A new algorithm for column addition

TSG-10 ICME-11
Friday, July 11, 2008

Patricia Baggett
New Mexico State University
Las Cruces, New Mexico
USA
baggett@nmsu.edu

Andrzej Ehrenfeucht
University of Colorado
Boulder, Colorado
USA
andrzej.ehrenfeucht@colorado.edu

http://isis.nmsu.edu/~breakingaway/

1. Introduction

Children in elementary school are supposed to master the "standard" algorithm for adding a list of whole numbers (or even decimals) by about second or third grade.

But some pupils never completely master the algorithm, and others soon forget the procedure.

"Standard algorithm"– Right-to-left column algorithm in which each column is processed in top-down fashion, and the "carry" is written at the top of the next column to the left.

```
 13   ← carries
  26
  39
  58
  47
 ---
 170
```

The most complex addition exercises given in typical school textbooks in the United States in different years:

year	author	
1810	J. Joyce	sixteen 5-digit numbers
1901	E. E. White	nine 6-digit numbers
1934	L. J. Brueckner et al.	nine 3-digit numbers
1990	S. Hake & J. Saxon (4th edition)	seven 3-digit numbers

Why do many children not master written addition? The fault seems to lie in the algorithm itself.

The standard algorithm was designed six hundred years ago for merchants, bankers, and other professionals.
It does not fulfill modern educational or practical needs.
(See Van de Walle, 2005.)

2. The new algorithm.

We will show examples using whole numbers.

We will compare the new algorithm to the standard one with respect to:

1. The content of long-term memory (the set of facts that need to be recalled)

2. Modularity (dividing the algorithm into subparts that can be executed separately)

4. Properties of individual modules (no. of operations, their complexity, working memory load, no. of times working memory is updated)

4. Flexibility (choices a person makes during computation)

The new algorithm is better according to each of the criteria.

One-digit numbers:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Their complements to ten:	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**

When we add a column of numbers, instead of adding a digit, we may subtract its complement and mark this digit with a dot to Indicate that we need to carry one to the next column.
We may also add the digits in one column in any order. But in order to avoid errors, we cross out any digit that has been used.

Later we count the number of dots, and write it at the top of the next column to the left as a carry.

```
Example      2  2     ← carries
                1  2
                2  7•
                4  8•
                6• 1
             +  7• 3
             --------
             2  2  1
```

3. Historical sources

Writing dots next to a digit when the sum exceeds ten was used in the past.

We found examples in two arithmetic books, one from 1798 and another from 1846.

John Gough,
Practical Arithmetick
1798

PRACTICAL

ARITHMETICK

IN FOUR BOOKS.

I. Whole Numbers, Weights and Measures.
II. Fractions, Vulgar and Decimal.
III. Mercantile Arithmetick.
IV. Extraction of Roots, Progressions, &c.

EXTRACTED FROM

The Large and Entire Treatise,

And Adapted to the Commerce of Ireland, as well as that of Great-Britain.

For the use of SCHOOLS.

By JOHN GOUGH,
Author of the Practical English Grammar.

CAREFULLY REVISED BY
THOMAS TELFAIR, Philomath.

And now fitted to the Commerce of AMERICA.

With an APPENDIX of ALGEBRA,
By the Late W. ATDINSON, of Belfast.

DUBLIN: Printed:—

WILMINGTON: Re-printed, and Sold, by
PETER BRYNBERG.

M,DCC,XCVIII.
1798

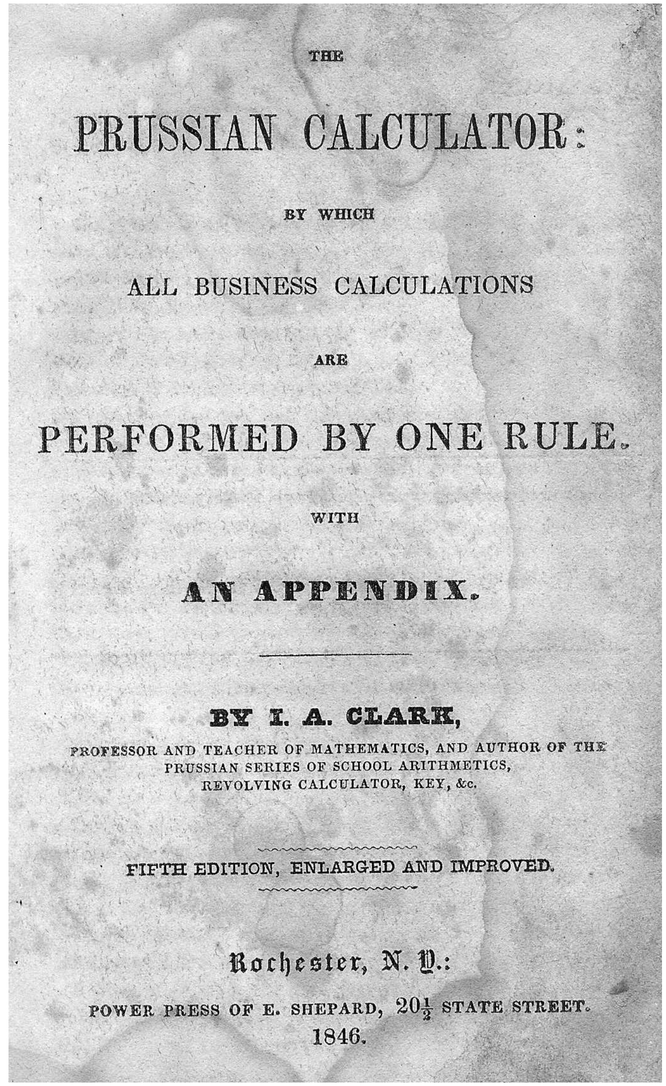
Rule II. *Example* 2.

When the numbers to be added are many, the following method may be practised. Begin with the lowest figure of units place (as before) and joining it to the figures above it (as *per* last rule) for every ten arising in the addition make a point over against the figure which added to the former maketh ten or more than ten, add the overplus above ten to the next figure above it, and so proceed to the top; then count the points and how many they are, so many carry, and add to the figure of the next place, and so proceed in like manner through all the places, and the points of the last places collect, and set their number to the left hand of the figure under the last place.

7.4.3.6
2 1 7 9.
5.0 8.7.
6 8 5.3
2.4.6 4.
7 2 8.7
31 3 0 6

This rule doth not differ essentially from the last, being only a contrivance to help the memory.

I. A. Clark,
The Prussian Calculator
(a business arithmetic)
1846



THE

PRUSSIAN CALCULATOR:

BY WHICH

ALL BUSINESS CALCULATIONS

ARE

PERFORMED BY ONE RULE.

WITH

AN APPENDIX.

BY I. A. CLARK,

PROFESSOR AND TEACHER OF MATHEMATICS, AND AUTHOR OF THE PRUSSIAN SERIES OF SCHOOL ARITHMETICS, REVOLVING CALCULATOR, KEY, &c.

FIFTH EDITION, ENLARGED AND IMPROVED.

Rochester, N. Y.:

POWER PRESS OF E. SHEPARD, $20\frac{1}{2}$ STATE STREET.

1846.

I have also observed some of the best accountants add by tens; or as we might with more propriety say, they add till the amount of their figures equals 10, then they drop 10 and continue to add till the amount again equals 10; they then drop and proceed as before, making a point with the pen or pencil, at each figure where the sum equals 10:—thus, in the example, 7 and 3=10·, 2 and 8=10·, 6 and 4=10·,

6·4·
7 6
9·8·
5·2
3 3·
2 7
———
3 5 0 Ans.

place the cipher below, count the tens, and carry them to the next column; then add as before, saying: 3 and 2+5+3= 13; drop 10 and carry 3; 3+9=12; drop 10 and carry 2; 2+7+6=15; make a point at 5, at 9, and 6; place the 5 under the column; then count the tens which are 3, and place the 3 at the left of the 5, &c.

In these historical examples, dots were written next to a digit when the sum exceeded ten.

Addition was carried out bottom-up and not top-down.

And the number carried (the number of dots in a column) was not written down, but remembered.

In our algorithm, one can add the digits in one column in any order. But to avoid errors, the user marks any digits that are used.

Let's look again at our example:

```
 2  2      ← carries
    1  2
    2  7•
    4  8•
    6• 1
  + 7• 3
  --------
  2  2  1
```

3.1. Modules, strategies, and macros

Modules of an algorithm are the parts of it that can be processed independently. Here, processing any list of numbers in one column that add to zero is a module.

In our example we had 4 modules:

2 and 8 in the right column
3 and 7 in the right column
4 and 6 in the left column, and
2, 1, and 7 in the left column

$$
\begin{array}{cccl}
 & 2 & \underline{2} & \quad \leftarrow \text{carries} \\
 & & \underline{1} & \underline{2} \\
 & & 2 & \underline{7}\bullet \\
 & & \underline{4} & \underline{8}\bullet \\
 & & \underline{6}\bullet & 1 \\
 + & & \underline{7}\bullet & \underline{3} \\
\hline
 & 2 & 2 & 1
\end{array}
$$

You may take a break after executing a module and resume your computation any time later, so even very long computations do not require a long attention span.

Frequency of occurrence of modules

Length of the sequence of digits 1 through 9 (no zeroes) to be added:	Percentage (rounded to .01%) of sequences of this length which contain modules:
2	11.11%
3	39.64%
4	79.52%
5	98.33%
6	99.94%
7	100.00%
8	100.00%
9	100.00%

Every sequence of length 10 contains a module.
(Computed by Michael Main, University of Colorado, Boulder.)

Another example

Typical long column written as a line:		**7**	**3**	**9**	**6**	**2**	**7**	**7**	**9**	**1**	**2**	**2**	**5**	**7**	**6**	**8**
Modules	**(1)**	**7**	**3**													
	(2)								**9**	**1**						
	(3)											**2**				**8**
	(4)				**6**	**2**					**2**					
	(5)			**9**			**7**	**7**						**7**		
Not in a module:													**5**		**6**	

3.2. Strategies

A strategy in a flexible algorithm consists of additional rules that put restrictions on available options. Some strategies lead to good performance, and some may lead to poor performance.

Using only the top-down (or bottom-up) order of addition of digits in a column is an example of an inefficient strategy. (See e.g. Gough, 1798; Clark, 1846, cited above.)

But different users may have different preferences, so the concept of a “good strategy” is subjective.

In this algorithm the number of different strategies (good and bad) is practically unlimited.

3.3. Macros

Macro-operations (macros) are groups of operations that a person executes as one unit.

Consider this example:

```
  1
  1
  1
  1
 +6
```

One person may compute it as 1, 2, 3, 4, minus 4, 0. This person uses 3 additions and 1 subtraction (cancellation).

Another person may see at a glance that this group of digits forms a module, corss out the ones, and mark 6.

The main difference between experts and novices executing a flexible algorithm is in their use of macros.

Remark.
Using macros in order to speed up addition is not new. See for example their treatment in a business arithmetic book, Sutton & Lennes, published in 1938.

4. Comparison of the new algorithm to the standard algorithm

4.1 Content of long-term memory

In order to use either algorithm (standard or new), children need to master (i.e., recall effortlessly) some addition “facts”.

The standard algorithm requires mastery of 81 facts (only addition and not subtraction) belonging to 45 families. For example, 6 + 7 = 13 and 7 + 6 = 13 are two facts in the same family.

Remark.
A family of addition and subtraction facts is described by one equality, a + b = c, which contains the following facts (also described by equalities):

$$a + b = c,\ \ b + a = c,\ \ c - a = b,\ \ c - b = a.$$

The new algorithm also requires 81 facts (45 addition and 36 subtraction), but they belong to only 25 families, because they are restricted to pairs of numbers whose sum is ≤ 10.

For example, 2 + 6 = 8, 6 + 2 = 8, 8 – 2 = 6, and 8 – 6 = 2, belong to the same family.

So the new algorithm requires 25 equalities, and not 45.

We don't say that children should not learn more facts!
We onlysay they don't need more facts in order to learn this algorithm.

4.2 Modularity

The standard algorithm requires that the computation of a whole column be done in one pass without interruptions, because partial sums have to be remembered. So its difficulty increases when more numbers are added.

The new algorithm requires only that *individual modules* be computed without interruption.
The most common modules contain only two digits, and modules containing more than five digits are very rare, so the difficulty doesn’t increase when more numbers are added.
The only increase in difficulty is due to counting the marks.
When we add 5 numbers, we need to count only up to 4 marks per column.
When we add 20 numbers, we may need to count almost 20 marks.

4.3. Properties of modules

We say that working memory is updated when we add to or subtract from the number that is held in memory and we have to remember the result.

In the standard algorithm, adding *n* non-zero digits in a column requires *n – 2* memory updates.
Processing a module containing *m* numbers requires *m – 3* updates.

An example

Standard algorithm		New algorithm			
Think:			Module number:		
5	5	5•			3
7	12	7•	1		
4	16	4			3
3	19	3	1		
4	23	4		2	
1	24	1			3
6	30	6•		2	
+ 8	38	+ 8			
38		38			

	Standard algorithm	New algorithm
Additions and subtractions	7	1
Cancellations	0	3
Memory updates	6	0

4.4. Flexibility

The standard algorithm is rigid—it prescribes every step of the computation.
Therefore after it is mastered it can be performed automatically without any thought.
This was important in the past for accountants and other human computers who spent hours doing sums.
But today computers and not humans should do all mindless calculators.

The new algorithm requires planning and reflection in finding and choosing which modules to process.
The same task can be done in many different ways, some better than others.
Because modules are very small units, and computation can be interrupted after each module, discussions can be carried out even during computation.

5. Experimental data

We taught this algorithm to 19 students (3 practicing teachers and 16 future teachers) taking a class in elementary mathematics at New Mexico State University in fall 2007.

They practiced it for approximately 15 minutes per week for 14 weeks.

In an anonymous questionnaire given at the end of the course, 18 students said that they liked the new algorithm and one said that she didn't.
The main reasons for liking it are illustrated by these comments:

"It is easy and it is challenging…"
"So much easier and interesting."
"Makes adding fun and fast!"
"I think it allows you to add more easily."

We also observed that 15 of the 19 students who learned this algorithm started using it inother tasks that required adding several whole numbers or decimals.

We plan to continue to collect experimental data from both adults and children.