

Scan

AS 206

+

Gault & Clint

add to several

→ 5206
A 1468
A 14675
AS6/4

"CURIUSER AND CURIUSER" SAID ALICE. FURTHER REFLECTIONS ON AN INTERESTING RECURSIVE FUNCTION

D. GAULT AND M. CLINT

Department of Computer Science, The Queen's University of Belfast,
Belfast BT7 1NN, N. Ireland

(Received March 1988)

In this paper the close relationship between a previously investigated recursive function and the familiar Fibonacci sequence is established. An efficient iterative program for the computation of the recursive function, which is based on this relationship, is given. The correctness of the program follows from an earlier proof of equivalence of the recursive function and a function based on the Fibonacci sequence. The structural dissimilarity of the functional specification and the implementation suggests that it is unlikely that the latter can be generated from the former using standard transformation techniques.

KEY WORDS: Recursive function, Fibonacci sequence, program verification.

CR CATEGORIES: F.3, G.0.

1. INTRODUCTION

Burton and Campbell [1] have defined a recursive function, which they describe as curious, and have investigated it to show that it has an alternative closed form representation. The function is in fact even more curious than they revealed. In this paper it will be shown that the function is also the sequence of partial sums of a self-generating function and, in addition, that is closely related to a Fibonacci sequence. On the basis of these observations a concise iterative program for its evaluation is given.

2. OBSERVATIONS ON h

Definitions

A curious function

Burton and Campbell define the curious function, h , as follows:

$$h: \mathbb{N} \rightarrow \mathbb{N}$$

$$h(0) = 0$$

$$h(k) = k - h(h(k-1)), \quad k > 0.$$

Sequence of partial sums of a sequence: The sequence B is said to be the sequence of partial sums of the sequence A if the n th term in B is equal to the sum of the first n terms in A .

Observations: The integer sequence generated by h , of which the table below gives an initial subsequence, does not appear in Sloane's classic *Handbook of Integer Sequences* [2].

5206
8
Table 1 Table of $h(k)$ $0 \leq k < 50$

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	2	3	3	4	4	5	6
10	6	7	8	8	9	9	10	11	11	12
20	12	13	14	14	15	16	16	17	17	18
30	19	19	20	21	21	22	22	23	24	24
40	25	25	26	27	27	28	29	29	30	30

Observation of this initial subsequence, H , reveals that the number of occurrences of the value x in H is as follows,

Table 2 Table of SG1

	0	1	2	3	4	5	6	7	8	9
0	1	2	1	2	2	1	2	1	2	2
10	1	2	2	1	2	1	2	2	1	2
20	1	2	2	1	2	2	1	2	1	2

A14675
A1468

and that the expression $h(k) - h(k-1)$ takes the values in the table below

Table 3 Table of SG2, $0 < k < 50$

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	1	1	0	1	0	1	1
10	0	1	1	0	1	0	1	1	0	1
20	0	1	1	0	1	1	0	1	0	1

5614

The sequence $SG1$ is included in Sloane [2]. The reference there directs us to Mulcrone's Self Generating function [3] which is itself the solution to a problem originally posed by Pennington [4].

The sequence $SG2$, generated by the expression $h(k) - h(k-1)$, and which is therefore the sequence for which the sequence H is the sequence of partial sums, bears a striking resemblance to $SG1$. It appears that $SG2(x) = SG1(x) - 1$, $x > 0$.

3. PENNINGTON'S COWS, AND THE SEQUENCES $SG1$ AND $SG2$

The problem posed by Pennington [4] was stated as follows:

"A rancher bought a white cow, and the following year a red one. Each succeeding year he duplicated his purchases of the preceding two years, buying the same number of cows, of the same colours and in the same order. Thus, in the third year, he bought a white and then a red cow; in the fourth year, a red, then a

white, and then a red cow; and so on. What was the colour of the n th cow?"

Mulcrone's solution [3] is as follows:

Associate the number 1 with a white cow and the number 2 with a red cow, then the sequence defined by the problem is:

$$1; 2; 1, 2; 2, 1, 2; 1, 2, 2, 1, 2; 2, 1, 2, 1, 2, 2, 1, 2; \dots$$

where ";" denotes year boundaries.

This sequence, ignoring year boundaries, is the self generating sequence **SG1**. If the cows had been allocated the numbers 0 and 1 instead, **SG2** would have been produced.

$$0; 1; 0, 1; 1, 0, 1; 0, 1, 1, 0, 1; 1, 0, 1, 0, 1, 1, 0, 1; \dots$$

Further observations on SG2

DEFINITION

$$\mathbf{Fib}: \mathbb{N} \rightarrow \mathbb{N}$$

$$\mathbf{Fib}(0) = 1$$

$$\mathbf{Fib}(1) = 1$$

$$\mathbf{Fib}(n) = \mathbf{Fib}(n-1) + \mathbf{Fib}(n-2), \quad n > 1$$

Suppose that **SG2** is partitioned as follows

$$\begin{array}{cccccccc} 0; & 1; & 0; & 1; & 1, & 0; & 1, & 0, & 1; & 1, & 0, & 1, & 1, & 0; & 1, & 0, & 1, & 1, & 0, & 1; & \dots \\ \hline & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & & & & & & & & & & & & & \end{array}$$

i.e. starting with 0 and introducing boundaries after **Fib**(0), **Fib**(1), **Fib**(2), ... components.

Note that the cows purchased in the n th group ($n > 2$) is the sequence formed by concatenating the subsequences with indices 1 to $n-2$.

Let S_i be the i th subsequence of the partitioning described above. Let $\#S_i$ and $\#r_i$ denote the number of items in S_i and the number of ones in S_i , respectively.

- iv) 1, for $n > 2$,
- i) $S_n = S_1 \circ S_2 \circ \dots \circ S_{n-2}$
- ii) $\#S_n = \mathbf{Fib}(n-1)$
- iii) $\#r_n = \mathbf{Fib}(n-2)$
- $$\sum_{i=1}^n \#S_i = \mathbf{Fib}(n+1)$$
- v)
$$\sum_{i=1}^n \#r_i = \mathbf{Fib}(n)$$

(Note: \circ denotes sequence catenation.)

Proofs of these properties are straightforward but tedious and are therefore omitted.

The observations above suggest that, since the curious function is closely related to the self-generating function, **SG2**, which in turn is closely related to the Fibonacci function, **Fib**, it may be fruitful to investigate the possibility that **h** can be explicitly represented in terms of **Fib**. It will be shown later that such an explicit relationship can indeed be established. First, however, it is helpful to discuss a particularly apt system for the representation of natural numbers.

4. FIBARY REPRESENTATION OF NUMBERS

DEFINITION The Fibary representation of a number is similar to its binary representation with the exception that the weight of each position increases in a Fibonacci rather than an exponential manner. Thus, for example, the weight of the positions from right to left are 1, 1, 2, 3, 5, 8, 13 etc. rather than 1, 2, 4, 8, 16, etc.

Note, however, that the Fibary representation of a number may not be unique.

Table 4 Examples of Fibary representations

<i>n</i>	<i>Possible Fibary representations for n</i>					
0	00	or	0			
1	10	or	01			
2	100	or	11			
3	1000	or	101	or	110	
4	1010	or	1001	or	1010	
5	10000	or	1100	or	1011	
6	10010	or	10001	or	1110	or 1101
7	10100	or	10011	or	1111	
8	100000	or	11000	or	10110	or 10101

A normalised Fibary representation is characterised as follows:

The Fibary representation of 0 is 00

The Fibary representation of 1 is 10

The Fibary representation of $n > 1$ is constructed as follows:

If n is greater than or equal to the m th term in the Fibonacci sequence and less than the $m + 1$ th term then its Fibary representation is:

$$100\dots 00 + \text{normalised Fibary representation for } n - \text{Fib}(m).$$

m zeros

(The operation of addition is the same as for binary representation since no carries are involved.)

Table 5 Examples of normalised Fibary representations

n	Normalised Fibary representation for n
0	00
1	10
2	100
3	1000
4	1010
5	10000
6	10010
7	10100
8	100000

Notes

All normalised Fibary representations terminate with 0.

Each number representation has the structure $((10)0^{m_n})^n$ with $m_n, n \geq 0$.

Consequently, the string representing a number can never have two adjacent occurrences of 1.

The successor function

This function is defined by the following cases:

Case (i) If k has the normalised representation $x00(10)^y$ then $k+1$ has the normalised representation $x01(00)^y$.

Case (ii) If k has the normalised representation $x00(10)^y0$ then $k+1$ has the normalised representation $x01(00)^y0$.

Case (iii) If k has the normalised representation $x000$ then $k+1$ has the normalised representation $x010$.

Subtraction

Later it will be required to subtract a Fibary representation with a particular structure from a normalised Fibary representation with a related structure. The effect of this operation is captured by the following theorem.

THEOREM 1 *If the number with normalised Fibary representation $x00$ has the number with Fibary representation x subtracted from it, the difference has normalised Fibary representation $x0$.*

Proof A proof of this result is equivalent to proving that $x + x0 = x00$.

The proof is by induction

i) Base step:

$$x = 1$$

$$10 + 1 = 1 + 1 = 2 = 100.$$

ii) Induction step:

Suppose that $x0$ is the normalised Fibary representation of an arbitrary Natural number k and assume the result for all $n \leq k$.

The proof proceeds by addressing the three possible structures for k described above. The proof in each case is similar so only one case is treated below.

Consider Case (iii) above:

If k has the normalised Fibary representation $z000$ then $k+1$ has the normalised Fibary representation $z010$.

Since $x0 = z000$ it follows that $x = z00$ and $x00 = z0000$.

Thus $z000 + z00 = z0000$ by the induction hypothesis.

To show that the result is true for $k+1$, first note that $z010 = z000 + 10$.

Then

$$\begin{aligned} z010 + z01 &= (z000 + 10) + (z00 + 1) \dots \text{Fibary arithmetic} \\ &= (z000 + z00) + (10 + 1) \dots \text{by commutativity of } + \\ &= z0000 + 100 \dots \text{by induction hypothesis} \\ &= z0100 \end{aligned}$$

5. AN ALTERNATIVE DEFINITION of \mathbf{h}

DEFINITION Let \mathbf{G} be defined recursively by:

$$\mathbf{G}: \mathbb{N} \rightarrow \mathbb{N}$$

$$\mathbf{G}(0) = 0$$

$$\begin{aligned} \mathbf{G}(k) &= \mathbf{Fib}(m-1) + \mathbf{G}(k - \mathbf{Fib}(m)) \\ &\text{where } \mathbf{Fib}(m) \leq k < \mathbf{Fib}(m+1). \end{aligned}$$

Let $\mathbf{F}: \mathbb{N} \rightarrow \mathbb{N}$ be defined by:

$\mathbf{F}(n)$ = the number with Fibary representation that is the normalised Fibary representation of n with the last 0 removed.

$$\text{e.g. } \mathbf{F}(7) = \mathbf{F}(10100) = 1010 = 4$$

$$\mathbf{F}(20) = \mathbf{F}(1010100) = 101010 = 12$$

THEOREM 2 \mathbf{F} satisfies the definition of \mathbf{G} .

Proof By induction

i) Base step:

The normalised Fibary representation of 1 is 10

$$\mathbf{G}(10) = \mathbf{Fib}(0) + \mathbf{G}(0) \text{ by the recursive definition}$$

$$= 1 + 0 = 1$$

$$\mathbf{F}(10) = 1$$

ii) Induction step:

Assume the result for all $n < k$

$$G(k) = \text{Fib}(m-1) + G(k - \text{Fib}(m)) \text{ where } \text{Fib}(m) \leq k < \text{Fib}(m+1)$$

k has the normalised Fibary representation of $10x0$ (where x has $m-3$ digits)

$$G(k) = \text{Fib}(m-1) + G(x0).$$

Since $x < k$ it follows by the induction hypothesis that $G(x0) = x$.

Then $G(k) = \text{Fib}(m-1) + x = 10x$.

$$F(10x0) = 10x.$$

THEOREM 3 *The functions F and h are equivalent.*

Proof By induction

i) Base step:

This follows directly from the definitions of F and h .

$$F(0) = 0 = h(0)$$

$$F(1) = 1 = h(1).$$

ii) Induction step:

Assume that $F(j) = h(j)$ for all $j \leq k$.

$h(k+1) = k+1 - h(h(k))$... from the definition of h .

By the hypothesis $h(k) = F(k)$ and, since $h(k) \leq k$,

$$h(h(k)) = F(F(k))$$

$$h(k+1) = k+1 - F(F(k)).$$

However, k has one of the following normalised representations

a) $x00(10)^y$

b) $x00(10)^y0$

c) $x000$.

Consider each of these cases in turn

Case (a)

$$F(k) = x00(10)^{y-1}1 \quad \dots \text{definition of } F$$

$$= x01(00)^{y-1}0 \quad \dots \text{normalisation}$$

$$F(F(k)) = x01(00) \quad \dots \text{definition of } F$$

$$h(k+1) = x01(00) - x01(00) \quad \dots \text{successor}$$

$$= x01(00)^{y-1}0 \quad \dots \text{subtraction}$$

$$= F(k+1).$$

Case (b)

$$\begin{aligned}
 \mathbf{F}(k) &= x00(10)^y && \dots \text{definition of } \mathbf{F} \\
 \mathbf{F}(\mathbf{F}(k)) &= x00(10)^{y-1} && \dots \text{definition of } \mathbf{F} \\
 &= x01(00)^{y-1}0 && \dots \text{normalisation} \\
 \mathbf{h}(k+1) &= x01(00)^y0 - x01(00)^{y-1}0 && \dots \text{successor} \\
 &= x01(00) && \dots \text{subtraction} \\
 &= \mathbf{F}(k+1).
 \end{aligned}$$

Case (c)

$$\begin{aligned}
 \mathbf{F}(k) &= x00 && \dots \text{definition of } \mathbf{F} \\
 \mathbf{F}(\mathbf{F}(k)) &= x0 && \dots \text{definition of } \mathbf{F} \\
 \mathbf{h}(k+1) &= x010 - x0 && \dots \text{successor} \\
 &= 10 + x000 - x0 && \dots \text{Fibary arithmetic} \\
 &= 10 + x00 && \dots \text{subtraction} \\
 &= 01 + x00 && \dots \text{Fibary equivalence} \\
 &= x01 && \dots \text{Fibary arithmetic} \\
 &= \mathbf{F}(k+1).
 \end{aligned}$$

6. ITERATIVE PROGRAM FOR CALCULATING \mathbf{F}

The program below is easily extracted from the functional specification for \mathbf{G} . Since \mathbf{G} is primitive recursive and therefore total and since \mathbf{F} satisfies the specification of \mathbf{G} the program computes \mathbf{F} . Since \mathbf{F} has been shown to be equivalent to \mathbf{h} the program also computes \mathbf{h} .

```

fib, prevfib: = 1, 1;
{ Loop invariant
  There exists  $m$  such that
  fib =  $\mathbf{Fib}(m)$  &
  prevfib =  $\mathbf{Fib}(m-1)$  }
do fib <= n
  → fib, prevfib: = fib + prevfib, fib
  od;
f, n: = 0, N;
{ Loop invariant
  f =  $\mathbf{h}(N) - \mathbf{h}(n)$ 
  There exists  $m$  such that
  fib =  $\mathbf{Fib}(m)$  &
  prevfib =  $\mathbf{Fib}(m-1)$  &
  n <  $\mathbf{Fib}(m+1)$  }
do n > 0
  → if n >= fib
    → n, f: = n - fib, f + prevfib
    fi;
  fib, prevfib: = prevfib, fib - prevfib
  od;


```

The assertions which annotate the program may be used to generate the verification conditions on which the proof of correctness of the program rests. The proofs of these conditions follow straightforwardly from the discussion given earlier.

7. CONCLUSION

In this paper a further investigation of an interesting recursive function has been undertaken. This led to the discovery of an efficient iterative algorithm for its evaluation. The study provides further evidence that the ability to construct efficient implementations from functional specifications can sometimes require a degree of intuition beyond the scope of the transformational methods advocated by Burstall and Darlington [5] and others. The deficiency of such methods has already been pointed out by Dijkstra [6].

References

- [1] R. P. Burton and D. M. Campbell, A curious recursive function, *Intern. J. Computer Math.* **19** (1986), 245-257. 
- [2] N. J. A. Sloane, *A Handbook of Integer Sequences*, Academic Press, 1973.
- [3] T. Sulcrone, Solution, the red and white cows, *American Mathematical Monthly* **64** (1957), 197-198.
- [4] J. V. Pennington, Problems, the red and white cows, *American Mathematical Monthly* **63** (1956), 491.
- [5] R. M. Burstall and J. Darlington, A transformation system for developing recursive programs, *Journal of Assoc. Comput. Mach.* **24** (1977), 44-67.
- [6] E. W. Dijkstra, *Selected Writings on Computing: A Personal Perspective*, Springer-Verlag, 1982, pp. 215-216 and pp. 230-232.