Italian Folk Multiplication Algorithm Is Indeed Better: It Is More Parallelizable

Martine Ceberio[1], Olga Kosheleva[2], and Vladik Kreinovich[1]

Departments of [1]Computer Science and [2]Teacher Education
University of Texas at El Paso
500 W. University, El Paso, TX 79968, USA
mceberio@utep.edu, olgak@utep.edu, vladik@utep.edu

Abstract. Traditionally, many ethnic groups had their own versions of arithmetic algorithms. Nowadays, most of these algorithms are studied mostly as pedagogical curiosities, as an interesting way to make arithmetic more exciting to the kids: by applying to their patriotic feelings – if they are studying the algorithms traditionally used by their ethic group – or simply to their sense of curiosity. Somewhat surprisingly, we show that one of these algorithms – a traditional Italian multiplication algorithm – is actually in some reasonable sense better than the algorithm that we all normally use – namely, it is easier to parallelize.

1 Formulation of the Problem

How we learn to multiply numbers. How students learn multiplication is school?

– First, they memorize the multiplication table – which enables them to multiply 1-digit numbers.
– Then, they learn how to multiply a multi-digit number by a digit.
– Finally, they learn how to multiply two multi-digit numbers.

Let us recall how this is taught in school.

To multiply a multi-digit number by a digit, e.g., multiply 23 by 4,

```
 23
X 4
---
  ?
```

we start with the lowest digit – in this case, with 3, and multiply it by 4. From the multiplication table, we know that the result is 12, so we place 2 in the corresponding digit of a product, and remember 1 as a *carry*, to be added to the next digit:

```
 23
X 4
---
 ?2
```

Then, we multiply the next digit (in this case, 2) by 4, getting 8, and add the carry (in this case, 1) to this product, getting 9:

```
 23
X 4
---
 92
```

Similarly, if we multiply 23 by 6, we:

– get $3 \cdot 6 = 18$, so the carry is 1, and
– then compute $2 \cdot 6 + 1 = 13$,

so the result is:

```
 23
X 6
---
138
```

Once the students master the art of multiplying a multi-digit number by a digit, they learn how to multiply two multi-digit numbers:

– first, we multiply the first number by each digit of the second number, and
– then, we add up all the resulting products.

For example, to multiply 23 by 64, we first perform the above two multiplications, and then add the results:

```
    23
  X 64
  ----
    92
+ 138
------
  1472
```

Ethnic multiplication algorithms. In the past, different ethic groups used different algorithms for multiplication. Probably the most well known is the Russian multiplication algorithm (see, e.g., [1, 3, 5]), in which to compute the product $a \cdot b$, we, in effect:

– translate the second number b into the binary code, i.e., represent it as
$$b = 2^{i_1} + 2^{i_2} + \ldots + 2^{i_k}$$
for some $i_1 > i_2 > \ldots > i_k$, then
– consequently double the first number a, to get the values
$$a, \quad 2^1 \cdot a, \quad 2^2 \cdot a, \quad \ldots, \quad 2^{i_1} \cdot a,$$

– and after this, add the products corresponding to the powers of 2 that form b:

$$a \cdot b = 2^{i_1} \cdot a + 2^{i_2} \cdot a + \ldots + 2^{i_k} \cdot a.$$

If we only have two numbers a and b to multiply, the Russian multiplication algorithm seems to require a lot of unnecessary steps, but it starts making sense if we have to multiply the same number a by different values b. For example – and this is where this algorithm originated – a merchant is selling some material by yards, and he (in the old days, it was usually he) needs to find the price of different amounts of material. In this case, a – the price per yard – remains the same, while the length b changes. The advantage is that in this case, we perform all the doublings only once – as result, we only need:

– to translate into binary code – and for this, it is sufficient to divide by 2, and then
– to add the corresponding products $2^i \cdot a$.

Different ethnic groups had different algorithms. For example, in the traditional Italian folks multiplication algorithm (see, e.g., [4] and references therein), we:

– multiply each digit of the first number by each digit of the second number, and then
– add the results.

For example, in this algorithm, the multiplication of 23 by 64 takes the following form:

```
    23
  X 64
  ----
    12  = 3 x 4
   18   = 3 x 6
    8   = 2 x 4
+ 12
-------
  1472
```

How are ethnic algorithms viewed now. At present, the ethnic algorithms are studied mostly by historians of science and by pedagogues. To pedagogues, such algorithms are an interesting way to make arithmetic more exciting to the kids.

In general, studying different algorithms raises the students' curiosity level. Also, studying algorithms of one's own ethnic group is enhanced by the students' patriotic feelings – although, strangely enough, OK and VK, when studying arithmetic in Russia, never heard of the Russian multiplication algorithm.

What we show in this paper. Our goal is to show that ethnic algorithms actually make sense – many of them are, in some sense, better than the algorithm that we learn at school. We have already mentioned this for the Russian multiplication algorithm.

In this short paper, we show that the Italian folk multiplication algorithm also has its advantages over the traditional modern-school multiplication – namely, the Italian algorithm is easier to parallelize.

2 Italian Algorithm Is Better: An Explanation

Why parallelization. Nowadays, most multiplication is performed by computers, and computers have no problem multiplying large numbers. However, in the past, multiplication was not easy. In the Middle Ages, when even literacy was rather an exception, those who could multiply never needed to do a back-breaking menial work: they could easily find employment as assistants to merchants.

If one needs to multiply two large numbers, and the result is important – a natural idea is to ask for help, to divide the job, so that two specialists in this complex art of multiplication could perform some operations at the same time ("in parallel") and thus, speed up the process.

In a nutshell, this is the same reason why modern computer-based computations use parallelization: if a computation takes too long on a single processor, a reasonable idea is to have several processors working in parallel.

Which algorithm is easier to parallelize. From this viewpoint, it is desirable to check which of the two algorithms – the usual one or the Italian folk one – is easier to parallelize.

How do we gauge easiness? Each of the two algorithms consists of two stages:

- the first, multiplication stage, and
- the second stage, in which add the multiplication results.

Clearly, addition is much easier than multiplication. From this viewpoint, when we talk about parallelization, we should emphasize the need to parallelize the first (multiplication) part of each algorithm.

What can be parallelized in the traditional multiplication algorithm. In the traditional multiplication algorithm, to compute $a \cdot b$, we multiply a by each of the digits of b, and then add the resulting products. Multiplication of a by each of the b's digits does not depend on the multiplication on any other digit, so all these multiplications can be performed in parallel.

In the above example:

- one person can multiply 23 by 4, getting 92, while
- at the same time, another person could multiply 23 by 6, resulting in 138,

after which they can easily add the results.

However, no further parallelization is possible (unless we modify the algorithm). Namely, the way a multi-digit number is multiplied is sequential:

- we do not get the second-from-last digit of the product until we have computed the last digit,
- we do not get the third-from-last digit of the product until we have computed the second-form-last digit, etc.

What can be parallelized in the Italian folk multiplication algorithm. In contrast, in the traditional Italian algorithm, when we first multiply each digit of the first number by each digit of the second number, all these multiplications can be done in parallel.

For example, when we multiply 23 by 64:

- the first person multiplies 3 by 4,
- the second person multiplies 3 by 6,
- the third person multiplies 2 by 4, and
- the fourth person multiplies 2 by 6.

All these four multiplications can be performed at the same time – i.e., in parallel – after which all that remains is an easy task of *adding* all four multiplication results.

Conclusion. We see that the Italian algorithm is indeed better than the traditional one – in the sense that it is easier to parallelize than the traditional multiplication algorithm.

Caution. The above arguments make sense to us, but the readers should be warned that, while these arguments seem reasonable, they do not work if we consider a traditional computer science approach to algorithm complexity – which is based on considering the length of the inputs tending to infinity; see, e.g., [2].

Indeed, as the number of digits B in the second number b increases, it becomes much larger than 10. In this case, in the traditional multiplication algorithm, we no longer need to perform B multiplication of a by a 1-digit number: it is sufficient to find the product of a by each of the 10 digits, and then simply place the corresponding product into the resulting sum. (To be more precise, we need 8 multiplications, since multiplying by 0 and 1 is trivial.)

This observation makes the traditional algorithm somewhat easier – but still, multiplying a very long number by a digit is, in the traditional algorithm, not naturally parallelizable.

Acknowledgments

This work was supported in part by the US National Science Foundation grant HRD-1242122.

References

1. L. N. H. Bunt, P. S. Jones, and J. D. Bedient, *The Historical Roots of Elementary Mathematics*, Dover, New York, 1988.
2. Th. H. Cormen, C. E. Leiserson, R. L., Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts, 2009.
3. D. E. Knuth, *The Art of Computer Programming. Vol. 2. Seminumerical Algorithms*, Addison Wesley, Reading, Massachusetts, 1969.
4. A. Llorente, "3 sencillos métodos para aprender a multiplicar sin calculadora", *BBE Mundo*, posted November 22, 2017, http://www.bbc.com/mundo/noticias-42020116
5. J. I. Vargas and O. Kosheleva, "Russian peasant multiplication algorithm, rsa cryptosystem, and a new explanation of half-orders of magnitude", *Journal of Uncertain Systems*, 2007, Vol. 1, No. 3, pp. 178–184.