Mexican Folk Arithmetic Algorithm Makes Perfect Sense

Julio C. Urenda[1,2], Christian Servin[3], Olga Kosheleva[4], and Vladik Kreinovich[2]

[1,2,4]Departments of [1]Mathematical Sciences, [2]Computer Science, and [4]Teacher Education, University of Texas at El Paso
500 W. University, El Paso, Texas 79968, USA
jcurenda@utep.edu, olgak@utep.edu, vladik@utep.edu
[2]Computer Science and Information Technology Systems Department
El Paso Community College (EPCC), 919 Hunter Dr.
El Paso, TX 79915-1908, USA, cservin1@epcc.edu

1. Standard Arithmetic Algorithms That Everyone Learns

- At school, kids all over the world study the exact same addition, subtraction, and other arithmetic algorithms.
- For example, for addition: as everywhere in the world:
  - We start with the lowest digits, add them.
  - If the result is larger than 10, we carry a one to the next digit.
  - Then we add it to the result of adding the next digits of the two numbers, etc.

2. Standard Arithmetic Algorithms (cont-d)

- If we want to add 89 and 23, we first add the last digits 9 and 3, resulting in digit 2 and a carry:

```
    .
   89
 + 23
 ----
    2
```

- Then, we add 8, 2, and the carried 1, resulting in 11

```
   89
 + 23
 ----
  112
```

3. How Mexico is Different

- Interestingly, in Mexico, many parents teach their kids somewhat different "folk" arithmetic algorithms.
- One of us (JCU) who grew up in Mexico indeed learned different algorithms from his parents.
- There are several such algorithms.
- One interesting Mexican folk arithmetic algorithm was recently described in a pedagogical paper.
- In this algorithm, we start not with the lowest but with the highest digits.
- In the above example, we start by adding 8 and 2:

```
   89
 + 23
 ----
  10
```

4. How Mexico is Different (cont-d)

- Then, we add the next digits.
- Since the result (12) is larger than 10, we add 1 to the previous digit.
- It all results, of course, in the same answer 112 as before:

```
  89
+ 23
----
 112
```

- A similar idea can be (and is) used for subtraction.

5. So What Is a Problem?

- The main point of the cited paper is that:
  - if a student uses a different algorithm,
  - it does not necessarily mean that the student does not know to add or subtract.
- The teacher needs:
  - to ask the student how he or she (in this case, it was a she) came up with this answer, and
  - to make sure that the corresponding algorithm is correct.
- From the pedagogical viewpoint, this probably solves the problem.
- The teachers are aware that students can use different algorithms.

6. So What Is a Problem (cont-d)

- Students are not frustrated – as they would be if:
  - they got a not-so-perfect grade
  - for producing a correct answer by means of a correct (although not standard) algorithm.
- Implicit in this paper is the assumption that:
  - the usual algorithm is optimal, and
  - modified versions can be tolerated but definitely not recommended.
- But our viewpoint is different.
- Most of the authors of this paper are computer scientists.
- As computer scientists, we know that there are many different algorithms for solving the same problem.

7. So What Is a Problem (cont-d)

- All computer science students learn:
  - that there are many different algorithms for sorting and searching,
  - that there are many algorithms for subtraction and multiplication, etc.
- How do we select an algorithm?
- Usually, we select the most efficient one, i.e., the one that runs the fastest.
- Otherwise, we would not be able to process as much information as computer currently do.

8. So What Is a Problem (cont-d)

- As computer scientists, we also know that:
  - the arithmetic algorithms that we all learn in elementary school
  - are often *not* the most efficient ones, and that different,
  - more efficient algorithms are implemented in computers.
- For example:
  - to make subtraction more efficient,
  - computers use 2's complement representation of negative numbers,
  - in which the same method applies both to positive and negative numbers,
  - in contrast to the algorithms that we learn in elementary school.

9. So What Is a Problem (cont-d)

- Similarly, to multiply two large numbers, a more efficient way is to use Fast Fourier Transform.
- From this viewpoint:
  - the very fact that Mexican folk algorithms have survived for so long,
  - in contrast to other historic ethnic algorithms like the Russian peasant multiplication algorithm,
  - seems to us an indication that these algorithms may be more efficient than the usual ones.
- And this is exactly what we show in this talk.
- We show that these algorithms *are*, in some reasonable sense, more efficient.

10. Let Us Look at How Computers Add Numbers

- As we have mentioned, when people design computers, they try to implement the most efficient algorithms.
- So:
  - to understand which algorithms are most efficient,
  - a natural idea is to look at how computers perform the corresponding operations.
  - i.e., in our case, how computers add and subtract.
- In this regards:
  - usual introductory Computer Science textbooks
  - provide a simplified version of how computers perform arithmetic operations.

11. How Computers Add Numbers (cont-d)

- According to these textbooks:
  - the main difference between how we add and how computers add
  - is that computers use binary numbers.
- Let us illustrate it on a simplified 5-bit example.
- If a computer needs to add $13 = 01101_2$ and $5 = 00101_2$, it first adds the last digits, getting a carry

```
      .
  01101
+ 00101
-------
      0
```

12. How Computers Add Numbers (cont-d)

- Then it adds the carry and the next digits:

```
  01101
+ 00101
-------
     10
```

- Then it adds the next digits:

```
    .
  01101
+ 00101
-------
    010
```

13. How Computers Add Numbers (cont-d)

- Then it adds 1, 0, and the carry:

```
    .
   01101
 + 00101
 -------
    0010
```

- Finally, it adds 0, 0, and the carry:

```
   01101
 + 00101
 -------
   10010
```

- So we get the correct result $18 = 10010_2$.

14. We Need to Go Beyond the Simplified Description

- The above operation took quite a few steps already for 5-bit numbers.
- In reality, modern computers deal with 64-bit numbers.
- For such numbers:
  - if each bit has to wait – as the simplified algorithm implies – for all previous bits,
  - it will take us 64 sequential bit operations to add two numbers.
- This was how very first computers were built, but this is definitely *not* how addition is performed now.
- In the actual computers, all bits are added at the same time, and all the carries are obtained at the same time.

15. Beyond the Simplified Description (cont-d)

- Then, on the next iteration, carries are added to the result, etc.
- In the above example, the first parallel step would result in the following:

```
    . .
   01101
 + 00101
 -------
   01000
```

- Then, we add the result 01000 and the binary number 01010 corresponding to carries:
  - 0 means no carry,
  - 1 means that there is a carry.

16. Beyond the Simplified Description (cont-d)

- By applying the same parallel procedure to the numbers 01000 and 01010, we get

```
 .
 01000
+ 01010
-------
 00010
```

- Finally, we add the result 00010 and the binary number 10000 corresponding to carries:

```
 00010
+ 10000
-------
 10010
```

- So, we get the desired result $18 = 10010_2$.

17. This Is Really More Efficient

- Here, instead of 5 iterations, we use only 3.
- The savings are even larger for 64-bit numbers.
- How is this related to the Mexican folk algorithm?
- Computers are simple machines.
- Crudely speaking, they see and process one symbol at a time.
- For example, to understand a simple 1000×1000 picture, they need to process all million bytes one by one.
- In contrast, human perception is highly parallel.
- We look at the picture and we immediately see, e.g., that this is a cat (if it is indeed a picture of a cat).
- Similarly, for addition, we can add two digits at the same time.

18. This Is Really More Efficient (cont-d)

- From this viewpoint, whether we start with the lowest digits or with the highest digits, does not matter much.
- For example, if we want to add 89 and 23, we can first add all the digits, getting

```
 ..
  89
+ 23
----
  02
```

- Then, we add – all the digits at the same time – the result 02 and the number 110 corr. to carries:

```
  02
+110
----
 112
```

19. This Is Really More Efficient (cont-d)

- This does not (yet) show that the Mexican folk algorithm is better.
- However, it does show:
  - that the perceived importance of starting with the lowest digits is actually not that important,
  - if we use an efficient algorithm instead of the one we learned in elementary school.

20. What About Subtraction

- A similar idea can be used for subtraction.
- Suppose that we want to subtract 89 from 112.
- We subtract each pair of digits, making all the needed borrowings:

```
  ..
 112
- 89
----
 133
```

21. What About Subtraction (cont-d)

- Then from the result 133, we subtract – digit-by-digit – the number 110 corresponding to the borrowings:

```
 133
-110
----
 023
```

- Thus, we get the correct value 23.

22. Let Us Look at Practical Situations Where People Use Addition and Subtraction

- So far, we considered the problem as a pure theoretical one:
  - we need to add or subtract numbers, and
  - the question is how to best do it.
- But *why* do we need to add or subtract numbers?
- By the way, this is not simply a rhetorical question.
- There are (unfortunately) many public figures who claim that:
  - arithmetic is not needed in real life and
  - thus, should not be taught in schools.

23. Practical Situations (cont-d)

- As many of them say it:
  - I was tortured in school by having to learn very complex and very useless stuff like fractions.
  - I barely got a passing grade on that, this traumatized me for life.
  - So let us stop torturing kids and ruining their lives.
- Let us give a real-life example of where subtraction is needed:
  - you are buying something by paying cash
  - and yes, there are still many small places where you need to pay cash,
  - you do not have the exact change,
  - so you are expecting the change.

24. Practical Situations (cont-d)

- In general:
  - if you paid the amount a and you need to pay the amount $b < a$,
  - then you need to get the difference $a - b$ back as change.

25. A Reasonable Idea

- How would you prefer your change to be given to you?
- Take into account that in most countries, there are no 15-dollar or 15-peso bills.
- Bills are either in tens, or in hundreds, etc.
- In the past, in Russia, sellers were *required* to first give you the largest nominations, then the smaller ones, etc.
- For example, suppose that:
  - you pay with a 100-rouble bill and
  - the thing you bought costs 21.56 roubles
  - to be more precise, 21 roubles and 56 kopecks.
- You should get 78.54 rubles change.

26. A Reasonable Idea (cont-d)

- You are supposed to get:
  - first 70 roubles (e.g., $50 + 10 + 10$),
  - then 8 roubles (e.g., $5 + 1 + 1 + 1$),
  - then 50 kopecks (could be one coin) and,
  - finally, 4 kopecks.
- Why is this better than starting with kopecks?
- Because if you get distracted or simply forget about the last step, you still get the most of the change.
- On the other hand:
  - if this is done in the opposite order, and you are supposed to get 70 roubles last,
  - you miss most of the change if distracted.
- This happened sometimes, which is why this requirement was imposed.

27. How Is This Related to the Mexican Folk Algorithm?

- In most US stores, a machine computes the change automatically.
- In this case, it does not matter that much how you compute the difference.
- But in some places – and in many places in the old days – the difference $a - b$ had to be computed orally.
- From this viewpoint, it is indeed desirable to start with the largest digit, not with the smallest digit.
- This is exactly as the Mexican folk algorithm suggests.

28. Another Practical Example

- A specific example of such a situation is a restaurant.
- There usually, at some point:
  - after getting the largest nominations as a change,
  - many customers suggest that the waiter keeps the remaining part of the change as a tip.
- In this case, there is even more need for a Mexican-style algorithm.
- Indeed, in such situations, no one cares that much about the exact computation of a different.
- For example, I can leave 4 roubles and whatever kopecks remain as a tip.
- So there is no need to compute the exact amounts all the way to the lowest digits.
- This brings us to another aspect of practical examples.

29. In Many Practical Situations, We Operate Under Uncertainty

- In real life, situations when we need to compute the exact difference are somewhat rare.
- Much more frequently, we compute *approximate* differences.
- This is why this paper was submitted to a fuzzy conference.
- For example, when we go shopping with cash, we rarely know exactly how much money exactly we have.
- For example, we may know that we have approximately 200 roubles.

30. We Operate Under Uncertainty (cont-d)

- Then, when we see something that costs, say, 149 roubles, we need to check whether
  - after buying this attractive (but someone expensive) object
  - we will still have enough money to buy what we planned to buy from the very beginning.
- In this case, we subtract 149 from "approximately 200".
- Of course, since we do not know the exact value of the original amount, it makes no sense to subtract exactly.
- "Approximately 50" is a much more adequate answer than "approximately 51".

31. We Operate Under Uncertainty (cont-d)

- In this case:
  - not only we do not need to compute the lower digits, these digits will be misleading,
  - letting us mistakenly think that the amount is between 50.5 and 51.5.
- In such situations, it is also more efficient:
  - to start with the higher digits, and
  - not to go all the way to the lowest ones.
- We may also be subtracting two approximate numbers.
- For example:
  - we have approximately 50 roubles left after downtown shopping, and
  - we are thinking whether we have enough money left to have a nice lunch at a fancy downtown cafeteria.

32. We Operate Under Uncertainty (cont-d)

- If, in our experience, lunch there costs approximately 40 roubles, we can risk it.
- However, if our previous costs was about 50, we better not risk it.
- The difference may turn to out to be negative.

33. Conclusion

- The Mexican folk arithmetic algorithm starts adding and subtraction:
  - from the highest digits,
  - not from the lowest ones, as the usual algorithms.
- This may sound weird.
- However, in many realistic situations, it is actually better than the traditional ones.
- This is especially true in situations when we perform operations on fuzzily known quantities.

34. Acknowledgments

This work was supported in part by the National Science Foundation grants:

- 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science);
- HRD-1834620 and HRD-2034030 (CAHSI Includes).

It was also supported by the program of the development of the Scientific-Educational Mathematical Center of Volga Federal District No. 075-02-2020-1478.

35. Bibliography

- L. N. H. Bunt, P. S. Jones, and J. D. Bedient, *The Historical Roots of Elementary Mathematics*, Dover, New York, 1988.
- Th. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts, 2009.
- J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, Cambridge, Massachusetts, 2017.
- D. E. Knuth, *The Art of Computer Programming. Vol. 2. Seminumerical Algorithms*, Addison Wesley, Reading, Massachusetts, 1969.
- O. Kosheleva and K. Villaverde, *How Interval and Fuzzy Techniques Can Improve Teaching*, Springer Verlag, 2018.

36. Bibliography (cont-d)

- G. Krause, L. A. Maldonado Rodríguez, and M. Adams-Corral, "Listening to and understanding students' algorithms", *Mathematics Teacher: Learning and Teaching PK-12*, 2021, Vol. 114, No. 2, pp. 139–141.

- 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.