Algorithm analysis

By: Danila Kokin
Group: 193.1
Submission: 28th of April 2020
Supervisor: Aleksey Varenikov

Problem statement:
Implement a program, which can work with extremely large numbers, generate random ones, and the most important: implement 3 algorithms to calculate product randomly generated numbers of different lengths. The last, but not least is to perform graph to compare algorithms efficiency and estimate connection with theoretical base.

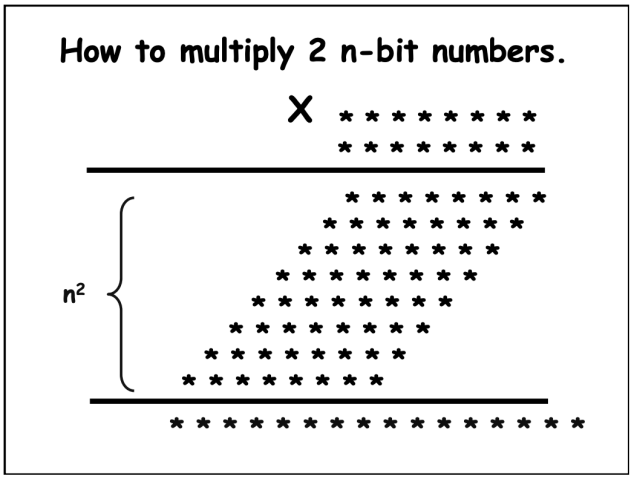
The multiplication algorithms used: "Grade School" algorithm, "Karatsuba" algorithm and "Divide and Conquer" algorithm.

Theoretical base:

Algorithm name	Complexity
Grade School	$O(n^2)$
Karatsuba	$O(n^{\log_2 3})$
Divide and Conquer	$O(n^2)$

Grade School algorithm formula:

xy =
$= (a_0 \cdot 10^0 + a_1 \cdot 10^1 + \cdots + a_n \cdot 10^n) \cdot y =$
$= (a_0 \cdot y + a_1 \cdot y \cdot 10^1 + \cdots + a_n \cdot y \cdot 10^n)$,
where
$y = (b_0 \cdot 10^0 + b_1 \cdot 10^1 + \cdots + b_n \cdot 10^n)$.

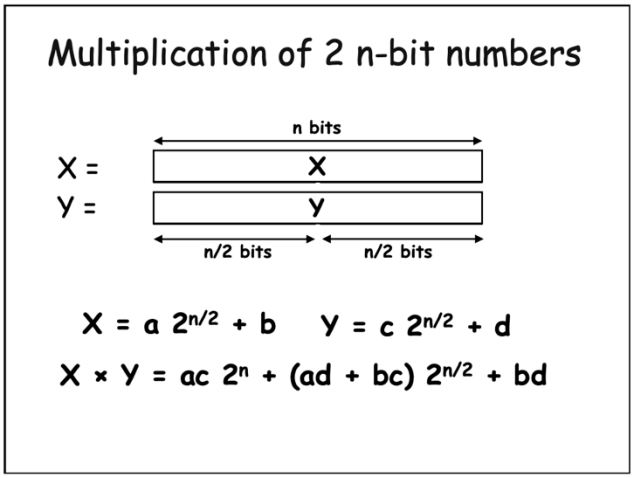


Divide and Conquer algorithm formula:

$xy = (a \cdot 10^{n/2} + b) \cdot (c \cdot 10^{n/2} \cdot d) = ac \cdot 10^n + (ad + bc) \cdot 10^{n/2} + bd$

where
$x = a \cdot 10^{n/2} + b$, where a, b are two halves of x.

$y = c \cdot 10^{n/2} + d$, where c, d are two halves of y.



Karatsuba algorithm formula:

$xy = (a \cdot 10^{n/2} + b) \cdot (c \cdot 10^{n/2} \cdot d) =$

$= ac \cdot 10^{n} + (ad + bc) \cdot 10^{n/2} + bd$

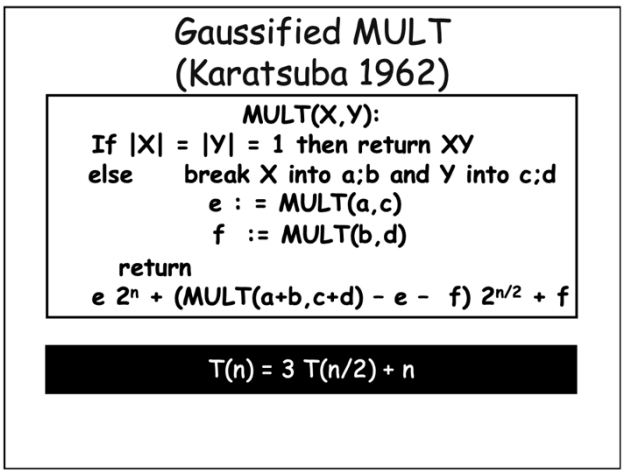
BUT: (ad + bc) is expressed as

$(a + b) \cdot (c + d) - ac - bd$

where

$x = a \cdot 10^{n/2} + b$, where a, b are two halves of.

$y = c \cdot 10^{n/2} + d$, where c, d are two halves of y.



Resources:

1. https://math.stackexchange.com/questions/1118130/grade-school-multiplication-algorithm-for-binary-numbers-explanation
2. http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15251-f10/Site/Materials/Lectures/Lecture22/lecture22.pdf
3. https://en.wikipedia.org/wiki/Karatsuba_algorithm
4. https://en.wikipedia.org/wiki/Multiplication_algorithm
5. https://en.wikipedia.org/wiki/Divide-and-conquer_algorithm

Implementation details:

Class Number was implemented as header file “Number.h”. It is custom data type which stores extremely large number (in little-endian notation to perform different actions easily) in vector as a private member.

```
23  class Number
24  {
25  private:
26      std::vector<int> data;
27  public:
```

Public field has 3 constructors (default, from vector, from size, copy-constructor), overloaded brackets operator to get number’s index, void function print to output data, void function which deletes meaningless zeroes. Also it has overloaded +, += and -= (it works faster) to do corresponding between numbers, overloaded << and <<=. (here it does the same as binary operation, it shifts number on several moves () to the right increasing its value by 10^1, 10^2…10^n where n is a parameter of << and <<=),

```
166      Number operator << (size_t shift) const
174     Number& operator <<= (size_t shift)
```

overloaded multiplication operator of number and digit to perform faster calculations in GSA, split functions which splits number on two halves, returning pair of numbers (is needed in Karatsuba and D&Q) and default destructor.

Another header file is “Multiplicator.h” (which includes “Number.h”). It contains 4 function headers in public field:

- function *“generate_random”* which returns randomly generated number of length given by parameter.

```
7       Number generate_random(size_t k)
```

- Method *“GradeSchool”* which performs multiplication of number and number by loop and overloaded * operator (mentioned earlier.)

```
22      Number GradeSchool (Number first, Number second)
```

- Method *“Divide&Conquer”* which performs multiplication of number and number by splitting number over and over and multiplies by 10 in some power (overloaded <<). It goes by recursion till the base case, where max length of two numbers equals 1 (here it performs multiplication using * operator).

```
35      Number Divide_and_Conquer (const Number& first, const Number& second)
```

- Method *“Karatsuba”* which performs multiplication of number and number by splitting number over and over and multiplies by 10 in some power (overloaded <<). It goes by recursion till the base case, where max length of two numbers equals 1 (here it performs multiplication using * operator). The feature of this algorithm is that it uses already computed products (it saves lime and lowers complexity). Difference from D&Q algorithm and formula is given above.

```
71      Number Karatsuba (const Number& first, const Number& second)
```

The main.cpp file contains Multiplicator header file and has int main, which opens and fills CSV file in a loop (from 1 digit to adjustable limit, I used 40000), taking average duration of algorithms’ performances (out of several test which is also adjustable, I used 3) and closes file at the end of the program.

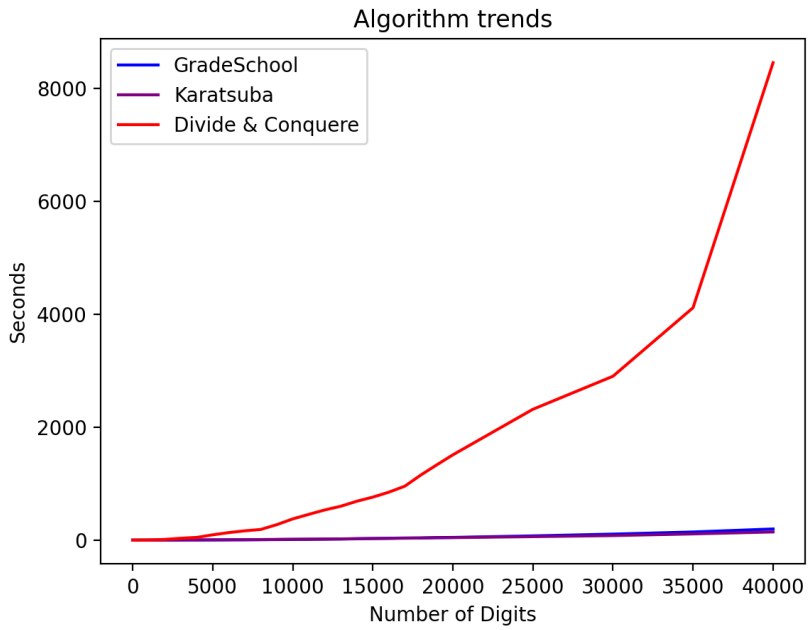
Also python program was implemented to sketch graph based on data taken from CSV file. It uses *“pandas”* and *“matplotlib”* libraries to present accurate graph.

```
import matplotlib.pyplot as plt
import pandas as pd
```

URL: https://github.com/danila231101/dsba-ads2020-hw1

Results

This graph was constructed using data computed by my program.



Step for measurements is 1 from 1 to 10, 10 for 10 to 100, 100 for 100 to 1000, 1000 from 1000 to 20000, 5000 from 20000 to 40000. Number of tests for each measurement to get average result is 3.

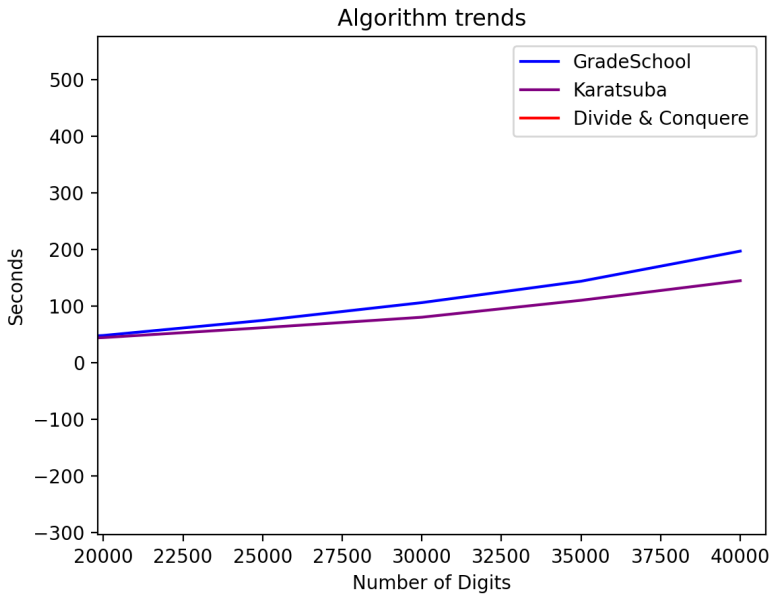
From the given illustration it is seen that before 250-digit number difference between algorithms is seamless. Nevertheless, from 250 digits time of "Divide&Conquer" multiplication starts to increase. Data proves the following:

length	GradeSchool	Karatsuba	Divivde&Conquer
200	0.005602	0.032097	0.151698
300	0.01228	0.060079	0.337859

From 5000-digit number it is clearly seen that D&Q algorithm slows down considerably and this deceleration continues to increase. On this value GSA is still faster than Karatsuba, but situation changes on 20000 value.

length	GradeSchool	Karatsuba	Divivde&Conquer
5000	2.81625	4.89824	93.8601
20000	47.9678	44.207	1510.33

Karatsuba becomes faster than GSA from value 20000 and dominates by minimal time from that point among others. To be sure, here is a closer look on graph.



Obtained data shows the same:

length	GradeSchool	Karatsuba	Divivde&Conquer
40000	197.082	144.795	8455.88

Conclusion:

Removing obscenities and features of realization, stated problems solved. Theoretical part is proved, algorithms' trends approximately fits curves describes their complexities (however D&Q algorithm have complexity $O(n^2)$, it computes much slower because of the feature of the realization) Karatsuba performs faster than GSA from some point as it should be.