

When is a number Fibonacci?

Phillip James

Department of Computer Science, Swansea University

March 26, 2009

Abstract

This article looks into the importance of the Fibonacci numbers within Computer Science, commenting on how to compute a Fibonacci number. It introduces an efficient test as to whether or not a number is Fibonacci, and proves the correctness of this test.

1 Introduction

Proposed by Leonardo Pisa in 1202¹, the Fibonacci numbers are a series of numbers of a certain form. Pisa's collection of mathematics 'Liber Abaci' [FS02] introduces the Fibonacci numbers through the well known Rabbit problem [FS02, CLR96]. The problem asks 'If we have a pair of rabbits, one male and one female, how many pairs of rabbits can be bred from this pair in a single year, given that a pair of rabbits do not breed within their first month'. Pisa shows that in a perfect situation (no deaths of rabbits etc), each pair of rabbits produces another pair of rabbits every month. This pattern of growth leads to a series of numbers that are known as the Fibonacci numbers.

Definition 1. *Fibonacci Numbers*

The sequence of numbers beginning 0, 1, 1, 2, 3... where each successive number in the sequence is the sum of the two preceding numbers. More precisely the sequence can be defined through the recurrence relation

¹Within European mathematics.

$$F_n = \begin{cases} 0, & \text{if } n = 0 \\ 1, & \text{if } n = 1 \\ F_{n-1} + F_{n-2}, & \text{if } n > 1 \end{cases}$$

The Rabbit problem given by Pisa is just one example of how the Fibonacci numbers arise within nature. There are many more interesting examples within nature, including the number of spirals on pine cones, and the pattern of seeds within sunflowers [Hun70].

The remainder of this article shall look at how the Fibonacci numbers play their role within computer science, and also look at some of the problems when computing with Fibonacci numbers.

2 Fibonacci Numbers Within Computer Science

Within Computer Science, the Fibonacci numbers and their properties are used in many different ways. Here we give a very brief outline of some ways in which the Fibonacci numbers are used. We shall then look at how the Fibonacci numbers can be computed, giving an example algorithm for such a task, and commenting on the efficiency of certain approaches.

2.1 Memory Allocation

A major operation performed by all computers, is the management of system memory. The so called ‘Buddy Scheme’ [DW96] is one particular scheme for performing memory management. This scheme involves recursively splitting up the system memory into smaller and smaller ‘buddies’ or pairs of blocks of memory. When a process requires a certain amount of system memory, the smallest block which is large enough for the data is chosen through this recursive buddy scheme. Usually the splitting sizes for splitting the memory into blocks uses a decreasing binary scheme (splitting in sizes which are powers of 2). Using such a binary based scheme means that the size difference between larger block pairs is rather big. If instead a splitting scheme following the Fibonacci sequence is used [DW96], the the difference between successive block sizes is always smaller than it would be using the binary scheme. This tends to generally reduce wastage of memory in the allocation process, improving the memory management scheme.

2.2 Run-time Of The Euclidean Algorithm

The Euclidean Algorithm [Sti06] is an algorithm that is largely used within cryptography and security in both Computer Science and the real world. Given two natural numbers, the algorithm computes the greatest common divisor of the two numbers. When analysing the run-time of the Euclidean Algorithm, it has been shown to have a worst case run-time of $O(n)$ [CLR96]. Interestingly this proof also shows that the inputs that require the largest number of computation steps, are inputs where the two numbers are successive Fibonacci numbers.

2.3 Computation Of Fibonacci Numbers

Using the recurrence relation for the Fibonacci numbers given in definition 1, we gain a natural recursive algorithm to compute the n^{th} Fibonacci number. For example consider the following algorithm written in the C programming language, that returns the value of the n^{th} Fibonacci number.

```
int fib(int n)
{
    if (n==0)
    {
        return 0;
    }
    else if (n==1)
    {
        return 1;
    }
    else
    {
        return fib(n-1)+fib(n-2);
    }
}
```

Figure 1: Recursive algorithm to compute the n^{th} Fibonacci number

Here to compute for example F_{100} , the algorithm will recursively compute $F_{99} + F_{98}$, building up the recursive tree in Figure 2.

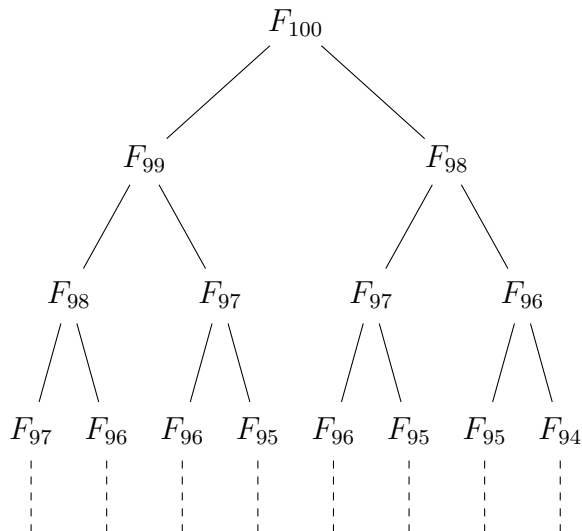


Figure 2: Recursive tree created to compute F_{100}

We see that the run-time of this algorithm grows exponentially as we try to compute larger Fibonacci numbers, mainly due to the number of repeated computations. For this reason it would be much better to use an algorithm based on the closed form definition of the Fibonacci numbers known as Binets formula [Hun70].

Definition 2. *Binets formula*

$$F(n) = \frac{\varphi^n - (-1/\varphi)^n}{\sqrt{5}} = \frac{\varphi^n - (-1/\varphi)^n}{\sqrt{5}}, \text{ where } \varphi = \frac{1+\sqrt{5}}{2}$$

An algorithm computing this equation would then simply run in constant time². Thus allowing larger Fibonacci numbers to be calculated in reasonable amounts of time.

3 Is A Number Fibonacci

Given that we can compute the value of a Fibonacci number efficiently, the next natural question to ask is, given a number x can we determine efficiently whether or not x is a Fibonacci number. For example is 19523 a

²Assuming the mathematical operations involved run in constant time.

Fibonacci number? Here the naive approach would be to calculate the series of Fibonacci numbers until we reach or exceed the value 19523. Again this approach would be rather inefficient, and take longer and longer the larger the value being tested gets. In the next section we shall look at a more elegant test of whether or not a number is a Fibonacci number.

Here we shall show that any positive integer x is a Fibonacci number if and only if $5x^2 \pm 4$ is a perfect square (i.e. an integer square number). To do this we shall firstly introduce two lemma's which we shall then use to prove our final theorem. We should note that this theorem and a corresponding proof were first given by Gessel in [Ges72]. Though the proof we present here takes the more commonly used Computer Science approach of applying the induction principle.

Lemma 1. *The equation*

$$y^2 - xy - x^2 = \pm 1,$$

is satisfied by $(x, y) = (f_n, f_{n+1})$, for all $n \geq 0$, $x, y \in \mathbb{N}$.

Proof. By induction on n .

- Base Case: ($n = 0$)

If $n = 0$ we have the Fibonacci pair $(F_0, F_1) = (0, 1)$, by substitution we gain

$$1^2 - 0 \cdot 1 - 0^2 = 1$$

Thus the base case holds

- Induction Step: ($n = n + 1$)

Assuming $F_{n+1}^2 - F_n F_{n+1} - F_n^2 = \pm 1$ as our induction hypothesis, now using $n + 1$ we gain

$$F_{n+2}^2 - F_{n+1} F_{n+2} - F_{n+1}^2 = \pm 1$$

Substituting $F_{n+2} = F_{n+1} + F_n$ we obtain

$$\begin{aligned} & (F_n + F_{n+1})^2 - F_{n+1}(F_n + F_{n+1}) - F_{n+1}^2 \\ &= F_n^2 + 2F_n F_{n+1} + F_{n+1}^2 - F_n F_{n+1} - 2F_{n+1}^2 \\ &= F_n^2 + F_n F_{n+1} - F_{n+1}^2 \\ &= -(F_{n+1}^2 - F_n F_{n+1} - F_n^2) && \text{(i.h.)} \\ &= -(\pm 1) = \pm 1 \end{aligned}$$

Thus the induction step holds. \square

Lemma 2. *If (x, y) is a pair of positive integers satisfying the equation*

$$y^2 - xy - x^2 = \pm 1,$$

then $(x, y) = (F_n, F_{n+1})$ for some $n \geq 0$.

Proof. By induction on $(x + y)$

Firstly, we note that since x and y are positive, it must be that $x \leq y$, since if $x > y$ then

$$y^2 - xy - x^2 < y^2 - y^2 - y^2 = -y^2$$

and hence $y^2 - xy - x^2 < -1$.

Secondly, if $x = y$ then substituting y for x we obtain

$$y^2 - xy - x^2 = y^2 - y^2 - y^2 = -y^2 = \pm 1,$$

hence setting $(x, y) = (F_1, F_2)$, satisfies this equation.

Finally assuming that $1 \leq x < y$, and that $y^2 - xy - x^2 = \pm 1$. We have that the pair of positive integers

$$(a, b) = (y - x, x)$$

also satisfies the equation $y^2 - xy - x^2 = \pm 1$, since

$$\begin{aligned} b^2 - ab - a^2 &= x^2 - (y - x)x - (y - x)^2 \\ &= x^2 - xy + x^2 - y^2 + 2xy - x^2 \\ &= -(y^2 - xy - x^2) = \pm 1 \end{aligned}$$

Therefore, by induction, $(a, b) = (F_n, F_{n+1})$ for some n , and hence

$$x = b = F_{n+1} \text{ and } y = a + x = F_n + F_{n+1}$$

meaning $(x, y) = (f_{n+1}, f_{n+2})$ \square

Theorem 1. *A positive integer x is a Fibonacci number if, and only if,*

$$5x^2 \pm 4$$

is a perfect square.

Proof. The solution of the quadratic equation $y^2 - xy - x^2 = \pm 1$ is

$$y = \frac{x \pm \sqrt{x^2 + 4(x^2 \pm 1)}}{2}$$

or equivalently

$$y = \frac{1}{2}(x \pm \sqrt{5x^2 \pm 4}).$$

Assuming y is positive we can simplify this to

$$y = \frac{1}{2}(x + \sqrt{5x^2 \pm 4}).$$

” \Rightarrow ”

Assuming x is a Fibonacci number F_n for some n , then by lemma 1, y must also be a Fibonacci number, namely F_{n+1} . Thus for y to be a Fibonacci number (and hence an integer), $\sqrt{5x^2 \pm 4}$ must be an integer. Thus $5x^2 \pm 4$ must be a perfect square.

” \Leftarrow ”

Conversely, assuming $5x^2 \pm 4$ is a perfect square, and given that x is a positive integer, y must also be a positive integer, since by basic mathematical laws of addition and multiplication we have:

- If x is odd, then given the fact that the set of odd numbers is closed under multiplication, both x^2 and thus $5x^2$ must be odd. Adding 4 to an odd number will again yield an odd number. Therefore $5x^2 \pm 4$ is odd and a perfect square. If we in turn take the square root of an odd perfect square, the result will be odd. So finally we have

$$\frac{1}{2}(x + \text{oddvalue}).$$

Since x is odd, adding two odd numbers will result in an even number, which when divided by 2 will result in an integer. Thus under the assumption x is odd y must be an integer.

- If x is even, then given the fact that the set of even numbers is also closed under multiplication, x^2 will be even. Now multiplying a even number by 5 will also yield an even number and thus $5x^2$ must be even. Adding 4 to

an even number will again yield an even number. Therefore $5x^2 \pm 4$ is even and a perfect square. If we in turn take the square root of an even perfect square, the result will be even. So finally we have

$$\frac{1}{2}(x + \text{evenvalue}).$$

Since x is even, adding two even numbers will result in an even number, which when divided by 2 will result in an integer. Thus under the assumption x is even y must be an integer.

Given that y must be a positive integer, by lemma 2, both x and y must be Fibonacci numbers. \square

4 Summary

We have taken a very brief look into the Fibonacci numbers, commenting on their uses within Computer Science. We have shown the form of a simple function to compute Fibonacci numbers, and finally presented a theorem and proof of how we may efficiently test whether or not a number is a Fibonacci number.

References

- [CLR96] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to algorithms*. The MIT press, 2nd edition, 1996.
- [DW96] Nell Dale and Henry M. Walker. *Abstract data types: specifications, implementations, and applications*. D. C. Heath and Company, Lexington, MA, USA, 1996.
- [FS02] Leonardo Fibonacci and Laurence E. Sigler. *Fibonacci's Liber Abaci*. Springer, 2002.
- [Ges72] Ira Gessel. Problem H-187. volume 10 of *Fibonacci Quarterly*, pages 417–419, 1972.
- [Hun70] H. Huntley. *The Divine Proportion: A Study in Mathematical Beauty*. Courier Dover Publications, 1970.
- [Sti06] Douglas Robert Stinson. *Cryptography: Theory and Practice*. CRC Press, 3 edition, 2006.