Emil Akopyan

Group 191-2

26 April 2020

Workshop trainer:

Sergey Shershakov

Homework 1

Report

Problem Statement

The the main task of the project consisted in implementing 3 multiplication algorithms — the so-called Grade School Multiplication, Karatsuba multiplication algorithm and the Divide-and-Conquer approach. The goal was to test experimentally the running time of these algorithms and compare it with theoretical expectations.

Theoretical background

The time complexity of Karatsuba algorithm is:

$$T(n) = 3T(\frac{n}{2}) + O(n)$$

While that of the Divide-and-conquer algorithm is calculated by

$$T(n) = 4T(\frac{n}{2}) + O(n)$$

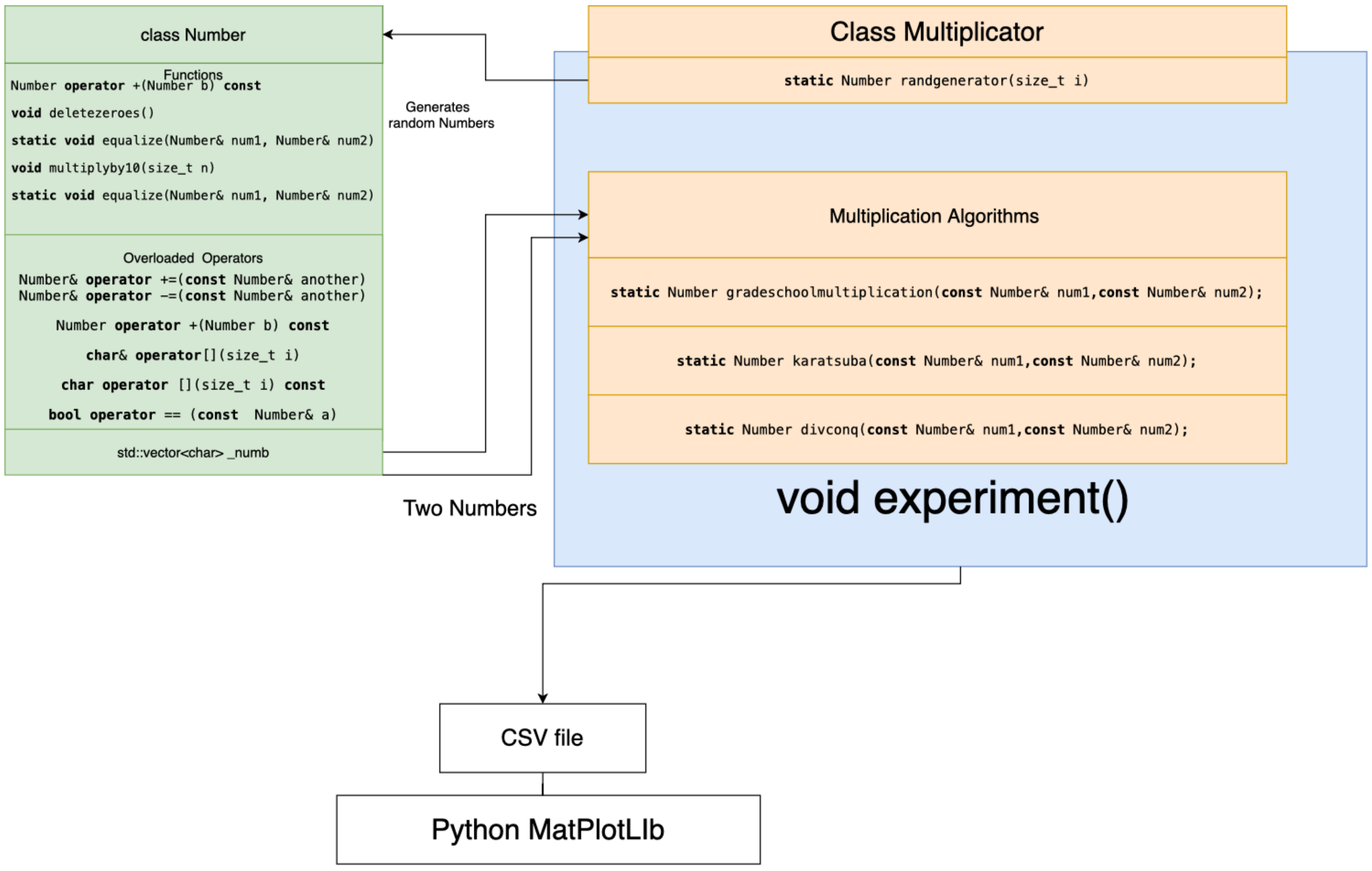
This partially explains the result we will see later. However, practical results do not completely meet expectations. The complexity of the GSM and DC algorithms is $O(n^2)$, while that of Karatsuba is $O(n^{\log_2 3})$[1]. In other words, it was expected that, starting from some point, Karatsuba would outperform the others..

[1] https://en.wikipedia.org/wiki/Karatsuba_algorithm

Research plan

1. Create a C++ programme which would test different algorithms and record their running time.
2. Use Python's Matplotlib to export and visualise the data.
3. Analyse the data, compare it with the theoretical expectations. Explain the results.

Implementation



```
Number operator +(Number b) const
void deletezeroes()
static void equalize(Number& num1, Number& num2)
void multiplyby10(size_t n)
static void equalize(Number& num1, Number& num2)
```

```
Number& operator +=(const Number& another)
Number& operator -=(const Number& another)
Number operator +(Number b) const
char& operator[](size_t i)
char operator [](size_t i) const
bool operator == (const  Number& a)
```

```
static Number randgenerator(size_t i)
```

```
static Number gradeschoolmultiplication(const Number& num1,const Number& num2);
static Number karatsuba(const Number& num1,const Number& num2);
static Number divconq(const Number& num1,const Number& num2);
```

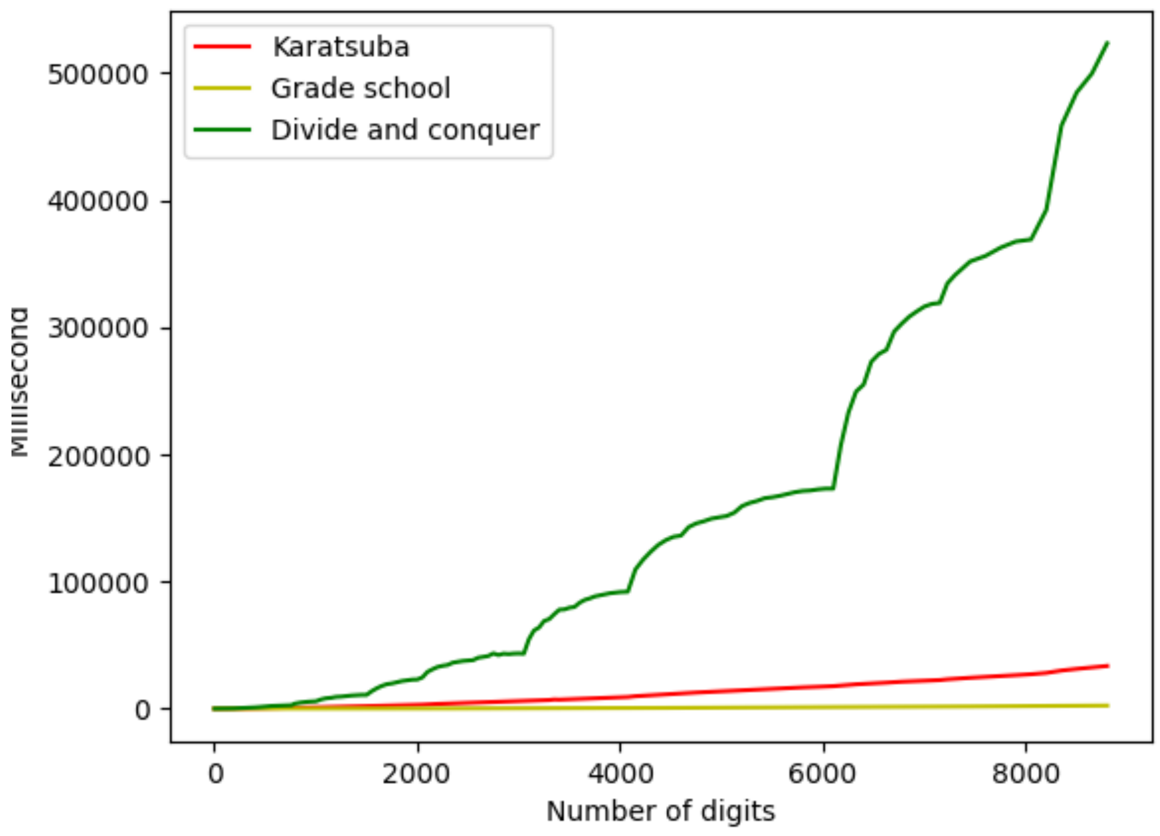
To do that, our strategy was the following. We created numbers of different length, ranging from 1 to around 9000, multiplied them 3 times with different methods and recorded the the mean. These measures were saved in a special CSV file, the information in which was subsequently visualised with MatPlotLib library for Python.

Technical Implementation:

Numbers are stored in the Number class, which has only one field of vector of chars. It contains individual digits in the reverse order, because it make the GSM algorithm more intuitive and clear and does not the others. All the multiplication algorithms are contained in the Multiplicator class. It also contains a static Number randgenerator(size_t), which generates random numbers of a given length.

Results:

Practical results do not completely meet the expectations. The GSM, it turned out, was the fastest among them, and even Karatsuba did not show better results at large numbers. The DC algorithms is certainly the slowest algorithm among them.



Conclusion

On numbers that I was able to compute on my computer, Karatsuba turn out to be slower than grade school multiplication. A better implementation would make it possible. The slowest of all three algorithms is the divide and conquer algorithm. I suggest that this can happen for the following reasons:

1. It is slowed by recursion, which implies a huge number of copying operations. From the formula for the time complexity we know that at each stage DC algorithm creates 4 subproblems, so that is why this problem is especially acute with DC algorithm.
2. If it receives numbers of different length, it will perform the number of calculations, necessary for the bigger one.

The first explains why Grade School Multiplication is the fastest algorithm dealing with numbers that were considered. The algorithm includes very simple operations which do not require any considerable resources, such as arithmetic of separate digits. At the same time , others include a huge number of costly operations, such as copying of classes. That’s why our implementation of Karatsuba algorithm fails to outperform the GSM. But still, it must happen on some very large number.

Repository
https://github.com/emlakp/dsba-ads2020-hw1