Representing Numbers Using Fibonacci Variants

Stephen K. Lucas
Department of Mathematics & Statistics
James Madison University
Harrisonburg, VA 22807

September 2014

1 Introduction

The Fibonacci numbers and their variants are among the most popular sequences in recreational mathematics, and even have a journal (*The Fibonacci Quarterly*) dedicated to them. Many books have been written about them, including Dunlap [4] and Vajda [17]. Fibonacci numbers are defined by the recurrence relation $f_n = f_{n-1} + f_{n-2}$ with initial conditions $f_0 = 0$ and $f_1 = 1$, and have the closed form representation $f_k = \left(\phi^k - (1-\phi)^k\right)/\sqrt{5}$, where $\phi = (1+\sqrt{5})/2$ is the well known golden ratio, also with its own book by Livio [13].

One property of Fibonacci numbers is that every natural number can be uniquely represented by a sum of distinct non-consecutive Fibonacci numbers starting from $f_2 = 1$, since $f_1 = f_2 = 1$ is a repeat, and $f_0 = 0$ won't contribute to a sum. Consecutive Fibonacci numbers can, of course, be replaced by the next largest. This was first discovered by Eduourd Zeckendorf in 1939, but only published by him in 1972 [19]. The first publication of the result was by Lekkerkerker [12] in 1952. As a result, representing a natural number in this form is known as its Zeckendorf representation, and is easily found using a greedy algorithm: given a number, subtract the largest Fibonacci number less than or equal to it, and repeat until the entire number is used up. For example, consider 825. The largest Fibonacci number less than 825 is $f_{15} = 610$, and $825 - 610 = 215$. Then $f_{12} = 144$ and $215 - 144 = 71$, $f_{10} = 55$ and $71 - 55 = 16$, $f_7 = 13$ and $16 - 13 = 3$, and finally $f_4 = 3$. Therefore, $825 = f_{15} + f_{12} + f_{10} + f_7 + f_4$, which can be represented as $(10010100100100)_Z$, where the Z indicates Zeckendorf representation, and digits from right to left indicate whether or not f_2, f_3 and so on are included in the sum for the number.

The lack of consecutive Fibonacci numbers in Zeckendorf representations of natural numbers means a pair of ones can be used to separate numbers in a list. This means different numbers in a list can be represented using a different number of digits, known as a variable length encoding. Traditionally, a fixed number of digits (in a given base) are used to represent every number in a list, which may waste space. The first part of this paper compares the efficiency of representing numbers using Zeckendorf form versus

traditional binary with a fixed number of digits, and shows when Zeckendorf form is to be preferred. We shall also see what happens when variants of Zeckendorf form are used.

Not only can we represent natural numbers as sums of Fibonacci numbers, we can also do arithmetic with them directly in Zeckendorf form. We include a survey of past approaches to Zeckendorf representation arithmetic, as well as some improvements.

2 Zeckendorf Proofs

Proving that the Zeckendorf representation exists and is unique for every natural number is straightforward enough to include here. Existence is proven by induction, and begins with $1 = f_2$, $2 = f_3$, and $3 = f_4$. If we assume every natural number up to n has a Zeckendorf representation, consider $n + 1$. If it is a Fibonacci number, we are done. Otherwise, there is some j such that $f_j < n + 1 < f_{j+1}$. Since $n + 1 - f_j < n$, it has a Zeckendorf representation, and additionally since $n+1-f_j < f_{j+1}-f_j = f_{j-1}$, it doesn't contain f_j or f_{j-1}. Thus the Zeckendorf representation of $n+1$ is that for $n+1-f_j$ with f_j included, and each Fibonacci number in the representation occurs at most once and non-consecutively. By induction, we are done.

Before proving uniqueness, we need the result that any sum of distinct non-consecutive Fibonacci numbers, whose largest is f_n, is strictly less than f_{n+1}. Again, we can use induction, and since $f_2 = 1 < f_3 = 2$, $f_3 = 2 < f_4 = 3$ and $f_2 + f_4 = 4 < f_5 = 5$, it is initially true. Now assume that any sum of distinct non-consecutive Fibonacci numbers whose largest is f_n is strictly less than f_{n+1}. Then a sum with largest number f_{n+1} can be split into f_{n+1} and a sum with largest Fibonacci number at most f_{n-1}, which is strictly less than f_n by the assumption. The combination is thus strictly less than $f_n + f_{n+1} = f_{n+2}$, and we are done.

And now to uniqueness. Assume that there are two different sets of non-consecutive Fibonacci numbers that have the same sum, A and B. If there are any Fibonacci numbers common to both collections, remove them from both by set difference, and let C and D be the differences $C = A - B$ and $D = B - A$. Since the same Fibonacci numbers are being removed from both A and B, the Fibonacci numbers in the smaller sets C and D still have the same sum, and have no common numbers. Let f_c and f_d be the largest elements of C and D respectively, and since there are no common numbers, $f_c \neq f_d$. In addition, without loss of generality, assume $f_c > f_d$. But by our previous result, the sum of D is strictly less than f_{d+1} and so must also be strictly less than f_c. But the sum of C is at least f_c. The only way this is possible is for C and D to in fact be empty sets that sum to zero. But this means that the sets A and B must be the same, and we do have a unique representation.

3 Efficiency of Number Representations

A number's Zeckendorf representation, being a string of zeros and ones, looks a lot like a base two representation, but shouldn't be confused with it. For example, the base two representation of 825, which equals $512 + 256 + 32 + 16 + 8 + 1$, is $(1100111001)_2$. This

requires fewer digits than its Zeckendorf representation, and so is more efficient in terms of digits required. In fact, the base two representation of a natural number will always be shorter than its Zeckendorf representation, because the larger the base, the smaller the number of digits required. Zeckendorf form does not formally have a base, but since $\phi \approx 1.618$ and $1 - \phi \approx -0.618$, f_k is the closest natural number to $\phi^k/\sqrt{5}$, so the ratio between Fibonacci numbers approaches ϕ. Thus Zeckendorf representation roughly has the base $\phi \approx 1.618 < 2$, and so is less efficient than binary.

3.1 Lists of Natural Numbers

While Zeckendorf representation is less efficient (in terms of number of digits required) than binary, its big advantage is that it cannot contain a pair of consecutive ones. So instead of using a fixed number of digits to represent individual natural numbers in a list, as many digits as are necessary can be used for each number, with a pair of ones separating consecutive numbers. We can do even better if we reverse the order of the digits, with least to most significant digits from left to right. Using the reverse of the standard way we write numbers, the rightmost digit of a number's Zeckendorf representation will always be a one. So when a pair of ones is used to separate numbers, the first one is part of the first number, and only the second one is the separator between numbers. For example, the stream of digits "10010101110001011011" can be separated out into "10010101," "1000101," and "01," or $f_2 + f_5 + f_7 + f_9$, $f_2 + f_6 + f_8$, and f_3, or 53, 30, and 2.

This reversed way of representing numbers in a list is sometimes known as "Fibonacci coding," and is particularly useful when there is no prior knowledge of the range of the list of numbers. In this case, using a base two representation with a fixed number of binary digits (bits) is problematic. We might overestimate how big the numbers will get and waste space in base two, or underestimate and have numbers we can't represent.

Even when we know exactly what range of numbers to expect in a list, there are occasions when Fibonacci coding is superior to traditional base two. For example, consider a list of numbers known to range from one to a million, with each number being equally likely. A base two representation of numbers in this range will require at least twenty bits per number. Using Fibonacci coding, the same distribution of numbers require on average 27.8 bits per number, so more space and thus less efficient. But if the numbers from one to ten are equally likely (with probability 9999/100,000) and one million occurs with probability only one in ten thousand, then the base two list stays at twenty bits per number, but the Fibonacci coding reduces to only 4.6 bits per number on average. While this may be an extreme case, the saving is substantial, particularly when smaller numbers are more likely. As another example, consider nonnegative integers chosen with Poisson distribution ($P(X = k) = \lambda^k e^{-\lambda}/k!$) and $\lambda = 4$. Practically, output is integers from one to thirty one, and Fibonacci coding requires 4.6 bits per number. Binary would require five bits per number. In these last two cases, lists of numbers are more efficiently written using Fibonacci coding.

3.2 Arbitrary Reals and Continued Fractions

There is one area where Fibonacci coding has not previously been applied and it is particularly appropriate – the representation of arbitrary reals as continued fractions. A continued fraction representation of a real is essentially a list of natural numbers. The list has no a priori upper bound, and smaller natural numbers occur more often than larger ones.

As a reminder, continued fractions are closely related to the algorithm for finding the greatest common divisor of two natural numbers using integer division. For example, given 236 and 24 we can successively find $236 = 9 \times 24 + 20$, then $24 = 1 \times 20 + 4$, and $20 = 5 \times 4 + 0$, which tells us that $\gcd(236, 24) = 4$. But we can also use these steps to write

$$\frac{236}{24} = 9 + \frac{20}{24} = 9 + \frac{1}{24/20} = 9 + \frac{1}{1 + \frac{4}{20}} = 9 + \frac{1}{1 + \frac{1}{20/4}} = 9 + \frac{1}{1 + \frac{1}{5}}.$$

Any fraction can be rewritten in this form of fractions within fractions (hence the name continued fractions) for which all the fractions have one as their numerator. A common more compact notation rewrites this as $236/24 = [9; 1, 5]$, and in general, a simple continued fraction for a (positive) fraction is

$$\frac{p}{q} = b_0 + \cfrac{1}{b_1 + \cfrac{1}{b_2 + \cfrac{\vdots}{b_{n-1} + \cfrac{1}{b_n}}}} \equiv [b_0; b_1, b_2, \ldots, b_n],$$

where b_0 is an integer, and the b_i's for $i > 0$ are natural numbers. The compact notation on the right is much more convenient when the number of terms in the continued fraction representation becomes large. The b_i's are traditionally called partial quotients. Let us use the notation $\lfloor x \rfloor$, stated "the floor of x," for the largest integer less than or equal to x. Then the algorithm for finding partial quotients (adapted from integer division) is: given some real number x, set $x_0 = x$ and $b_0 = \lfloor x_0 \rfloor$, then

$$x_i = \frac{1}{x_{i-1} - b_{i-1}} \quad \text{and} \quad b_i = \lfloor x_i \rfloor \quad \text{for} \quad i = 1, 2, \ldots.$$

If x is rational, eventually some x_i will be an integer and the continued fraction terminates. If x is irrational, the sequence of partial quotients goes forever. For our example with $x = 236/24$,

$$x_0 = \frac{236}{24}, \qquad b_0 = \left\lfloor \frac{236}{24} \right\rfloor = 9,$$

$$x_1 = \frac{1}{236/24 - 9} = \frac{1}{5/6} = 6/5, \qquad b_1 = \left\lfloor \frac{6}{5} \right\rfloor = 1,$$

$$x_2 = \frac{1}{6/5 - 1} = \frac{1}{1/5} = 5, \qquad b_3 = \lfloor 5 \rfloor = 5,$$

k	Prob.	k	Prob.
1	0.415037	10	0.011973
2	0.169925	100	1.41434×10^{-4}
3	0.093109	1000	1.43981×10^{-6}
4	0.058894	10,000	1.44241×10^{-8}
5	0.040642		
6	0.029747	> 10	1.25531×10^{-1}
7	0.022720	> 100	1.42139×10^{-2}
8	0.017922	> 1000	1.44053×10^{-3}
9	0.014500	$> 10{,}000$	1.44248×10^{-4}

Table 1: Probabilities of various partial quotients occurring in a random continued fraction

and we are done.

Continued fractions have many elegant features, and a straightforward introduction is Olds [15]. Many introductory texts on number theory, including the classic Hardy and Wright [10], includes a chapter on continued fractions, but the feature of relevance here is the Gauss-Kuzmin theorem, that tells us that for almost all irrationals between zero and one, as $n \to \infty$ the probability that the nth partial quotient is k is

$$\lim_{n\to\infty} P(k_n = k) = -\log_2\left(1 - \frac{1}{(k+1)^2}\right).$$

Khinchin [11] gives as particularly clear derivation of this result. Its importance from our perspective is that arbitrarily large partial quotients are possible, but increasingly unlikely, in a continued fraction representation. Table 1 shows the probabilities of the first few natural numbers occurring as any given partial quotient in the continued fraction representation of an arbitrary irrational, as well as the probabilities that large partial quotients can occur. Thus Fibonacci coding is an ideal choice for representing continued fraction partial quotients for arbitrary irrationals.

For example, consider ln(2), whose first twenty thousand partial quotients are summarized in table 2. Since there are only 20,000 partial quotients to work with, some of the probabilities aren't exactly equal to those in the theoretical distribution, but they are remarkably close. The largest partial quotient happens to be 963,664. If we knew this beforehand, we could encode this continued fraction using binary with twenty bits per number. The Fibonacci coding of these partial quotients requires on average the shockingly low 3.74 bits per number. Not only do we not need to know the largest number in the sequence beforehand, but we have an extraordinarily compact way of representing the sequence. For comparison purposes, a version of Lochs' theorem [14, 18] states that if m is the number of terms in a number's continued fraction expansion and p is the number of correct digits in the continued fraction when converted to binary representation, then almost always

$$\lim_{n\to\infty} \frac{m}{n} = \frac{6(\ln 2)^2}{\pi^2} \approx 0.292.$$

Assuming a long enough continued fraction, this tells us that a number with m partial

k	Prob.	k	Prob.
1	0.4152	10	0.01285
2	0.1668	100	2.5×10^{-4}
3	0.09405	1000	0
4	0.0577	10,000	0
5	0.0397		
6	0.03065	> 10	1.284×10^{-1}
7	0.0222	> 100	1.45×10^{-2}
8	0.0179	> 1000	1.65×10^{-3}
9	0.0145	$> 10{,}000$	1.5×10^{-4}

Table 2: Probabilities of the first 20 000 partial quotients occurring in ln(2)

quotients will be accurate to about $3.42m$ bits in base two. Since $3.74 > 3.42$, the binary representation of ln(2) is slightly more efficient than the Fibonacci coding of its continued fraction to about sixty-eight thousand binary digits, or about twenty thousand decimal digits. But we lose all the additional information given by the continued fraction partial quotients.

As another example, consider the first twenty thousand partial quotients of π. In this case the largest partial quotient is 74,174, and the Fibonacci coding requires 3.71 bits per number. Slightly better than in the ln(2) case, but still slightly worse than the pure binary representation.

In conclusion, lists of natural numbers where smaller numbers occur more often than larger ones are more compactly represented using Fibonacci coding instead of traditional binary representations. This is particularly striking when representing lists of partial quotients for arbitrary irrationals, where in addition to varying magnitude there is no prescribed upper bound. Unfortunately, the binary representation is still slightly more efficient than the Fibonacci coded continued fraction representation, if all you are interested in is a high precision representation.

4 Generalizing Fibonacci Coding

We have already seen how the effective base of Zeckendorf representation is $\phi \approx 1.618 < 2$, so more digits are required than in traditional binary. But there is no reason why we have to be limited to traditional Fibonacci numbers. The *Fibonacci Quarterly* is filled with many variants on the Fibonacci sequence. The *tribonacci* sequence is defined by $t_n = t_{n-1} + t_{n-2} + t_{n-3}$ with $t_{-1} = t_0 = 0$ and $t_1 = 1$, and continues $1, 2, 4, 7, 13, 24, 44, \ldots$. The *tetranacci* sequence is defined by $u_n = u_{n-1} + u_{n-2} + u_{n-3} + u_{n-4}$ with $u_{-2} = u_{-1} = u_0 = 0$ and $u_1 = 1$, and continues $1, 2, 4, 8, 15, 29, 56, \ldots$. The earliest description of these series dates to 1963 in Feinberg [5]. As with Fibonacci numbers, every number can be uniquely represented as sums of tribonacci or tetranacci numbers. Three consecutive ones cannot appear in a tribonacci representation, and four consecutive ones cannot appear in a tetranacci representation. These are special cases of sequences of numbers that can be used to represent arbitrary integers, as described in Frankel [7]. Our interest in these

k	(a)	(b)	(c)	(d)	(e)
2	27.82	4.60	4.57	3.74	3.71
3	23.34	5.00	4.96	4.35	4.33
4	22.86	5.90	5.85	5.28	5.27
5	23.40	6.90	6.85	6.26	6.25

Table 3: Bits on average for k-bonacci representations for (a) equally spaced, one to a million, (b) uneven distribution one to ten and one million, (c) Poisson with $\lambda = 4$, (d) $\ln 2$ partial quotients, and (e) π partial quotients.

representations is that the tribonacci numbers grow like x^n where x is the largest root of $x^3 - x^2 - x - 1 = 0$, or about 1.8393, and tetranacci numbers grow like x^n where x is the largest root of $x^4 - x^3 - x^2 - x - 1 = 0$, or about 1.9276. These roots are closer to two than the golden ratio, and so the number of digits required to represent natural numbers using these sequences will get closer to how many are required using binary digits, keeping the advantage that variable length coding is possible. The disadvantage is that the number of repeated ones needed to separate numbers increases.

There is no reason why we should stop with tetranacci numbers, although the naming conventions become cumbersome. Define a sequence of k-bonacci numbers by

$$u_n = \sum_{i=1}^{n} u_{n-i}, \quad \text{with } u_1 = 1 \text{ and } u_i = 0 \text{ for } i < 0.$$

Then Fibonacci, tribonacci and tetranacci numbers are 2-bonacci, 3-bonacci and 4-bonacci numbers, respectively. The 5-, 6- and 7-bonacci numbers grow like 1.9659^n, 1.9836^n and 1.9920^n respectively, and the effective bases are approaching 2. Since $k - 1$ digits are used to separate different numbers in variable length coding, there will be some optimum k-bonacci sequence to minimize the length of an encoding of a sequence of numbers, depending on their distribution.

Let us return to the examples where we compared Fibonacci coding to binary. Table 3 lists how many bits on average are required with k-bonacci coding with various values of k in the various examples. For uniformly distributed numbers, 4-bonacci numbers are the best sequence, but are still inferior to binary when we happen to know the upper bound of a million. In every other case, Fibonacci coding is superior. Alas, the gains made by shortening the length of the sequence of digits required by increasing the effective base has been lost to the additional digits required to separate individual numbers.

5 Arithmetic

Not only can natural numbers be represented as sums of Fibonacci numbers, but arithmetic can easily be done on them in this form. Here we return to Zeckendorf form with most significant digits on the left. After numbers are combined, the resulting sum of Fibonacci numbers will usually not be in Zeckendorf form, because it will include pairs of successive ones or numbers greater than one. Luckily, a sum of Fibonacci numbers can be easily returned to Zeckendorf form by a combination of two rules:

- The *pair rule*: since $f_n = f_{n-1} + f_{n-2}$ or $f_n - f_{n-1} - f_{n-2} = 0$, subtracting one from successive digits adds one to the digit immediately to their left, which can be represented by the transformation $(\ldots(+1)(-1)(-1)\ldots)_Z$.

- The *two rule*: subtracting $f_{n+1} = f_n + f_{n-1}$ from $f_n = f_{n-1} + f_{n-2}$ we get $f_{n+1} + f_{n-2} - 2f_n = 0$. Subtracting two from a digit adds one to the digit to its left (like an ordinary base two carry), and additionally adds one to the digit two to the right. This can be represented by the transformation $(\ldots(+1)(-2)(0)(+1)\ldots)$, where the (0) indicates no change to the digit. It is this nonstandard carry that makes arithmetic with Zeckendorf representation more entertaining.

The nonstandard carry with the two rule means we need to be more careful near the right edge of the number. Since $2f_1 = f_2$ and $2f_2 = f_3 + f_1$, the special cases of the two rule at the right edge are $(\ldots(+1)(-2))_Z$ and $(\ldots(+1)(-2)(+1))_Z$.

5.1 Addition

To add numbers in Zeckendorf form, we simply add the digits, then apply the pair and two rules as necessary to return to Zeckendorf form. For example, consider

$$(101001001)_F + (100101001)_F = (201102002)_F.$$

To return to Zeckendorf form, it is easiest visualize using a checkerboard representation, where boxes represent successive Fibonacci numbers, and counters in a box count how many multiples of that Fibonacci number are needed. The pair and two rules govern how counters can be moved. Figure 1 shows one way of returning $(201102002)_F$ to Zeckendorf form. At the top we represent $(201102002)_F$. Then each successive row of the figure shows how applications of the pair rule (three times), the two rule (once) and the pair rule one more time leads to a distribution of counters in Zeckendorf form. The sum is thus $(10000010101)_Z$.

In this example, we did not apply a systematic approach when returning to Zeckendorf form. Early discussions of addition by Freitag and Phillips [8] in 1998 and Fenwick [6] in 2003 did not suggest a systematic approach. In 2002, Tee [16] suggested a recursive approach that had the rather pessimistic bound of $O(n^3)$, which states that the number of applications of pair or two rules required in returning to Zeckendorf form is proportional to the cube of the number of digits n. In 2013, Ahlbach *et al.* [1] showed how returning a sum to Zeckendorf form could be achieved using exactly $3n$ applications of pair or two rules. Their algorithm uses three passes through the digits, looking at groups of four successive digits. Specifically, their first pass from left to right performs the replacements (x is any digit) $020x \to 100(x+1)$, $030x \to 110(x+1)$, $021x \to 110x$ and $012x \to 101x$, which they show eliminates any two's in the representation combined with a "clean-up" operation at the end. Their second pass from right to left performs the replacement $011 \to 100$, which eliminates the pattern 1011 from the representation, and the third pass from left to right repeats the second pass, and eliminates any remaining successive ones without additional consecutive ones.

Here, we introduce an improvement on Ahlbach *et. al* [1] that reduces the process to two passes. We start with the same first pass to remove twos. Then insert a leading

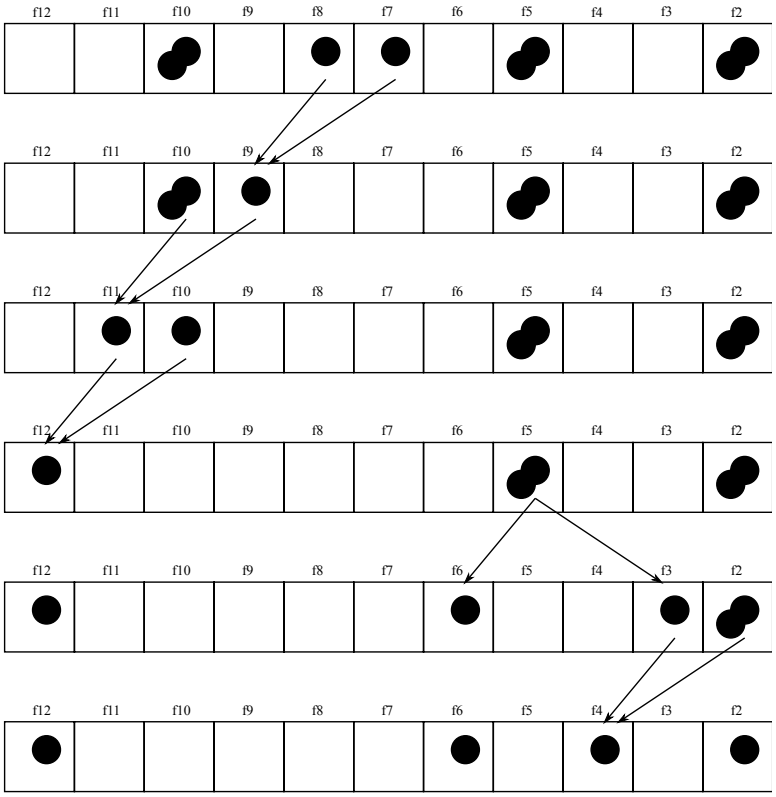

Figure 1: Using a checkerboard to visualize returning a number to Zeckendorf form.

zero, and the second pass from left to right does the replacement $(01)^k 1 \rightarrow 1(0)^{2k}$, where the notation $(01)^k$ means k copies of the pair of digits zero and one. This deals with any possible carries or long sequences of ones, and can be achieved by a single pass using a pair of pointers, one at the first 01 pair, the second moving to the right. If a pair of consecutive zeros are found, then the left pointer can be moved to match the right pointer, and we continue through the number. We cannot improve the efficiency further, due to the carry from the two rule going in both directions.

For example, let us reconsider $(201102002)_F$. Applying the first left to right pass (with an inserted leading zero) gives the sequence

$$(0201102002)_F \rightarrow (1002102002)_F \rightarrow (1011002002)_F \rightarrow (1011010012)_F \rightarrow (1011010101)_F.$$

The second pass gives the sequence $(01011010101)_F \rightarrow (10000010101)_F$ as before.

5.2 Subtraction

As with addition, subtraction can be done in a variety of ways. The simplest is a form of reallocation, as currently taught in most American elementary schools. Digit by digit, $0 - 0 = 0$, $1 - 0 = 1$ and $1 - 1 = 0$. In the $0 - 1$ case we go to the left in the digits of the first number, and find the first 1. Applying the pair rule in reverse then replaces the triple of digits 100 by 011. Repeat this with the right most of the new ones until there

is a one in the first number available for canceling. The rightmost digit is a special case, where we replace $(10)_Z$ by $(02)_Z$ before subtraction if necessary.

After using this technique, the number may not be in Zeckendorf form, but it will certainly not contain any two's, so a single pass of the new approach described for addition will reduce it to Zeckendorf form. Thus, at worst, subtraction can be done in essentially three passes. In the worst case the reallocation steps require moving up and down the entire first number (two passes at worst), then one pass to eliminate pairs of ones. For example, consider $(101000010)_Z - (010000101)_Z$. Matching digits by reallocation requires three applications of the pair rule, replacing the problem by $(011101102)_Z - (010000101)_Z = (1101001)_Z$. Eliminating pairs gives $(10001001)_Z$, the final answer.

It is worth mentioning that Fenwick [6] initially recommended this reallocation approach, then went on to a much more complicated complement approach. Tee [16] also had a slow $O(n^3)$ algorithm. Ahlbach *et al.* [1] just subtracted digits, and added an additional pass to eliminate negative digits. They then used their addition algorithm, so four passes in total. The approach here is the most efficient.

5.3 Multiplication

There are four distinct ways to be found in the literature to perform multiplication. In the same way that traditional multiplication is performed digit by digit, Freitag and Phillips [8] multiplied numbers by adding the products of Fibonacci numbers that appear in the Zeckendorf representations of the numbers. They proves the odd and even rules

$$f_m f_{2i} = \sum_{j=0}^{i-1} f_{m+2i-2-4j} \quad \text{and} \quad f_m f_{2i+1} = f_{m-2i} + \sum_{j=0}^{i-1} f_{m+2i-1-4j},$$

where $m > 2i$ and $2i+1$ respectively. They recommend converting back to Zeckendorf form after each addition to avoid arbitrarily large digits.

Tee [16] suggested using Russian Peasant multiplication, which can be algebraically written as: if y is even, then $xy = (2x)(y/2)$, else $xy = x + x(y-1) = x + (2x)((y-1)/2)$. Doubling a number in Zeckendorf form is easy, just replace every one by two, and return to Zeckendorf form. This requires three passes using the new technique. Halving is just as easy using the reversed pair rule, $(\cdots(-1)(+1)(+1)\cdots)$. From left to right, apply the reversed pair rule to any one or three (threes can accumulate from multiple additions). Twos are initially ignored, and after the pass all digits will be zero or two, apart from possibly the last pair. To halve, replace twos by ones, and the last pair of digits identify if the number was initially odd. Looking at the the special cases of halving and potentially odd,

$$(00) \to (00)\text{F}, \quad (01) \to (00)\text{T}, \quad (10) \to (01)\text{F}, \quad (11) \to (01)\text{T}, \quad (12) \to (10)\text{F},$$
$$(20) \to ((10)\text{F}, \quad (21) \to (10)\text{T}, \quad (22) \to (11)\text{F}, \quad (31) \to (11)\text{T}.$$

One more pass will be required to eliminate pairs of ones. For example, with 45, $(10010100)_F \to (01110100)_F \to (00220100)_F \to (00220011)_F$. Halving and applying

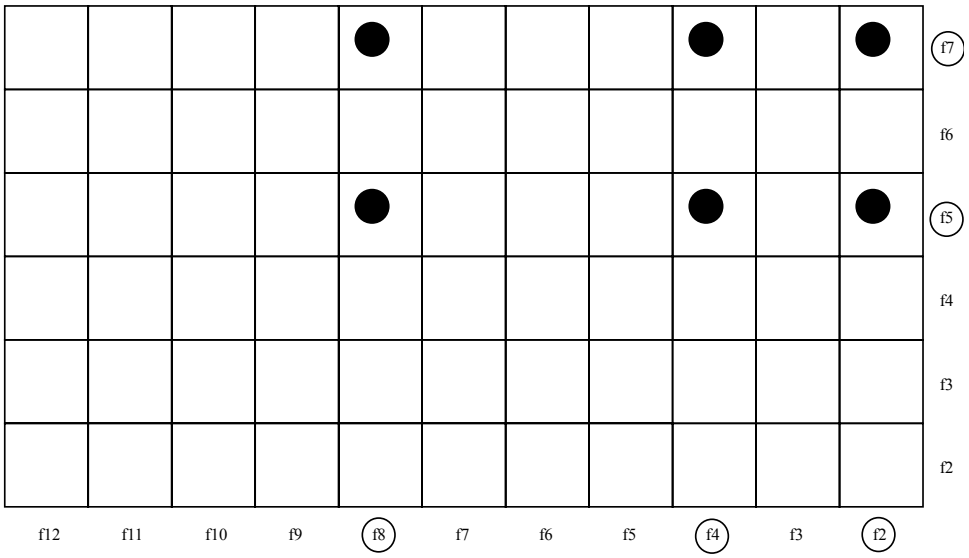

Figure 2: Multiplication using a checkerboard in Zeckendorf form, initial setup.

the (11) last pair rule, we get $(110001)_F$, with true. One last pass replaces this by $(1000001)_F = 22$, which is half of 45, which is odd.

Fenwick [6] suggested a variant of Egyptian multiplication. This version doesn't use doubling, and instead adds the previous two numbers. Unfortunately, this technique is less effective than Russian Peasant, because there are more additions than doubles.

A final technique for multiplication is motivated by John Napier's technique of addition and multiplication in binary using a checkerboard, as described in Gardner [9]. To begin, label the rows and columns with the Fibonacci numbers used in Zeckendorf form. Then, since each number is represented by a sum of Fibonacci numbers, by the distributive rule their product can be laid out as counters where each column is associated with the first number, each row with the second. For example, consider 25×18, or $(1000101)_Z \times (101000)_Z$. Figure 2 shows how this would be initially laid out. I have circled the relevant Fibonacci numbers used to represent the first and second numbers.

We now want to return the number to Zeckendorf form, where we can now work down columns as well as across rows. One systematic approach is to use the pair rule in reverse in the vertical direction to remove every counter in the top row, then use the addition approach to return the rows that have changed back to Zeckendorf form. We repeat this until a single row remains at the bottom. As with addition, some care needs to be taken with the final step. Figure 3 shows the intermediate steps for 25×18. While it may not look terribly obvious as a figure, physically moving counters around a checkerboard and applying the pair and two rules is an easy process to follow. It is also very easy to implement on a computer using a two dimensional array. There is a similarity here to the Freitag and Phillips [8] approach, but it is not immediately obvious whether the two dimensional array cuts down on the amount of work required to find the solution.

We note that Ahlbach *et al.* [1] avoid multiplication by converting to binary, multiplying in binary, then converting back to Zeckendorf form.

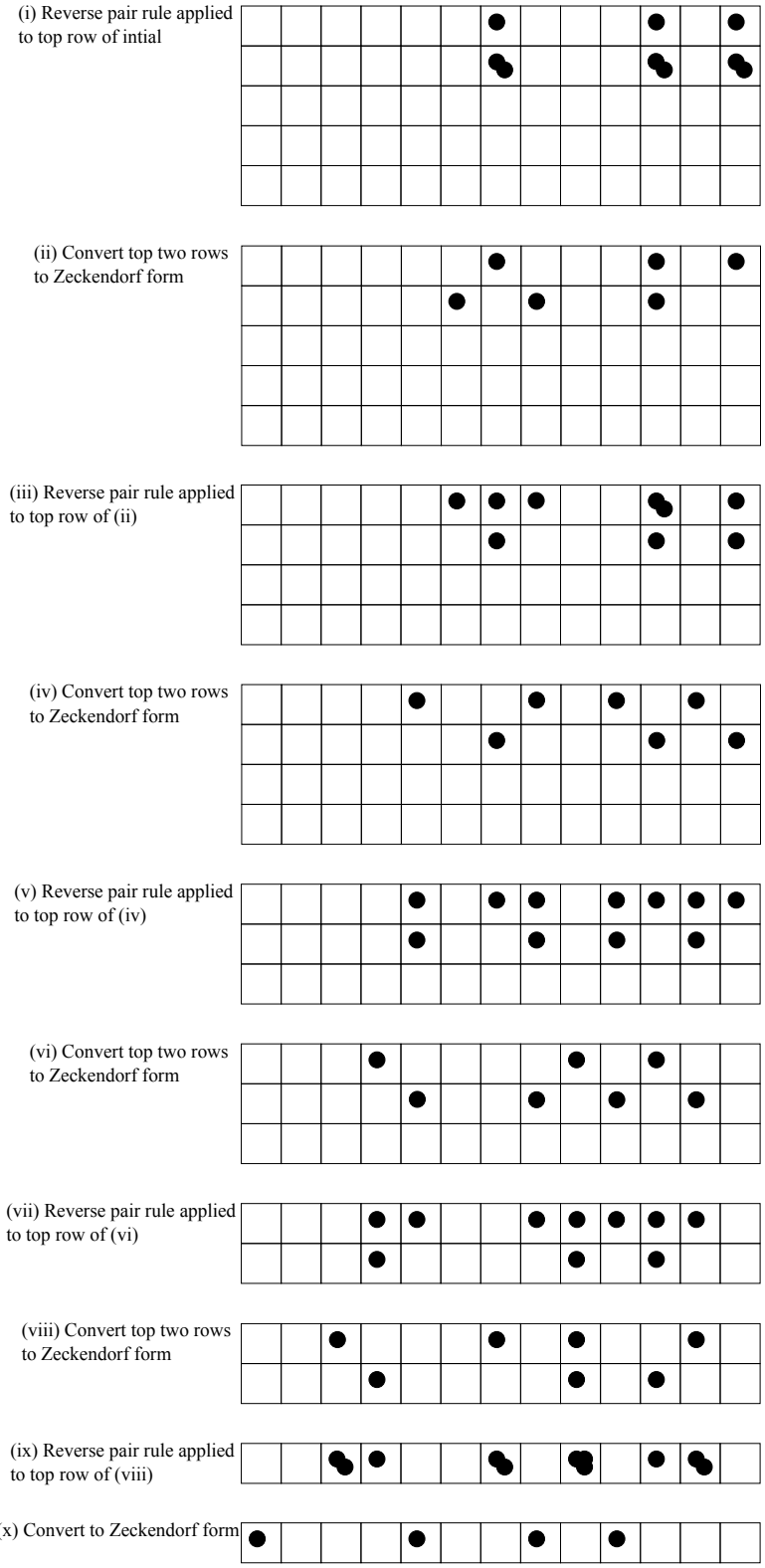

Figure 3: Multiplication using a checkerboard in Zeckendorf form, successive steps.

6 Conclusion

We have seen how Zeckendorf notation is an excellent way of representing a stream of natural numbers, particularly when either the maximum of the stream is not known beforehand, or smaller numbers are more likely than larger ones. The technique is particularly useful when representing the sequence of partial quotients that make up the continued fraction representation of an arbitrary irrational. While it is possible to generalize to sequences where each is the sum of more than two previous numbers, it turns out to be of limited utility.

We have also seen how addition and subtraction of numbers in Zeckendorf form is a straightforward task, and have shown a new approach to returning a sequence to Zeckendorf form that is more efficient. We have also seen four different ways of multiplying, and that using a checkerboard as a calculation tool has a certain elegance.

There are a number of future directions to consider. Which of these multiplication algorithms is most efficient? Does this depend on the magnitude of the numbers? Some of the authors already cited have suggested integer division (quotient and remainder) algorithms. How do they compare, particularly with our new efficient addition algorithm? In addition, Bunder [3] showed how all integers can be represented by Fibonacci numbers with negative coefficient, and Anderson and Bicknell-Johnson [2] showed how vectors built on k-bonacci numbers can be used to represent points in $\mathbb{Z}^{k-1}$. Can we do arithmetic on these in the same way?

References

[1] C. Ahlbach, J. Usatine, C. Frougny and N. Pippenger, Efficient Algorithms for Zeckendorf Arithmetic, *Fibonacci Quart.* **51**(3) (2013) 249-255.

[2] P.G. Anderson and M. Bicknell-Johnson, Multidimensional Zeckendorf representations, *Fibonacci Quart.* **49**(1) (2011) 4-9.

[3] M.W. Bunder, Zeckendorf representations using negative Fibonacci numbers, *Fibonacci Quart.* **30**(2) (1992) 111-115.

[4] R.A. Dunlap, *The Golden Ratio and Fibonacci Numbers*, World Scientific Publishing Company, 1998.

[5] Mark Feinberg, Fibonacci-tribonacci, *Fibonacci Quart.* **1**(3) (1963) 71-74.

[6] P. Fenwick, Zeckendorf integer arithmetic, *Fibonacci Quart.* **41**(5) (2003) 405-413.

[7] A.S. Fraenkel, Systems of numeration, *American Mathematical Monthly* **92**(2) (1985) 105-114.

[8] H.T. Freitag and G.M. Phillips, Elements of Zeckendorf arithmetic, *Applications of Fibonacci Numbers* **7** 129-132, G.E. Bergum, A.N. Philippou and A.F. Horadam (Eds.), Kluwer, Dordrecht 1998.

[9] Martin Gardner, *Knotted doughnuts and other mathematical entertainments*, W. H. Freeman and Company, 1986.

[10] G.H. Hardy and E.M. Wright, *An introduction to the theory of numbers, 5th edition*, Oxford University Press, London, 1979.

[11] A. Ya. Khinchin, *Continued Fractions*, Dover 1997 (originally University of Chicago Press 1964).

[12] C.G. Lekkerkerker, Voorstelling van natuurlijke getallen door een som van getallen van Fibonacci, *Simon Stevin* **29** (1952) 190-195.

[13] Mario Livio, *The Golden Ratio: The Story of ϕ, the World's Most Astonishing Number*, Broadway, 2003.

[14] G. Lochs, Vergleich der Genauigkeit von Dezimalbruch und Kettenbruch, *Abh. Hamburg Univ. Math. Sem.* **27** (1964) 142-144.

[15] C.D. Olds, *Continued Fractions*, Random House, 1963.

[16] G.J. Tee, Russian peasant multiplication and Egyptian division in Zeckendorf arithmetic, *Austral. Math. Soc. Gaz.*, **30**(5) (2003) 267-276.

[17] Steven Vadja, *Fibonacci and Lucas Numbers, and the Golden Section: Theory and Applications*, Dover Publications, 2007.

[18] Eric W. Weisstein, Lochs' Theorem, from *MathWorld–A Wolfram Web Resource* `http://mathworld.wolfram.com/LochsTheorem.html`.

[19] E. Zeckendorf, Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas, *Bull. Soc. Roy. Sci. Liége* **41** (1972) 179-182.