Can We Learn Algorithms from People Who Compute Fast: An Indirect Analysis in the Presence of Fuzzy Descriptions

Olga Kosheleva[1]

1.Department of Teacher Education
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
olgak@utep.edu

Abstract— *In the past, mathematicians actively used the ability of some people to perform calculations unusually fast. With the advent of computers, there is no longer need for human calculators – even fast ones. However, recently, it was discovered that there exist, e.g., multiplication algorithms which are much faster than standard multiplication. Because of this discovery, it is possible than even faster algorithm will be discovered. It is therefore natural to ask: did fast human calculators of the past use faster algorithms – in which case we can learn from their experience – or they simply performed all operations within a standard algorithm much faster? This question is difficult to answer directly, because the fast human calculators' self-description of their algorithm is very fuzzy. In this paper, we use an indirect analysis to argue that fast human calculators most probably used the standard algorithm.*

Keywords— Fast Fourier transform, fast human calculators, fast multiplication, fuzzy description

1 People Who Computed Fast: A Historical Phenomenon

In history, several people have been known for their extraordinary ability to compute fast. Before the 20 century invention of computers, their computational abilities were actively used.

For example, in the 19 century, Johann Martin Zaharias Dase performed computations so much faster than everyone else that professional mathematicians hired him to help with their calculations; see, e.g., [1, 3]. Dase:

- computed π with a record-breaking accuracy of 200 digits,

- calculated the logarithms table with 7 digit accuracy, and

- performed many other computational tasks.

Karl Friedrich Gauss himself recommended that Dase be paid by the Academy of Science to perform the calculations.

2 People Who Computed Fast: How They Computed?

Calculations have been (and are) important in many practical problems. Because of this practical importance, people have therefore always been trying to speed up computations. One natural way to speed up computations is to learn from the people who can do computations fast.

3 People Who Computed Fast: Their Self-Explanations Were Fuzzy and Unclear

In spite of numerous attempts to interview the fast calculators and to learn from them how they compute, researchers could not extract a coherent algorithm. One of the main reasons is that most of the fast calculators were *idiots savants*: their intellectual abilities outside computations were below average. Their explanations of their algorithms were always fuzzy and imprecise, formulated in terms of words of natural language rather than in precise mathematical terms.

To us familiar with fuzzy logic and its applications this fuzziness should not be surprising. It is normal that people in general – and not necessarily idiots savants – cannot precisely describe

- how they drive,

- how they operate machines,

- how they walk,

- how they translate from one language to another,

- how they recognize faces,

- how they control different situations, etc.

– this is why fuzzy control and other fuzzy techniques, techniques for translating this fuzzy description into a precise strategy, have so many practical applications; see, e.g., [4].

What may be somewhat unusual about fast computations is that while the self-description of fast calculators was fuzzy and imprecise, the results of the computations were always correct and precise.

4 With the Appearance of Computers, the Interest in Fast Human Calculators Waned

With the appearance of computers, the need for human calculators disappeared, and the interest in their skills waned. Such such folks may provide a good entertainment, they may be of interest to psychologists who study how we reason and how we perform menial tasks – but mathematicians were no longer interested.

Yes, fast human calculators can perform calculations faster than an average human being – but electronic computers can perform the same computations much much faster. From this viewpoint, fast human calculators remained a curiosity.

5 Why Interest Waned: Implicit Assumptions

One of the main reasons why in the 1940s and 1950s the interest in fast human calculators waned is that it was implicitly assumed that the standard techniques of addition and multiplication are the best.

For example, it was assumed that the fastest way to add the two n-digit numbers is to add them digit by digit, which requires $O(n)$ operations with digits.

It was also implicitly assumed that the fastest way to multiply two n-digit numbers is the standard way to multiply the first number by each of the digits of the second numbers, and then to add the results. Each multiplication by a digit and each addition requires $O(n)$ steps, thus multiplication by all n digits and the addition of all n results require $n \cdot O(n) = O(n^2)$ computational steps.

From this viewpoint, the only difference between a normal human calculator, a fast human calculator, and an electronic computer is in the speed with which we can perform operations with digits. From this viewpoint, the only thing we can learn from fast human calculators is how to perform operations with digits faster. Once the electronic computers became faster than fast human calculators, the need to learn from the fast human calculators disappeared.

6 A Surprising 1960s Discovery of Fast Multiplication Algorithms

The above implicit assumption about arithmetic operations was not questioned until a surprising sequence of discoveries was made in the 1960s; see, e.g., [2].

These discoveries started with the discovery of the Fast Fourier Transform algorithm, an algorithm that enables us to compute the Fourier Transform

$$\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \cdot \int f(t) \cdot \exp(i \cdot \omega \cdot t) \, dt, \qquad (1)$$

where $i = \sqrt{-1}$, in time $O(n \cdot \log(n))$ – instead of the $n \cdot O(n) = O(n^2)$ time needed for a straightforward computation of each of n values of $\hat{f}(\omega)$ as an integral (i.e., in effect, a sum) over n different values $f(t)$.

The ability to compute Fourier transform fast lead to the ability to speed the computation of the convolution of two functions:

$$h(t) \stackrel{\text{def}}{=} \int f(s) \cdot g(t-s) \, ds. \qquad (2)$$

A straightforward computation of the convolution requires that for each of the n values $h(t)$, we compute the integral (sum) of n different products $f(s) \cdot g(t-s)$ corresponding to n different values s. Thus, the straightforward computation requires $O(n^2)$ computational steps.

Indeed, it is known that the Fourier transform of the convolution is equal to the product of Fourier transforms. Thus, to compute convolution, we can do the following:

- first, we compute Fourier transforms $\hat{f}(\omega)$ and $\hat{g}(\omega)$;

- then, we compute the Fourier transform of h as

$$\hat{h}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega); \qquad (3)$$

- finally, we apply the inverse Fourier transform to the function $\hat{h}(\omega)$ and compute the desired convolution $h(t)$.

What is the computation time of this algorithm?

- Both Fourier transform and inverse Fourier transform can be computed in time $O(n \cdot \log(n))$.

- The point-by-point multiplication $\hat{h}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$ requires n computational steps.

Thus, the overall computation time requires

$$O(n \cdot \log(n)) + n + O(n \cdot \log(n)) = O(n \cdot \log(n)) \qquad (4)$$

steps, which is much faster than $O(n^2)$.

V. Strassen was the first to notice, in 1968, than this idea can lead to fast multiplication of long integers. Indeed, an integer x in a number system with base b can be represented as a sum

$$x = \sum_{i=1}^{n} x_i \cdot b^i. \qquad (5)$$

In these notations, the product $z = x \cdot y$ of two integers $x = \sum_{i=1}^{n} x_i \cdot b^i$ and $y = \sum_{i=1}^{n} y_i \cdot b^i$ can be represented as

$$z = \left(\sum_{i=1}^{n} x_i \cdot b^i \right) \cdot \left(\sum_{j=1}^{n} y_j \cdot b^j \right) =$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_i \cdot y_j \cdot b^i \cdot b^j = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i \cdot y_j \cdot b^{i+j}. \qquad (6)$$

By combining terms at different values b^k, we conclude that

$$z = \sum_{k=1}^{n} z_k \cdot b^k, \qquad (7)$$

where

$$z_k = \sum_i x_i \cdot y_{k-i}. \qquad (8)$$

This is convolution, and we know that convolution can be computed in time $O(n \cdot \log(n))$.

The values z_k are not exactly the digits of the desired number z, since the sum (8) can exceed the base b. Thus, some further computations are needed. However, even with these further computations, we can multiply two numbers in almost the same time

$$O(n \cdot \log(n) \cdot \log(\log(n))). \qquad (9)$$

The corresponding algorithm, first proposed by A. Schönhage and V. Strassen in their 1971 paper [5], remains the fastest known – and it is much faster than the standard $O(n^2)$ algorithm.

7 Fast Multiplication: Open Problems

Fast algorithms drastically reduced the computation time. Fast Fourier transform is one of the main tools for signal processing. These successes has led to the need to find faster and faster algorithms. From this viewpoint, it is is desirable to look for even faster algorithms for multiplying numbers.

The fact that researchers succeeded in discovering algorithms which are much faster than traditional multiplication gives us hope that even faster algorithms can be found.

8 Interest in Fast Human Calculators Revived

Where to look for these algorithms? One natural source is folks who did compute fast. From this viewpoint, the fact that they were unable to clearly explain what algorithm they used becomes an advantage: maybe the algorithm that they actually used is some fast multiplication algorithm? This possibility revived an interest in fast human calculators.

So, the question is:

- did fast human calculators use fast multiplication algorithm(s), or

- they used the standard algorithm but simply performed operations with digits faster?

9 Direct Analysis Is Impossible

The most well-known fast human calculator, Johann Martin Zacharias Dase, died almost 150 years ago. Even when he was alive, his self-descriptions were not sufficient to find out how exactly he performed the computations. There might have been hope that our knowledge of fast multiplication algorithms can help in this understanding, but it did not work out. In other words, the direct analysis of Dase's behavior has been impossible – and it is still impossible. We therefore need to perform an *indirect* analysis.

10 Indirect Analysis: Main Idea

A natural way to check which algorithm is used by a computational device – be it a human calculator or an electronic computer – is to find out how the computation time changes with the size n (= number of digits) of the numbers that we are multiplying.

- If this computation time grows with n as n^2, then it is reasonable to conclude that the standard algorithm is used – since for this algorithm, the computation time grows as n^2.

- On the other hand, if the computation time grows with n as $\approx n \cdot \log(n)$ (or even slower), then it is reasonable to conclude that a fast multiplication algorithm is used.

  - It maybe a Strassen-type algorithm, for which the computation time grows as $n \cdot \log(n)$.

  - It may be a (yet unknown) faster algorithm in which case the computation time grows even slower than $n \cdot \log(n)$.

11 Data That We Can Use

Interestingly, there is a data on the time that Dase needed to perform multiplication of numbers of different size. This data comes from fact that Dase's performance was analyzed and tested by several prominent mathematicians of his time – including Gauss himself. Specifically:

- Dase multiplied two 8-digit numbers in 54 seconds;

- he multiplied two 20-digit numbers in 6 minutes;

- he multiplied two 40-digit numbers in 40 minutes; and

- he multiplied two 100-digit numbers in 8 hours and 45 minutes.

12 Analysis

For the standard multiplication algorithm, the number of computational steps grows with the numbers size n as n^2. Thus, for the standard multiplication algorithm, the computation time for performing the computation also grows as n^2:

$$t(n) = C \cdot n^2. \qquad (10)$$

So, for this algorithm, two different number sizes $n_1 < n_2$, we would have $t(n_1) = C \cdot n_1^2$ and $t(n_2) = C \cdot n_2^2$ and thus,

$$\frac{t(n_2)}{t(n_1)} = \frac{n_2^2}{n_1^2} = \left(\frac{n_2}{n_1}\right)^2. \qquad (11)$$

For a faster algorithm, e.g., for the algorithm that requires $O(n \cdot \log(n))$ running time, the corresponding ratio will be smaller:

$$\frac{t(n_2)}{t(n_1)} = \frac{n_2 \cdot \log(n_2)}{n_1 \cdot \log(n_1)} = \frac{n_2}{n_1} \cdot \frac{\log(n_2)}{\log(n_1)}. \qquad (12)$$

Thus, to check whether a human calculators uses the standard algorithm or a faster one it is sufficient to compare the the corresponding time ratio $\frac{t(n_2)}{t(n_1)}$ with the square $\left(\frac{n_2}{n_1}\right)^2$:

- If the time ratio is smaller than the square, this means that the human calculator used an algorithm which is much faster than the standard one.

- On the other hand, if the time ratio is approximately the same as the square, thus means that the human calculators most probably used the standard algorithm – or at least some modification of it that does not drastically speed up the computations.

- If it turns out that the time ratio is larger than the square, this would mean, in effect, that a human calculator used an algorithm which is asymptotically even slower than the standard one – this can happen, e.g., if there is an overhead needed to store values etc.

According to the above data, we have:

- $t(8) = 0.9$ minutes,

- $t(20) = 6$ minutes,

- $t(40) = 40$ minutes, and

- $t(100) = 8 \cdot 60 + 45 = 525$ minutes.

Here, for $n_1 = 8$ and $n_1 = 20$, we have:

$$\frac{t(n_2)}{t(n_1)} = \frac{t(20)}{t(8)} = \frac{6}{0.9} \approx 6.7 > \left(\frac{n_2}{n_1}\right)^2 =$$

$$\left(\frac{20}{8}\right)^2 = 2.5^2 = 6.25. \qquad (13)$$

For $n_1 = 8$ and $n_2 = 40$, we have

$$\frac{t(n_2)}{t(n_1)} = \frac{t(40)}{t(8)} = \frac{40}{0.9} \approx 44 > \left(\frac{n_2}{n_1}\right)^2 =$$

$$\left(\frac{40}{8}\right)^2 = 5^2 = 25. \tag{14}$$

Finally, for $n_1 = 8$ and $n_2 = 100$, we have

$$\frac{t(n_2)}{t(n_1)} = \frac{t(100)}{t(8)} = \frac{525}{0.9} \approx 583 > \left(\frac{n_2}{n_1}\right)^2 =$$

$$\left(\frac{100}{8}\right)^2 = 12.5^2 \approx 156. \tag{15}$$

In all these cases, the computation time of a human calculator grows faster then n^2 corresponding to the standard algorithm.

The same conclusion can be made if instead of comparing each value n with the smallest value n_1, we compare these values with each other. For $n_1 = 20$ and $n_2 = 40$, we have

$$\frac{t(n_2)}{t(n_1)} = \frac{t(40)}{t(20)} = \frac{40}{6} \approx 7 > \left(\frac{n_2}{n_1}\right)^2 =$$

$$\left(\frac{40}{20}\right)^2 = 2^2 = 4. \tag{16}$$

For $n_1 = 20$ and $n_2 = 100$, we have

$$\frac{t(n_2)}{t(n_1)} = \frac{t(100)}{t(20)} = \frac{525}{6} \approx 88 > \left(\frac{n_2}{n_1}\right)^2 =$$

$$\left(\frac{100}{20}\right)^2 = 5^2 = 25. \tag{17}$$

Finally, for $n_1 = 40$ and $n_2 = 100$, we have

$$\frac{t(n_2)}{t(n_1)} = \frac{t(100)}{t(40)} = \frac{525}{40} \approx 13 > \left(\frac{n_2}{n_1}\right)^2 =$$

$$\left(\frac{100}{40}\right)^2 = 2.5^2 = 6.25. \tag{18}$$

Thus, it is reasonable to conclude that the fast human calculators did not use any algorithm which is faster than the standard one.

13   Possible Future Work

People who perform computations fast appear once in a while. It may be a good idea to record and analyze their computation time − and maybe record their fuzzy explanations and try to make sense of them.

Acknowledgment

References

[1] P. Beckmann, *A History of π*, Barnes & Noble, New York, 1991.

[2] Th. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 2001.

[3] D. R. Hofstadter, *Godel, Escher, Bach: an Eternal Golden Braid*, Basic Books, 1999.

[4] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic*, Prentice Hall, New Jersey, 1995.

[5] A. Schönhage and V. Strassen, "Schnelle Multiplikation grosser Zahlen", *Computing*, 1971, Vol. 7, pp. 281–292.