The National Research University Higher School of Economics
Moscow
DSBA Department

REPORT

Subject «Algorithm and Data Structure»
Topic «Homework 1: Analysis of Multiplication Algorithms»

Group	193-2
Student	Xu Jingtao
Supervisor(workshop)	S. Shershakov
Submission Date	26. 04. 2020

Moscow

Contents

1. Introduction

Multiplication is an important process in any computation in the field of Computer Science. We were given the following approaches on the lecture: GradeSchool algorithm, Divide And Conquer algorithm and Karatsuba algorithm. These algorithms are explained and compared in this work. Moreover, we are going to create a test bed for these three multiplication algorithms which represents the results through a line table. The test bed is written in C++ and Python.

- C++: Number Class, Multiplication Class and Test Bed;
- Python: importing a csv file and drawing a graph;

2. Problem Formulation

In the first step, we created a Number class for storing an extreme big number.
`Number Class`:

```
class Number
{
protected:
    string _num;    // control the length
    int _len{};       // the length of big numb
    ...
```

This Number class has a std::string container for a big number, and it has all necessary operators, such as $+$, $-$, and comparison operators $>$, $==$, $!=$.

```
    ...
    Number operator+(const Number& rhv) const;
    Number operator-(const Number& rhv) const;

    bool operator>(const Number& rhv)const;
    bool operator==(const Number& rhv) const;
    bool operator!=(const Number& rhv) const;
    ...
```

And of course, having some interface functions is vital for us to get the access to the field of this class.

```
    void print() const { cout << _num; }   // print out the Numberber
    int getLen() { return _len; }
    string getStrNum() const { return _num ; }
```

The following step: Considering the Structure of The Multiplication Class It can be viewed from the graph 1:

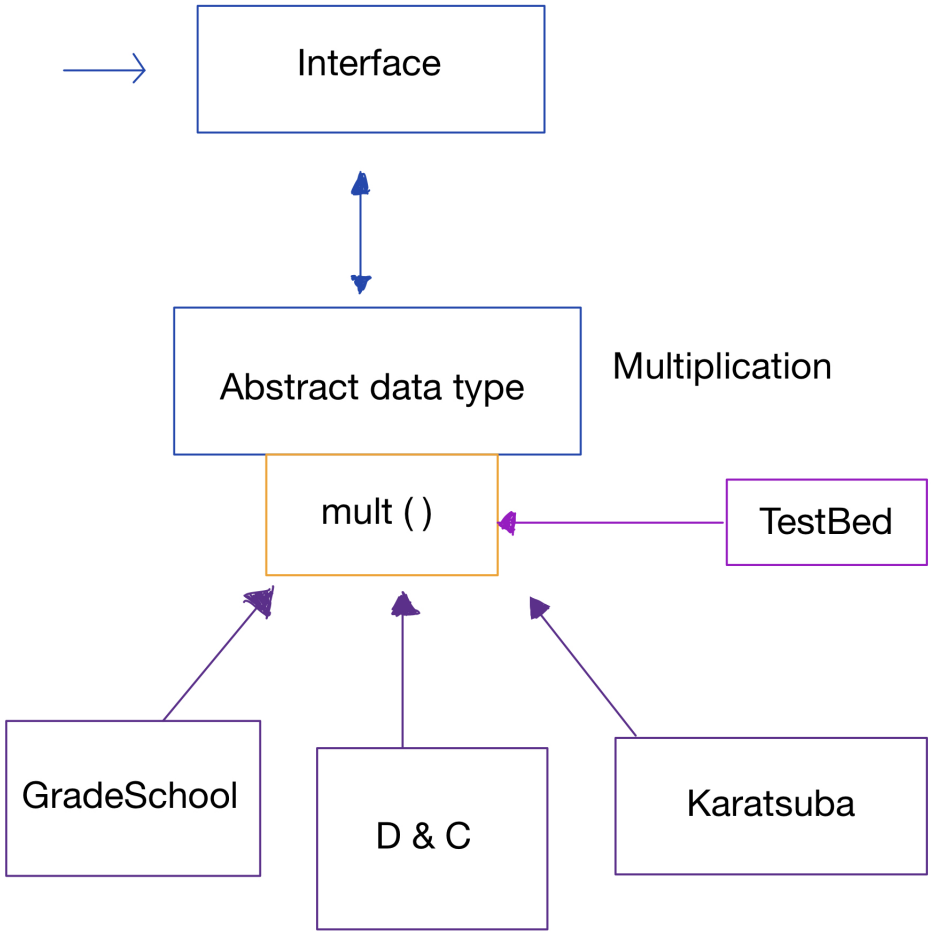


Figure 1. Structure of the class

In the TestBed, we generate two random numbers with `i` digits where `i` is increasing from 1 to a given number `i`.

```
for (size_t i = 1; i <= k; ++i)
{
    string strX = Multiplicator::genRandom(i);
    string strY = Multiplicator::genRandom(i);
    ...
}
```

Then another function called `calcRunTime()` is created for calculating the Running Time of those three multiplication methods.

```
double calcRunTime(Multiplicator* method, const Number& n1, const Number& n2)
{
    ...
```

```
    clock_t start1 = clock();
    Number prod = (*method).mult(n1, n2);
    start1 = clock()-start1;
    t += start1;
    ...
}
```

3. Algorithms

3.1. GradeSchool

GradeSchool is a derived class of The Multiplication Class. This function receives two Numbers and computes their product.

```
/// GradeSchool multiplication
Number GradeSchool::mult(const Number& n1, const Number& n2) const
{
    ...
}
```

Firstly, it gets the string form of the Numbers.

```
string x = n1.getStrNum();
string y = n2.getStrNum();
```

Then, it creates

```
std::deque<int> vec(x.size() + y.size() - 1, 0);
```

for storing the bits of the result number. The reason of creating a dequeue instead of any other containers is that the main feature of std::dequeue is double ends. It is good for us to insert the carry digit to the front of the queue using `.pushfront()`.

```
//insert the carry val. to the front of the queue
while (addflag != 0)
{
    int t = addflag % 10;
    vec.push_front(t);
    addflag /= 10;
}
```

After adding up all products of each two digits from `n1` and `n2`, we will eventually get the final answer.

This example uses long multiplication to multiply 23,958,233 (multiplicand) by 5,830 (multiplier) and arrives at 139,676,498,390 for the result (product).

```
      23958233
x         5830
---------------
      00000000 ( =      23,958,233 x     0)
```

```
     71874699  ( =      23,958,233 x    30)
    191665864  ( =      23,958,233 x   800)
+ 119791165    ( =      23,958,233 x 5,000)
---------------
  139676498390 ( =      139,676,498,390   )
```

It is not hard to see that we have a for-cycle which has time complexity - $\mathcal{O}(n)$. In this algorithm, we have to iterate these two strings n times. That is one $\mathcal{O}(n)$ cycle inside another one with time complexity $\mathcal{O}(n)$. As a result, we obtain $\mathcal{O}(n) \times \mathcal{O}(n) = \mathcal{O}(n^2)$.

The time complexity of GradeSchool algorithm is $\mathcal{O}(n^2)$.

3.2. Divide And Conquer

The mathematician Carl FriedRich Gauss introduced the formula:

$$bc + ad = (a + b)(c + d) - ac - bd \tag{1}$$

from the product of two complex numbers:

$$(a + bi)(c + di) = ac - bd + (dc + ad)i \tag{2}$$

In our case, suppose x and y are two n - digits integers. One breaks each of A and B into two integers of $\lfloor \frac{n}{2} \rfloor$ bits and $n - \lfloor \frac{n}{2} \rfloor$. let $k = \lfloor \frac{n}{2} \rfloor$,

$$xy \times wz = xw \times 10^{2k} + (xz + yw) \times 10^k + yz \tag{3}$$

This algorithm can be considered as a solution to conquer subproblems by solving them recursively. Once the subproblems become small enough that we no longer recurse, we have gotten down to the base case. Ex. When the size of the subproblem go down to 1, we just simply count the product by the library-based multiplication. Its performance(time complexity):

- Base Case: $T(1) = \Theta(1)$
- Divide: $\Theta(1)$
- Conquer: $2T(n/2)$ and $\Theta(n)$
- Combine: $\Theta(1)$

That is:

$$T(n) = 4T(n/2) + \Theta(n) = \Theta(n^2) \tag{4}$$

3.3. Karatsuba

Karatsuba Algorithm is an improved Divide and Conquer algorihm. From the formula $(A1 - A2) \times (B2 - B1) + A1B1 + A2B2 = A1B2 + A2B1$, the following formula is derived. It requires only three multiplication to find the result. The steps:

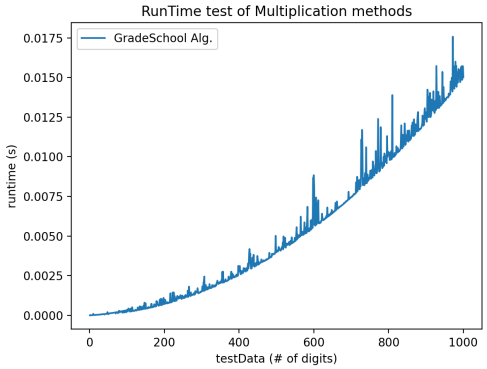
- $r = (x + y) \times (w + z) = xw + (xz + yw) + yz$
- $p = xw$
- $q = yz$
- $xy \times wz = p \times 10^{2k} + (r - p - q) \times 10^k + q$, where $k = \lfloor \frac{n}{2} \rfloor$

We can find out its time complexity by applying Master Theary.

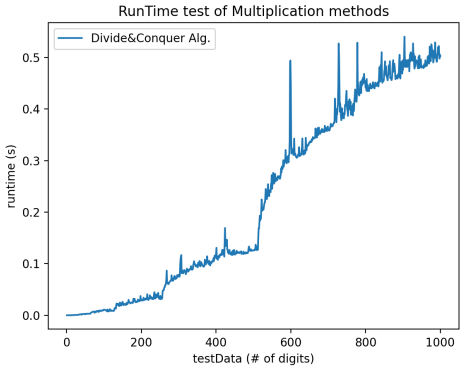
$$T(n) = 3T(n/2) + \Theta(n) = \Theta(n^{\lg 3}) \tag{5}$$

4. Results and Discussion

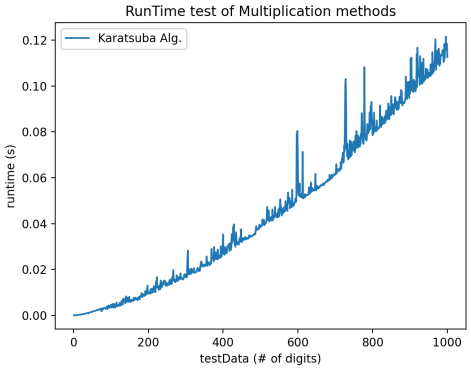
Let us start the comparison of the graphs drawn by the output data sets.



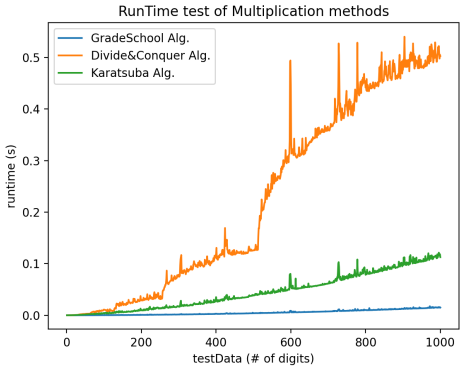
(a) pic1. GradeSchool Algo.



(b) pic2. DivideAndConquer Algo.



(c) pic3. Karatsuba Algo.



(d) pic4. allInOne

Figure 2. result graphs

According to the graph, we can say that Karatsuba Algorithm is faster than other two approaches. However, the Divide and Conquer algorithm and Karatsuba algorithm are slower than the GradeSchool algorithm (pic4). The only thing could trigger is that the amount of work for each subproblem in former is way larger than the latter. For instance, the addition in big number (string type) and some auxiliary method copy the string. The GradeSchool algorithm does not contain any auxiliary method. Therefore, these three graphs should be considered in two different scales. This is a possible way to shorten the gap between them. All in all, the Karatsuba algorithm is theoretically the fastest algorithm among these three.

Conclusion

In this research, we found out that Multiplication process for large numbers is an important problem in the field of Computer Science and the importance of algorithm. The time complexity of the algorithm varies with its implementation. In this case, we certainly failed to show the expected results.(practice is the sole criterion of truth.)

Links of Reference materials

[1] Thomas H. Cormen... Introduction to Algorithms — M.: MIT, 2009.

[2] $https://en.wikipedia.org/wiki/Multiplication_algorithm$