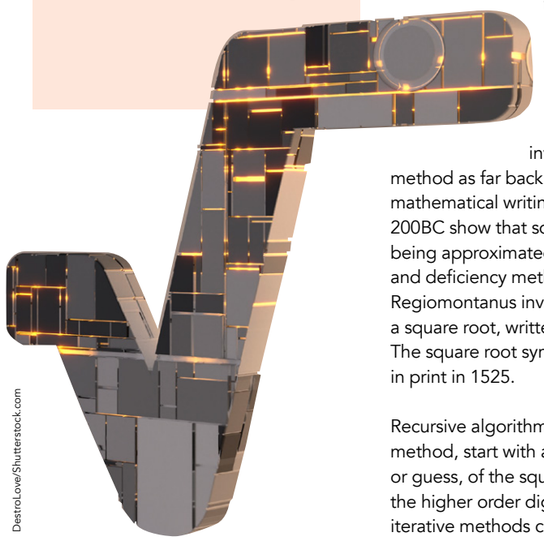
Discovering perfect squares and building square roots

Mathematicians have been faced with the problem of finding perfect squares and their roots since ancient times. Recent findings in computational number theory have enabled the development of efficient algorithms for discovering square numbers. Professor Philip Brown from the Department of Foundational Sciences (Mathematics) at Texas A&M University Galveston Campus has developed a new algorithm that can detect a perfect square and build its root using binary arithmetic. Professor Brown demonstrates how his algorithm is easily implemented and can test numbers with millions of digits.



DestroLove/Shutterstock.com

The problem of finding perfect squares, or square numbers that result from a number being multiplied by itself, together with the converse issue of finding their square roots, has been challenging mathematicians since ancient times.

A SQUARE HISTORY

No one actually knows who invented the square root, but it is thought that the knowledge of square roots originally came from dividing areas of land into equal parts so that the length of the side of a square became the square root of its area. The Babylonians and Greeks have been credited with the discovery of Heron's method, the precursor of Newton's iterative method, although Indian mathematicians are thought to have used a similar system around 800BC. The Egyptians calculated square roots using an inverse proportion method as far back as 1650BC. Chinese mathematical writings from around 200BC show that square roots were being approximated using an excess and deficiency method. In 1450AD Regiomontanus invented a symbol for a square root, written as an elaborate R. The square root symbol √ was first used in print in 1525.

Recursive algorithms, such as Newton's method, start with an approximation, or guess, of the square root and find the higher order digits first. Such iterative methods can be carried out on a computer using floating point arithmetic, but they are usually difficult to implement for very large numbers and computational difficulty can arise with the division operation. More recently, computational number theory, the area of number theory concerned with finding and implementing efficient computer algorithms, has enabled the development of algorithms involving sieve methods to decide whether or not a positive integer is a perfect power.

Prof Philip Brown from the Department of Foundational Sciences (Mathematics) at Texas A&M University Galveston Campus has developed a new algorithm for discovering square numbers. Combining elementary number theory and algorithmic number theory, this novel algorithm uses only addition, subtraction and multiplication, so the potentially problematic division operation is not required. The algorithm is easily implemented and can test numbers with millions of digits. Furthermore, this new algorithm tests with certainty whether or not a number is a perfect square, in contrast with other methods that can only test to a high degree of probability.

Konnor Chappell (a student at Texas A&M University at Galveston) helped Professor Brown write the Python code in order to implement the algorithm.

REVEALING PROPERTIES OF PERFECT SQUARES

Prof Brown has observed that some properties of perfect squares are revealed if the numbers are expressed in a base that is a power of 2, such as in the binary, octal and hexadecimal number systems. This underpins how his algorithm can identify a perfect square and build its square root, starting with the low order digits and working through to the high order digits.

PabloLagarto/Shutterstock.com

Because the algorithm builds square roots starting with the lower order digits, it is easier to comprehend if we read or label the digits from right to left.

Prof Brown demonstrates how the algorithm starts by converting the input number *N* from base 10 to base 2. If *N* is an even number, then its base 2 expression begins (on the right hand side) with a string of binary digits that are all equal to 0, e.g. the decimal numbers 4 and 20 are expressed as 100 and 10100 in base 2, respectively. When the input number, *N*, is an even number, the initial string of zeros of the binary representation of the number is truncated, leaving the resulting odd number to be tested. If this odd number is a square, then *N* is either a square or twice a square. This means that the algorithm only needs to continue when *N* is odd. So given an odd integer *N* in octal format the algorithm will determine if *N* is a perfect square and compute the square root if required.

OBSERVATIONS WITH BASE 8

Base 8 is a convenient number system to demonstrate Prof Brown's algorithm as it is closest to base 10. Multiplication by 4 in base 8, e.g. 1 x 4 = 4, 2 x 4 =10, always results in a number with a 0 or 4 on the right-hand side. Consequently, all even perfect squares expressed in base 8 begin (reading from right to left) with a 0 or 4. Prof Brown also shows that all odd perfect squares expressed in base 8 begin with a 1.

BASE 2^s

Moreover, Prof Brown demonstrates that for bases that are higher powers of 2, denoted 2^s, if the first base 2^s digit of some number *N* is not congruent (modulo 2^s) to the square of an odd number, then the number *N* is not a square number.

TIME COMPLEXITY

The time complexity of an algorithm quantifies the amount of time that an algorithm takes to run as a function of the length of the input. Prof Brown's algorithm tests whether a positive integer *N* is a square number, and/or computes the square root of *N* has time complexity of $O((\log^2 N)/s)$ where *s* is the power of the chosen base – for example, $s = 3$ for base 8 and $s = 4$ for base 16. This 'big O' notation means that the algorithm's performance is

Professor Brown has developed a theoretical basis for this algorithm that provides new insight into the properties of square numbers.

Regiomontanus (left) is considered the inventor of the square root symbol. Isaac Newton (right) developed a recursive algorithm laying the foundation for computational number theory.

Perfect squares are important numbers in a range of mathematical fields.

Jirsak/Shutterstock.com

directly proportional to $(\log^2 N)/s$, making the algorithm extremely efficient, especially when dealing with large numbers.

The time complexity is inversely proportional to *s*, so while it is a programming decision, the algorithm will be more efficient if the user chooses *s* to increase with *N*. In trials with input numbers *N* ranging from 1000 to 512000 digits, Prof Brown has observed that there appears to be optimal values of *s* that are proportional to the logarithm of the number of digits of the number *N* being tested.

DEVELOPMENTS

Prof Brown has developed a theoretical basis for this algorithm that provides new insight into the properties of square numbers using binary, octal and hexadecimal arithmetic. This work can also be extended to other number systems with bases that are even larger powers of 2.

The new algorithm can detect a perfect square and build its root using binary arithmetic.

The new algorithm can detect a perfect square and build its root using binary arithmetic. It is relatively straightforward to use and comparable in computational complexity and storage space requirements to other algorithms, with the advantage that the results are presented with certainty and require no interpretation of probability.

Within the algorithm, Prof Brown offers the user a number of options to tailor the algorithm to suit their individual requirements. For instance, the programmer can opt to solve for multiplicative inverses by using pre-computed selection matrices or deploying the extended Euclidean algorithm. They also have the choice of implementing the algorithm in base 8 or any other base that is a higher power of 2, which may depend on the size of the number to be tested.

ONGOING AND FUTURE WORK

The algorithm runs efficiently using number systems where the base is a power of 2. Currently, there is no straightforward way of employing other bases such as base 10. Prof Brown is busy developing a recursive implementation of the binary algorithm that uses a smaller value of *s* at each level of recursion, beginning with a value of *s* proportional to log *N*. This algorithm has an improved time complexity of $O(\log N \log^2 \log N)$ and it is possible to test numbers with billions of digits using a desktop computer in less than one hour. He is also considering whether the algorithm can be extended to test for perfect powers greater than 2, such as perfect cubes.

Binary System Table

Maxal Tamor/Shutterstock.com

	16	8	4	2	1
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1

	16	8	4	2	1
16	1	0	0	0	0
17	1	0	0	0	1
18	1	0	0	1	0
19	1	0	0	1	1
20	1	0	1	0	0
21	1	0	1	0	1
22	1	0	1	1	0
23	1	0	1	1	1
24	1	1	0	0	0
25	1	1	0	0	1
26	1	1	0	1	0
27	1	1	0	1	1
28	1	1	1	0	0
29	1	1	1	0	1
30	1	1	1	1	0
31	1	1	1	1	1

Professor Brown's algorithm uses binary arithmetic.

Behind the Research

Professor Philip Brown

E: brownp@tamug.edu **T:** +1 409 599 3579 **F:** +1 409 741 4388
W: http://www.tamug.edu/fsci/faculty-bios/BrownP.html

Research Objectives

Professor Brown has developed a new algorithm for discovering square numbers.

Detail

Philip Brown
2809 Ball Street
Galveston
Texas 77550
USA

Bio

After completing his undergraduate studies in Johannesburg, South Africa, Prof Brown enrolled in the mathematics graduate program at Texas A&M University. His PhD dissertation and subsequent research work has been primarily in the study of functions of complex numbers. He has published a textbook titled *Foundations of Mathematics*.

Collaborators

Konnor Chappell (a student at Texas A&M University at Galveston) helped write the Python code in order to implement the algorithm.

References

Brown, P.R. (2019). Detecting square numbers. *Quaestiones Mathematicae*, [online]. Available at: https://doi.org/10.2989/16073606.2019.1678530

Bernstein, D.J. (1998). Detecting perfect powers in essentially linear time. *Mathematics of Computation*, 67(223), 1253-1283.

Bach, E. and Sorenson, J. (1993). Sieve Algorithms for Perfect Power Testing. *Algorithmica*, 9, 13-328.

Personal Response

What initially inspired you to develop a new algorithm for discovering square numbers?

"I enjoy working on fundamental problems in mathematics and science. During my career I have published research work relating to certain fundamental constants of mathematics (including the number π) and physics (including the fine structure constant). I became interested in the problem of detecting square numbers when my mother, Ria Brown, who was a high school mathematics teacher, pointed out to me a pattern in the digits of perfect squares."