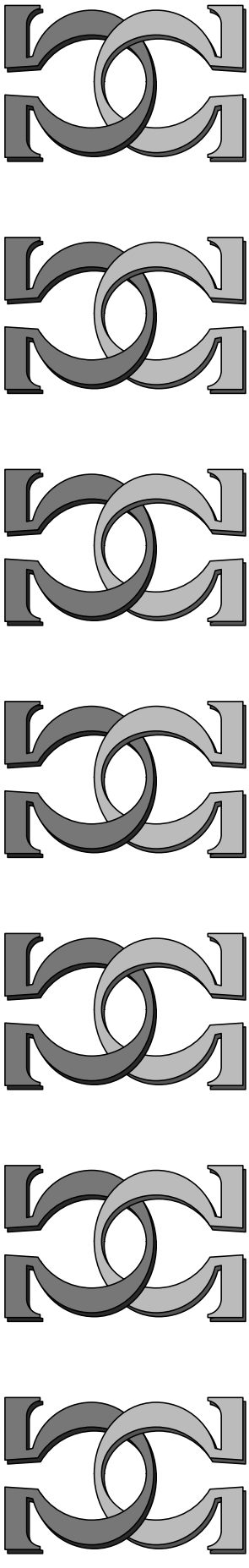

CDMTCS
Research
Report
Series

What Is the Value of $Taxicab(6)$? An Update

Cristian S. Calude[1],
Elena Calude[2],
Michael J. Dinneen[1]

[1]University of Auckland, NZ,
[2]Massey University at Albany, NZ

CDMTCS-261
April 2005

Centre for Discrete Mathematics and
Theoretical Computer Science

What Is the Value of $Taxicab(6)$? An Update

Cristian S. Calude
Department of Computer Science, University of Auckland, New Zealand
Email: cristian@cs.auckland.ac.nz

Elena Calude
Institute of Information Sciences, Massey University at Albany, New Zealand
Email: e.calude@massey.ac.nz

Michael J. Dinneen
Department of Computer Science, University of Auckland, New Zealand
Email: mjd@cs.auckland.ac.nz

April 29, 2005

Abstract

The famous story of the number 1729, the smallest integer which can be expressed as the sum of two positive cubes in two different ways, motivated the introduction of Taxicab Numbers. The smallest number expressible as the sum of two cubes in n different ways is called $Taxicab(n)$. So, $Taxicab(2) = 1729$. Further on, $Taxicab(5) = 48988659276962496$. Computing $Taxicab(n)$ is challenging and interesting, both from mathematical and programming points of view.

The exact value of $Taxicab(6)$ is not known; in view of the results obtained by Bernstein [1] and Rathbun [14] it follows that $Taxicab(6)$ is in the interval $[10^{18}, 24153319581254312065344]$. In [5] we proved that with probability greater than 99%, $Taxicab(6) = 24153319581254312065344$.

In this note we improve the method used in [5] in two ways: we use (1) a larger, and (2) a better quality random sampling, namely, a sample of 562,500 quantum random integers drawn from the above mentioned interval using Quantis, [10]. As a result, *we prove that the above value for* $Taxicab(6)$ *is true with probability greater than 99.8%.*

Keywords: Taxicab Numbers, Quantum random integer, sampling

AMS Subject Classification: 11-04, 62D05, 68Q30

1 Taxicab Numbers

The smallest number expressible as the sum of two cubes in n different ways is called the Taxicab Number of order n, $Taxicab(n)$. Hardy and Wright ([9]; Theorem 412) proved that $Taxicab(n)$ exists for every positive integer n, but the proof is of little use in finding the number.

The first two Taxicab Numbers are $Taxicab(1) = 2 = 1^3 + 1^3$ and $Taxicab(2) = 1729 = 1^3+12^3 = 9^3+10^3$, the Hardy-Ramanujan Number. In 1957, Leech [12] computed $Taxicab(3) = 87539319 = 167^3 + 436^3 = 228^3 + 423^3 = 255^3 + 414^3$, in 1991 Rosenstiel, Dardis, and Rosenstiel [15] (see also Butler's program [2]) showed that $Taxicab(4) = 6963472309248 = 2421^3+19083^3 = 5436^3+18948^3 = 10200^3+18072^3 = 13322^3+16630^3$, in 1997 Wilson [17] discovered $Taxicab(5) = 48988659276962496 = 38787^3 + 365757^3 = 107839^3 + 362753^3 = 205292^3 + 342952^3 = 221424^3 + 336588^3 = 231518^3 + 331954^3$ and in 1998 Bernstein (see [1]) verified that $Taxicab(6) \geq 10^{18}$, and in 2002 Rathbun [14] has found a six-way sum of cubes:

$$\begin{aligned} Taxicab(6) \leq 24153319581254312065344 &= 28906206^3 + 582162^3 \\ &= 28894803^3 + 3064173^3 \\ &= 28657487^3 + 8519281^3 \\ &= 27093208^3 + 16218068^3 \\ &= 26590452^3 + 17492496^3 \\ &= 26224366^3 + 18289922^3. \end{aligned}$$

In 2003 we proved (see [5]) that with probability greater than 99%, $Taxicab(6) = 24153319581254312065344$; the method used a sample of 22,500 pseudo-random integers in the interval $[10^{18}, 24153319581254312065344)$ produced by Mathematica.

2 The Sampling Method

There are various approaches to the computation of $Taxicab(n)$, such as [18, 13, 16, 19, 1]. The main idea is to use an efficient codification and a "computational structure" that supports insertion of new elements and removal of the smallest element (somethimes referred to as "priority queues"). None of these approaches seem to work for the calculation of $Taxicab(6)$. So, in what follows we are going to use the sampling approach (see [6]) described in [5]. Naturally, the result will not be exact, but the error will be less than 0.02%.

Here is the method used in [5]. Given a finite population of size N, we will estimate 3 proportions associated with 3 binary random variables, $P_i = P(X_i = 1), i = 1, 2, 3$, using a quantum random sample of size n. Each estimate should be correct within $\pm c\%$ in the sense that, if the sample shows p_i to be P_i, the percentage for the whole population is "sure" to lie between $p_i - c\%$ and $p_i + c\%$ (with "accuracy within $c\%$"). As we cannot guarantee an accuracy within $c\%$, we accept a probability α (0.0027 in our case) of getting "an unlucky sample" (which is in error by more than the desired $c\%$).

The process of random generation of the sample is equivalent with sampling with replacement (generated units are independent, repetitions are possible), according to a Binomial scheme. If $(u_1, \ldots, u_n)$ are generated items, we denote by m_i the number of items for which $X_i(u) = 1$, $i = 1, 2, 3$. Then, the estimates of P_i are $p_i = \frac{m_i}{n}, i = 1, 2, 3$. For a large value of n ($n > 100$), one can use the normal approximation for p_i, that is, p_i is approximately normal distributed $N\left(P_i, \frac{P_i(1-P_i)}{n}\right)$.

In order to estimate the value of n, we start from the simultaneous conditions

$$\Pr\left(|p_i - P_i| \geq z_{1-\frac{\alpha}{2}}\sqrt{Var\left(p_i\right)}\right) = \alpha, i = 1, 2, 3,$$

where $z_{1-\frac{\alpha}{2}}$ is the $(1-\frac{\alpha}{2})$–quantile of the $N(0,1)$ distribution, and $z_{0.99865} \approx 3$ (see, for instance, [8], Table II).

Accordingly, we will have

$$z_{1-\frac{\alpha}{2}}\sqrt{\frac{P_i\left(1-P_i\right)}{n}} = c, i = 1, 2, 3,$$

hence, the sample size is

$$\hat{n} = \max\left\{\frac{z^2_{1-\frac{\alpha}{2}} P_i\left(1-P_i\right)}{c^2}, i = 1, 2, 3\right\}.$$

The value of $\hat{n}$ depends on the unknown proportions P_i. As we don't have any knowledge regarding P_i, we will choose the "critical" value $P = 50\%$ (which maximizes the product $P(1-P)$). Hence, the "safest" estimation of the sample size is $\hat{n} = (z^2_{1-\frac{\alpha}{2}} \cdot 2500) \cdot c^{-2}$.

For our level of significance $\alpha = 0.0027$, one gets $z_{1-\frac{\alpha}{2}} = 3$. For an accuracy within $c = 0.02\%, \hat{n} = 562,500$, which is the size of the sample investigated by our program. (To get an accuracy of 0.01% we need to test a sample of size 2,250,000.)

3 Program, Quantum Random Sample and Conclusion

We ran our program (presented in [5]) on a sample of 562,500 quantum random integers in the interval $[10^{18}$, 24153319581254312065344) and we found no number satisfying the required condition, consequently, *with probability greater than 99.8%,* $Taxicab(6) = 24153319581254312065344$.

Obviously, the accuracy of the sampling method depends upon the "quality of the randomness" of the sample (see [7] for pitfalls in using traditional pseudo-random number generation techniques). The sampling used in this simulation was produced with the quantum random generator Quantis, [10].

Quantis is a physical random number generator based on an elementary quantum optics process. Photons are sent one by one onto a semi-transparent mirror and detected; the exclusive events (reflection vs. transmission) produce the quantum random bits "0"

and "1". The operation of Quantis is continuously monitored to ensure immediate detection of a failure and disabling of the quantum random bit stream. Quantum randomness passes all statistical testes for randomness, see [10]. No algorithm can produce infinitely many quantum random bits; see more in [4].

Using a supply of quantum random bits from this device, we elected to use our own procedure to generate integers within a range [*low*, *high*]. The Quantis software library [11] provides such a method. But, on a closer inspection of the source code, we decided the following simple algorithm is better because it avoids floating point operations for rounding. We embed the interval [*low*, *high*] into [*low*, *low* + 256^N], where $N = \lceil \log_{256}(high - low) \rceil + 1$. The algorithm extracts quantum random bits with `quantis.readByte()`, generates quantum random integers in the range of [*low*, *low* + 256^N], and skips integers larger than *high* to get quantum random integers in the interval [*low*, *high*].

In our case, $low = 10^{18}$, $high = 24153319581254312065344$, hence $N = 10$. A quantum random integer in the interval $[10^{18}, 10^{18} + 256^{10}]$ lies in the interval [*low*,*high*] with probability $p \approx 0.0199$. The geometric distribution can be used to model the number of quantum integers we need to generate in the larger interval to obtain one quantum random integer in [*low*, *high*]: we consider the probability p that the generated integer is in the interval [*low*, *high*], so $1 - p$ is the probability that the generated integer is in the interval (*high*, $10^{18} + 256^{10}$]; in the first case the integer is accepted while in the second case the integer is discarded. The expected value for the number of generated quantum random integers to hit the target [*low*, *high*] is $\sum_{n=1}^{\infty} n(1-p)^{n-1}p = 1/p \approx 50.0542$, that is, about 50.

The following program generates an infinite sequence of quantum random integers in the interval [*low*, *high*]. Naturally, the program was stopped when we reached our target of 562,500 quantum random integers.

```
Function RandomInteger(Integer low, Integer high)
    Integer rnum = 0
    repeat for ⌈log_256(high − low)⌉ + 1 steps
        rnum = rnum * 256 + quantis.readByte()
    end
    if rnum ≤ high − low return low + rnum
    else return RandomInteger(low, high)
end
```

The correctness of the program follows from the following two results (see [3]): (a) if $x_1x_2\cdots x_n\cdots$ is a binary (algorithmic) random sequence, then for every $k \geq 1$ the sequence over the alphabet $\{0,1\}^k$, $(x_1x_2\cdots x_k)(x_{k+1}x_{k+2}\cdots x_{2k})(x_{2k+1}x_{2k+2}\cdots x_{3k})\cdots$ is also (algorithmic) random, (b) if $x_1x_2\cdots x_n\cdots$ is an (algorithmic) random sequence over the alphabet $A \cup \{a\}$ (here A is an alphabet with at least two elements), then consistently deleting all occurrences of a in the sequence we obtain a sequence over A which is still (algorithmic) random.[1]

[1] In our case A has $256^{10} + 1$ elements.

When doing our reinvestigation for $Taxicab(6)$, we divided a sample of 562,500 quantum random integers into 10 pieces and ran our program concurrently on different computers[2] (which consisted of two Mac OSX G5s and four Linux Intel CPU boxes, some computers having multiple processors). The overall computation time took about 3 weeks, with some jobs finishing in about 2 weeks. Both our source code and the actual quantum random integers we used are available for downloading at [21].

Acknowledgment

We thank P. Dance and R. Burrowes for assistance with the Quantis generator. We thank the anonymous referee for useful comments which improved the presentation.

References

[1] D. J. Bernstein. Enumerating solutions to $p(a) + q(b) = r(c) + s(d)$, *Mathematics of Computation* 70, 233 (2000), 389–394.

[2] B. Butler. C Program for Ramanujan Quadruples, `http://www.durangobill.com/Rama4.html`, 30 May 2001.

[3] C. S. Calude. *Information and Randomness: An Algorithmic Perspective*, 2nd Edition, Revised and Extended, Springer-Verlag, Berlin, 2002.

[4] C. S. Calude. Algorithmic randomness, quantum physics, and incompleteness, in M. Margenstern (ed.). *Proceedings of the Conference "Machines, Computations and Universality" (MCU'2004)*, Lectures Notes in Comput. Sci. 3354, Springer, Berlin, 2005, 1–17.

[5] C. S. Calude, Elena Calude, M. J. Dinneen. What is the value of $Taxicab(6)$?, *J. UCS* 9, 10 (2003), 1196–1203.

[6] W. G. Cochran. *Sampling Techniques*, John Wiley, New York, 1977 (third edition).

[7] D. E. Eastlake 3rd, S. Crocker, J. Schiller, *Randomness Recommendations for Security*, RFC 1750, December 1994, 30 pp.

[8] A. Hald. *Statistical Tables and Formulas*, John Wiley, New York, 1965.

[9] G. H. Hardy, E. M. Wright, *An Introduction to the Theory of Numbers*, Oxford University Press, London, 1954, 3rd edition.

[10] `http://www.idquantique.com/qrng.html`.

[11] `http://www.idquantique.com/files/quantislibrary.pdf`.

[2] Our program runs on any platform that can use the GMP library to represent arbitrarily large integers [20].

[12] J. Leech. Some solutions of Diophantine equations, *Proc. Cambridge Phil. Soc.* 53 (1957), 778–780.

[13] I. Peterson. Taxicab Numbers, *Science News Online*, http://www.sciencenews.org/20020727/mathtrek.asp.

[14] R. L. Rathbun. Sixth Taxicab Number?, http://listserv.nodak.edu/scripts/wa.exe?A2=ind0207&L=nmbrthry&P=R530, July 16, 2002.

[15] E. Rosenstiel, J. A. Dardis, C. R. Rosenstiel. The four least solutions in distinct positive integers of the Diophantine equation $s = x^3 + y^3 = z^3 + w^3 = u^3 + v^3 = m^3 + n^3$, *Bull. Inst. Math. Appl.* 27 (1991), 155–157.

[16] W. Schneider. Taxicab Numbers, http://www.wschnei.de/number-theory/taxicab-numbers.html, 3 February 2003.

[17] D. W. Wilson. The fifth Taxicab Number is 48 988 659 276 962 496, *J. Integer Seq.* 2 (1999), Article 99.1.9, 1, http://www.research.att.com/~njas/sequences/JIS/wilson10.html.

[18] Sequence A011541, *On-Line Encyclopedia of Integer Sequences*, http://www.research.att.com/cgi-bin/access.cgi/as/njas/sequences/eisA.cgi?Anum=A011541.

[19] The Taxicab Problem, http://euler.free.fr/taxicab.htm.

[20] The GNU GMP library, http://www.swox.com/gmp/.

[21] ftp://ftp.cs.auckland.ac.nz/pub/research/CDMTCS/taxicab/.