The Stick Problem

Augustine Bertagnolli
Auburn University Montgomery

October 25, 2013

Abstract

Given sticks of possible sizes one through six, what is the smallest number of sticks you can have to ensure that you are able to form a perfect square? The Pigeonhole Principle tells us that if we have nineteen sticks we would have at least four of one of the sizes, but can we do better if we take partitions into account? This is one case of the stick problem which, though simple in statement, proves to be not so simple in solution. In this paper, we define the stick problem clearly, discuss our methods for approaching and simplifying the problem, provide an algorithm for generating solutions, and present some computer generated solutions for specific cases.

1 Introduction

Some of the most popular problems in mathematics are ones which are simple in statement yet not so simple in solution. Examples can be found in some currently unsolved problems in mathematics, such as the Collatz conjecture and Goldbach's conjecture. The former, proposed by Lothar Collatz in 1937, states:

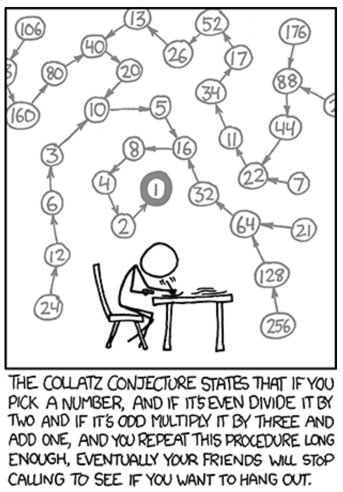

106
13
52
176
40
80
20
26
17
88
160
10
5
34
44
3
8
16
11
22
7
4
1
32
6
2
64
21
12
128
24
256
THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

Figure 1: `http://xkcd.com/710/`

> Given any natural number as a starting point, if you follow the sequence of operations
> (1) If even, divide by 2,
> (2) If odd, multiply by 3 and add one,
> indefinitely, you will always arrive to 1.

For example, if you start with 7, the sequence would be 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1. Since its statement, this problem has received enormous attention from mathematicians around the world, but it has yet to be proved or disproved. [3]

The latter conjecture mentioned, Goldbach's conjecture, dates back to the 18th century with the German mathematician Christian Goldbach who proposed that every even integer greater than 2 can be written as the sum of two primes. For example, $24 = 13 + 11$. Again, although receiving perhaps the most attention of any problem in pure mathematics, the conjecture remains neither proven nor disproven. [4]

While it is not the purpose of this paper to expound on such unsolved problems or to even suggest that the stick problem approaches them in complexity, it is worth noting the simplicity with which these problems are stated as well as the seeming lack of "practical" application. To this we can only echo the sentiment of the great mathematician and physicist Carl Friedrich Gauss, as quoted in [1], that "the greatest thing is purely mathematical thinking; this is worth much more than the application of mathematics." And so we present the stick problem with the simple purpose of making the problem clear for anyone interested in taking part in its solution as well as sharing what work has been done thus far.

2 The Problem

2.1 A Question

Imagine you are given a standard six-sided die and told that each time you roll the die you will receive a stick the length of the number you rolled. The question is: *What is the minimum number of times you must roll the die in order to ensure that you can form a perfect square out of your sticks?*

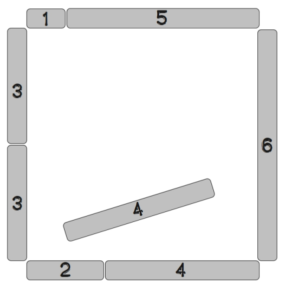


Figure 2: Forming a square from the roll sequence 4,1,4,3,3,6,5,2.

If you have 19 sticks, by the Pigeonhole Principle you will be sure to have at least 4 of the same size sticks and therefore able to make a perfect square. This is because under the worst case you might obtain 3 of each size 1 through 6 which gives you $3 \times 6 = 18$ sticks, then on the 19th roll you will be sure to gain a 4th of one of the sizes. Unfortunately, the question we asked was not so simple. You do not have to form the square out of the same size sticks. For instance, if you roll the sequence 4, 1, 4, 3, 3, 6, 5, 2, you could form a square with side length 6 by combining sticks in the following way: 4+2, 1+5, 3+3, and 6, even though you only have at most 2 sticks of the same size.

The short answer to the question is 10. If you roll the die 10 times–that is, you have 10 sticks–you will always be able to form a perfect square from those sticks no matter what size sticks you have. This solution was achieved by nothing other than brute force. But naturally, the next question becomes: What if we have sticks of possible sizes 1 to n, and we are trying to make a k-sided polygon?

2.2 The Stick Problem

We define the **stick problem** to be:

> Given possible stick sizes one through n, what is the minimum number of sticks needed to ensure the ability to form a k-sided polygon (or simply 1 or 2 sides of the same length for $k = 1$ or 2)?

Before converting this problem into its purely mathematical version, we will first define and discuss partitions.

2.3 Partitions

As defined in [7], a **partition** is "a way of writing an integer n as a sum of positive integers where the order of the addends is not significant." For example, the partitions of 4 are 1+1+1+1, 1+1+2, 1+3, 2+2, and 4.

These can be written as the sequences (1, 1, 1, 1), (1, 1, 2), (1, 3), (2, 2), and (4). Every partition of a number n corresponds to one and only one solution set $(x_1, x_2, \ldots, x_n)$ to the equation

$$x_1 + 2x_2 + 3x_3 + \cdots + nx_n = n,$$

where $x_1, x_2, \ldots, x_n$ are nonnegative integers. For instance, the partitions of 4 as given correspond, respectively, to the solutions $(x_1, x_2, x_3, x_4) = (4, 0, 0, 0), (2, 1, 0, 0), (1, 0, 1, 0), (0, 2, 0, 0)$, and $(0, 0, 0, 1)$. [7]

Much work has been done on the partition function $P(n)$ which counts the number of partitions for a given number n. For example, in the previous paragraph we counted 5 partitions of the number 4. Thus, $P(4) = 5$. If the reader is interested, he can check [8] for a wonderful review of the various formulae and identities for $P(n)$.

2.4 The Stick Problem (A Mathematical Equivalent)

Let $A(n, k)$ be the minimum number s such that for all nonnegative solutions $(a_1, a_2, \ldots, a_n)$ with $a_1 + a_2 + \cdots + a_n = s$, there exist some nonnegative integers $x_{i,j}$ such that

$$\begin{aligned}
x_{1,1} + 2x_{1,2} + \cdots + nx_{1,n} &= t, \\
x_{2,1} + 2x_{2,2} + \cdots + nx_{2,n} &= t, \\
&\vdots \\
x_{k,1} + 2x_{k,2} + \cdots + nx_{k,n} &= t,
\end{aligned}$$

for some positive integer t, and $X_j = \sum_{i=1}^{k} x_{i,j} \leq a_j$ for all j.

In this formulation, t is the length of each side of the k-gon, s is the total number of sticks, a_i is the number of sticks of size i, and each t is formed by a partition of t with stick size no greater than n. In forming the k sides of length t, we cannot use more than the number of sticks we have of that size. This is why we have the condition that $X_j = \sum_{i=1}^{k} x_{i,j} \leq a_j$ for all $1 \leq j \leq n$. The answer to the stick problem, given n and k, is $A(n, k)$.

3 Generating Results

In order to generate results, we incorporated a program written in `C++`[1] utilyzing the concept of dynamic programming. That is, we were able to break the problem down into smaller subproblems for which we could generate and store data on the hard disk for use and reuse.

Due to the large number of solutions $(a_1, a_2, \ldots, a_n)$ and $(X_1, X_2, \ldots, X_n)$ to check for each s, especially with larger n and s, a one hundred percent brute force approach was not realistic. Therefore, we had to make some simplifications.

3.1 Some Simplifications

First of all, we noticed that s must be greater than or equal to k. That is, if we have less than k sticks, there is no possible way to make k partitions of some number t. Thus, when we begin our search for $A(n, k)$ we only need to start with $s_{\min} = k$.

[1]Download our program for free use at `http://ajbertagnolli.com/A.tar.gz`.

Next, we realized that only solutions $(a_1, a_2, \ldots, a_n)$ with $a_i < k$ need to be checked. In other words, if we have at least k of any one size stick a_i, then we are certain to have k partitions already.

Finally, and perhaps most importantly is obtaining the maximum of t. Without a maximum of t, we would not know when to stop checking a particular solution $(a_1, a_2, \ldots, a_n)$ for a corresponding set of equations.

Adding the system of equations in section (2.4) gives us

$$X_1 + 2X_2 + \cdots + nX_n = kt,$$

where $X_j = \sum_{i=1}^{k} x_{i,j} \leq a_j$ for all $1 \leq j \leq n$. We note that given n, k, and s, kt will be maximized if $a_n = a_{n-1} = a_{n-2} = \cdots = a_{n-c+1} = k-1$, and $a_{n-c} = s - c(k-1)$, where $c = \left\lfloor \frac{s}{k-1} \right\rfloor$ and $\lfloor z \rfloor$ is the "floor of z". That is, our s sticks are comprised of the maximum number $(k-1)$ of the largest sticks. Therefore, we have the inequality

$$\begin{aligned} kt &= (n-c)X_{n-c} + (n-c+1)X_{n-c+1} + \cdots + nX_n \\ &\leq (n-c)a_{n-c} + (n-c+1)a_{n-c+1} + \cdots + na_n \\ &= (n-c)(s-c(k-1)) + (k-1) \sum_{i=n-c+1}^{n} i \\ &= \frac{1}{2}(c(c+1)(k-1) + 2s(n-c)). \end{aligned}$$

Dividing both sides by k gives us the inequality

$$t \leq \frac{1}{2k}(c(c+1)(k-1) + 2s(n-c)).$$

Thus,

$$t_{\max} = \left\lfloor \frac{1}{2k}(c(c+1)(k-1) + 2s(n-c)) \right\rfloor.$$

3.2 Generating $(a_1, a_2, \ldots, a_n)$

Given n and s we produce all solutions to

$$a_1 + a_2 + \cdots + a_n = s$$

with $a_i < k$ for all $1 \leq i \leq n$ by producing each partition of s in nondecreasing order, assigning any remaining $a_i = 0$. For instance, if $n = 4$ and $s = 5$, one solution $(a_1, a_2, a_3, a_4) = (0, 0, 2, 3)$. For each such nondecreasing partition, the function `next_permutation` from the Standard Template Library (STL) of `C++` was used to produce all the permutations of that partition. Every permutation of every partition provides every possible nonnegative solution.

While we used our own algorithm to produce the partitions, there are other algorithms which may be more efficient, such as the $ZS1$ and $ZS2$ algorithms provided in [9]. The primary reason we used our own algorithm is that per our simplification described in section (3.1), we do not need any solution with an $a_i \geq k$, though $ZS1$ and $ZS2$ could be modified to include this condition.

Analytically, the total number of nonnegative integer solutions to $a_1 + a_2 + \cdots + a_n = s$ can be shown to be $\binom{n+s-1}{s}$. Though this number can get large fairly quickly, it is not as much of a limiting factor for computation time as is generating the possible solutions to the $k \times n$ matrix $[x_{i,j}]$ which fulfills the conditions set out in section (2.4).

3.3 Generating $(X_1, X_2, \ldots, X_n)$

Given n, k, and t, we need to find all possible solutions $(X_1, X_2, \ldots, X_n)$ where $X_j = \sum_{i=1}^{k} x_{i,j}$ and the conditions of section (2.4) are fulfilled. To do this, we build a database of files `[t]_[k].txt` for necessary t beginning with `[t]_1.txt` holding the ways to form 1 partition of t in the form $(X_1, X_2, \ldots, X_t)$, where $X_1 + 2X_2 + \cdots + tX_t = t$. For instance, if $t = 5$, the partition $(0, 0, 0, 2, 3)$ corresponds to the file entry $(0, 1, 1, 0, 0)$.

Let $p_1, p_2, \ldots, p_{P(n)}$ be each such solution. Then to calculate possible solutions for `[t]_2.txt`, we add 2 p_i's in every possible way with no regards for order (i.e. $p_1 + p_2$ is the same as $p_2 + p_1$). For example, two entries in the file `5_1.txt` would be

```
2 0 1 0 0
1 0 0 1 0
```

and would add to give us

```
3 0 1 1 0
```

for `5_2.txt`. That is, $(3, 0, 1, 1, 0)$ is one way to form 2 partitions of 5. In other words, if we have 3 sticks of size 1, 1 stick of size 3, and 1 stick of size 4, we can form 2 partitions of 5 out of the sticks. In this way, we build a file `[t]_2.txt` by taking two lines of `[t]_1.txt` (with repetition) in every possible way and adding them together.

Next, we take each line from `[t]_2.txt` and add to each line in `[t]_1.txt` in every possible way to obtain `[t]_3.txt`. We continue this process until the desired `[t]_[k].txt`.

Analytically, the number of solutions `[t]_[k].txt` produced by choosing (with replacement) k partitions of t out of $P(t)$ total partitions is equivalent to the number of nonnegative solutions to $y_1 + y_2 + \cdots + y_{P(t)} = k$, which is $\binom{P(t)+k-1}{k}$. But some of these combinations add to the same result, so our method of producing `[t]_[k].txt` produces a lot of duplicate lines which are removed before storing or using. Being able to predict and avoid when and how these duplicates show up would drastically improve computation time. As of now, the author is unable to do so and relies on a simple `UNIX` command to clear any duplicate entries in the file. A rough estimation on several files resulted in up to 50–90% reduction in file size after removal of duplicate lines. For example, the total number of combinations with replacement and disregard for order for `5_5.txt` would be $\binom{P(5)+5-1}{5} = 462$, but after removing the duplicate sums we have only 266 unique solutions. Nevertheless, we are able to obtain all possible solutions for $X_1 + 2X_2 + \cdots + tX_t = kt$ which fulfill the necessary conditions.

With these solutions, we can easily find every possible $(X_1, X_2, \ldots, X_n)$ given n, k, and t. We notice that if $t \leq n$, then all possible solutions are the lines in `[t]_[k].txt`, setting the remaining $X_{t+1} = X_{t+2} = \cdots = X_n = 0$. Otherwise, if $t > n$, we need the entries in `[t]_[k].txt` such that $X_{n+1} = X_{n+2} = \cdots = X_t = 0$. That is, the last $(n - t)$ entries must be 0. For our code, when checking each solution $(a_1, a_2, \ldots, a_n)$ to a given s, we can increment t from 1 to $t_{\max}$ and check these files to see if for that particular $\bar{a}$ there exists a solution $\bar{X}$ such that $X_i \leq a_i$ for all $1 \leq i \leq n$. If such a solution does exist, then it means we can form k partitions of t out of that particular configuration $\bar{a}$ of sticks.

3.4 Pseudocode

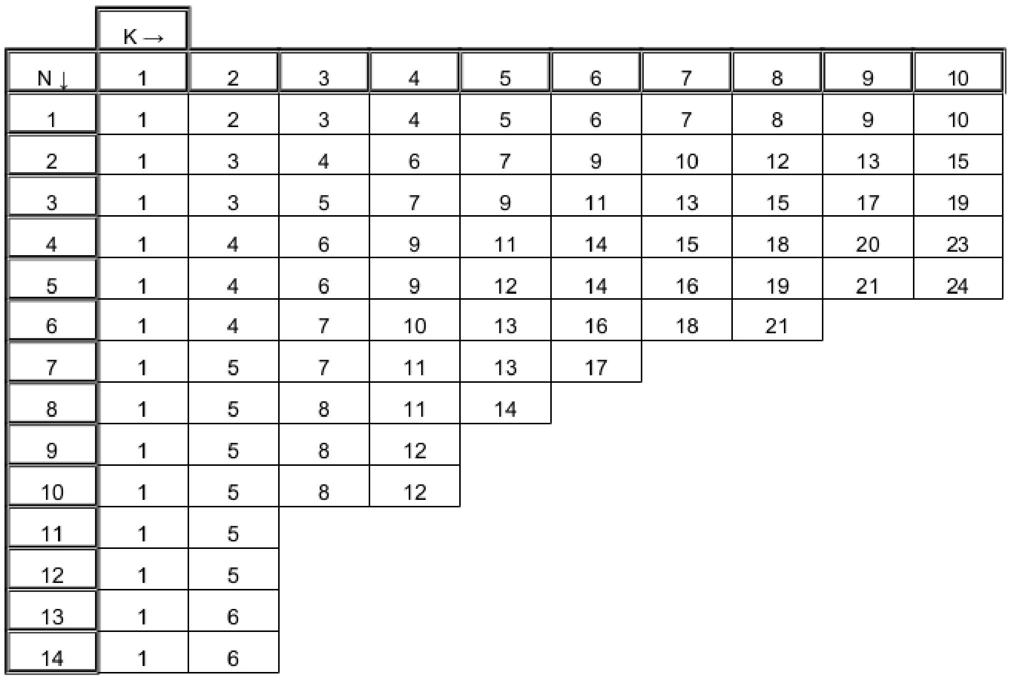
We provide the following pseudocode which gives a rough outline of how our program works. For detailed study, download the code mentioned at the beginning of section (3).

```
INPUT nonnegative integers $n, k$
OUTPUT solution $A(n, k)$ to stick problem
for $s = k$ to $\infty$
    set $c = \lfloor s/(k-1) \rfloor$
    set $t_{\max} = \lfloor (c(c+1)(k-1) + 2s(n-c))/(2k) \rfloor$
    for each solution $(a_1, a_2, \ldots, a_n)$
        set $flag = true$
        for $t = 1$ to $t_{\max}$
            for each solution $(X_1, X_2, \ldots, X_n)$
                set $flag = true$
                for $j = 1$ to $n$
                    if $X_j > a_j$
                        set $flag = false$
                        break
                    end if
                end for
                if $flag == true$
                    break
                end if
            end for
            if $flag == true$
                break
            end if
        end for
        if $flag == false$
            break
        end if
    end for
    if $flag == true$
        return $s$
    end if
end for
```

4 Results

4.1 Solutions $A(n, k)$

The following solutions to the stick problem were produced using our program.



	K→									
N↓	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	1	3	4	6	7	9	10	12	13	15
3	1	3	5	7	9	11	13	15	17	19
4	1	4	6	9	11	14	15	18	20	23
5	1	4	6	9	12	14	16	19	21	24
6	1	4	7	10	13	16	18	21		
7	1	5	7	11	13	17				
8	1	5	8	11	14					
9	1	5	8	12						
10	1	5	8	12						
11	1	5								
12	1	5								
13	1	6								
14	1	6								

Table 1: Solutions $A(n, k)$ to the stick problem.

4.2 Counterexamples

Our program was also designed to produce a counterexample for $A(n, k) - 1$. For example, the counterexample found for $A(6, 4) = 10$ is the case where we have 3 sticks of size 3, 1 of size 4, 3 of size 5, and 2 of size 6, for a total of 9 sticks. Thus if we have these 9 sticks, there is no way we can form 4 partitions (a perfect square) of the same number. This was designed as a fail-safe for the minimum of $A(n, k)$. While the algorithm itself is purely brute force and therefore in theory should not give a false result, it is impossible to prove that there was not some error in coding. But having a counterexample gives us a confident lower bound. For instance, if someone were to claim that $A(6, 4) = 9$, we could simply give the aforementioned counterexample to prove that $A(6, 4) > 9$. Some counterexamples are listed in Table 2.

n	k	$A(n,k) - 1$	$(a_1, a_2, \ldots, a_n)$
2	3	3	(1, 2)
2	4	5	(3, 2)
2	5	6	(3, 3)
3	3	4	(0, 2, 2)
3	4	6	(0, 3, 3)
3	5	8	(0, 4, 4)
4	3	5	(0, 1, 2, 2)
4	4	8	(0, 3, 3, 2)
4	5	10	(0, 4, 4, 2)
5	3	5	(0, 1, 2, 2, 0)
5	4	8	(0, 3, 3, 2, 0)
5	5	11	(0, 1, 4, 3, 3)
6	3	6	(0, 2, 0, 0, 2, 2)
6	4	9	(0, 0, 3, 1, 3, 2)
6	5	12	(0, 4, 0, 2, 4, 2)
7	3	6	(0, 2, 0, 0, 2, 2, 0)
7	4	10	(0, 0, 3, 0, 3, 2, 2)
7	5	12	(0, 4, 0, 2, 4, 2, 0)
8	3	7	(0, 0, 1, 0, 2, 1, 2, 1)
8	4	10	(0, 0, 3, 0, 3, 2, 2, 0)

Table 2: Counterexamples for $A(n, k) - 1$.

4.3 On-Line Encyclopedia of Integer Sequences (OEIS)

The On-Line Encyclopedia of Integer Sequences (OEIS)[2] is a freely accessible and searchable online database of integer sequences maintained by the non-profit The OEIS Foundation. For an integer sequence of interest, the database lists areas where the sequence has been found, comments on the sequence, references, formulas, and other relevant information pertaining to the sequence. Often to interpret results it can be helpful to search the database for a generated sequence. For example, if we come across the sequence 1, 1, 3, 3, 5, 5, 7, 7, we can search OEIS for it which takes us to the page for sequence number A109613 "Odd numbers repeated." Along with an expanded list of the sequence, it gives us a lot of other information, such as where the sequence has appeared in different areas of mathematics. For instance, the comments section tells us it is the number of rounds in a round-robin tournament with n competitors. So, naturally we were drawn to search OEIS for some of the integer sequences that appear in the rows and columns of TABLE 1.

The columns of TABLE 1 do not return any integer sequences, with the exception of column 1 (a trivial case) and column 3. The second row returns the sequence A032766, or "Numbers that are congruent to 0 or 1 mod 3." The third row appears to be the odd numbers which will of course return a sequence. It becomes more interesting for $n \geq 4$. The fourth row returns the sequence A189756, or "$n + \lfloor ns/r \rfloor + \lfloor nt/r \rfloor$; $r = 1, s = \sin(1), t = \cos(1)$". Finding this to be interesting, we used the program to calculate $A(4, 11)$ to see if the pattern continued to follow the sequence. Unfortunately, it breaks the given sequence at $A(4, 11) = 27$ as the next entry in A189756 is 28. We run into a similar problem with row 6. The sequence A184927 matches row 6 for 4, 7, 10, 13, 16, and 18, but $A(6, 8) = 21$ breaks the sequence. Row 5 matches the sequence A190304 up to $A(5, 10) = 24$ (as far as our program has calculated), but due to the sudden departure from similar sequences for larger k in rows 4 and 6, we suspect 5 will do the same for higher k.

5 Conclusion

While we have produced some results for specific values of n and k, there is still a lot of work to be done on the stick problem. There are clear patterns in TABLE 1, but can we prove mathematically that those patterns continue indefinitely? Some of the patterns break suddenly. This is right on track for the general complicated nature of partitions. For instance, even the number of partitions seem to follow the primes at first, then they stray. That is, the values of $P(n)$ from $n = 2$ to $n = 6$ are 2, 3, 5, 7, and 11, respectively, but then $P(7) = 15$. Since every aspect of the stick problem concerns partitions, then we cannot simply take the first few integers in the sequences at face value. So is there some formula or simple algorithm which can give us the solution for any given n and k?

Furthermore, even the computer-generated solutions take a long time to produce. Are there more efficient algorithms we can use to generate the solutions? It is this type of collaborative exploration that often leads to new mathematics and coding methods. In fact, similar (though far from equivalent) problems are studied today in coding theory and algorithm design problems such as the McNugget Problem [6] and the Knapsack Problem [5]. These are problems which on the surface seem silly, but underneath they test the mathematician and programmer to always search for better, more efficient, and more elegant methods. Once developed, such methods can be applied to real world problems wherever possible. And on that note we close with a quotation from Gottfried Leibniz, creator of infinitesimal calculus [2]:

> *"Even in the games of children there are things to interest the greatest mathematician."*

[2] `http://oeis.org`

Acknowledgments Special thanks to Alabama Supercomputer Authority for providing many hours of computation time, and to Doctors Enoch Lee, Luis Cueva-Parra, and Rhodes Peele for their advice, support, encouragement, and enthusiasm.

References

[1] G. Waldo Dunnington. *Carl Friedrich Gauss: Titan of Science*. The Mathematical Association of America, 1955.

[2] Gottfried W. Leibniz. Discours touchant la méthode de la certitude et de lart dinventer pour finir les disputes et pour faire en peu de temps de grands progrès, 1688.

[3] Eric W. Weisstein. “Collatz Problem.” From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/CollatzProblem.html`. [Online; accessed 29-April-2013].

[4] Eric W. Weisstein. “Goldbach Conjecture.” From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/GoldbachConjecture.html`. [Online; accessed 29-April-2013].

[5] Eric W. Weisstein. “Knapsack Problem.” From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/KnapsackProblem.html`. [Online; accessed 29-April-2013].

[6] Eric W. Weisstein. “McNugget Number.” From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/McNuggetNumber.html`. [Online; accessed 29-April-2013].

[7] Eric W. Weisstein. “Partition.” From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/Partition.html`. [Online; accessed 29-April-2013].

[8] Eric W. Weisstein. “Partition Function P.” From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/PartitionFunctionP.html`. [Online; accessed 29-April-2013].

[9] A. Zoghbi and I. Stojmenović. Fast algorithms for generating integer partitions. `http://www.site.uottawa.ca/~ivan/F49-int-part.pdf`, 1998. [Online; accessed 1-May-2013].