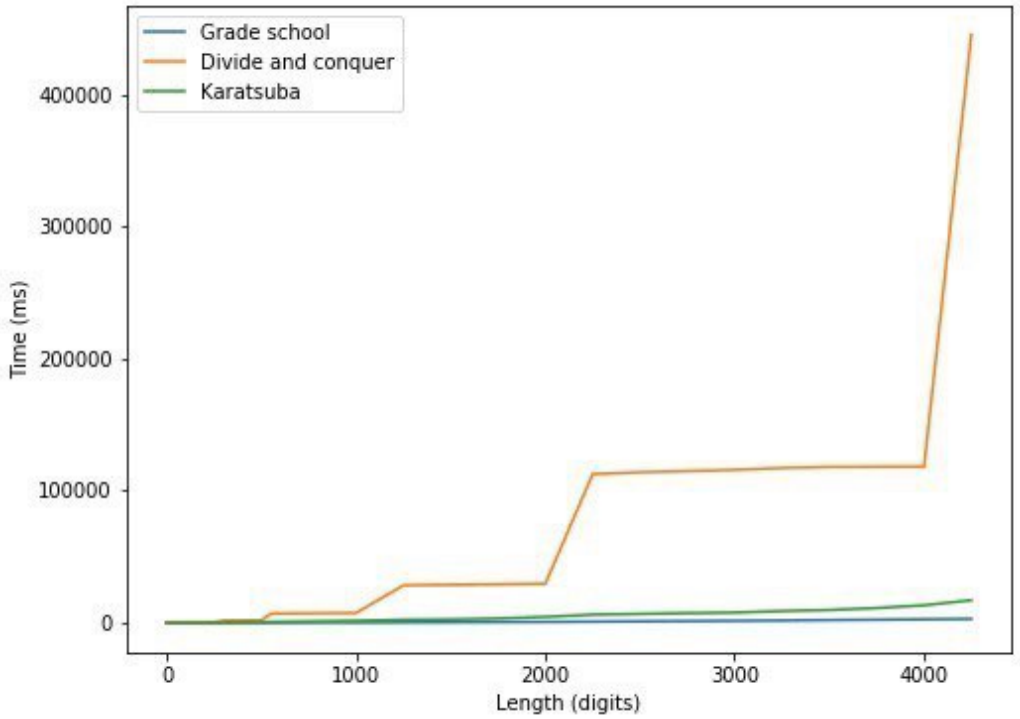

Figure 3.1: Running times

In the graph created by Python the result is clearly do not match with theoretical calculations. For "short" numbers (shorter than 50 digits) all three algorithms perform with almost same pace.

Divide and conquer approach stands from others being the slowest one. It also has "jumps" which are interchanged with almost linear plateaus.

Grade School Multiplication and Karatsuba are quite faster, although the last one shows a bit worse result.

4. Conclusion

Practical results stay far away from theoretical ones. The Grade School Multiplication showed the best results on both "small" and "big" numbers, while recursive algorithms turned out to be much slower. As the algorithms themselves are implemented correctly, it may be caused by poor implementation of class Number and lack of necessary semantics.

Further improvement opportunities:

- Implement the algorithms the way which is to avoid doing unnecessary copies and optimize code from point of memory consumption.
- Decompose method which calculates time and puts information into CSV-file.