Computing Fibonacci Numbers Fast using the Chinese Remainder Theorem

Adam Murakami
Oregon State University

Advisor: Paul Cull
Oregon State University

August 15, 2003

Abstract

The purpose of this paper is to investigate the calculation of Fibonacci numbers using the Chinese Remainder Theorem (CRT). This paper begins by laying down some general conclusions that can be made about the Fibonacci sequence. It will then go into specific cases of the CRT and how to calculate Fibonacci numbers with reduced forms of the CRT equations. For each of the cases, algorithms and analysis will be presented to try and give some perspective of the outcomes.

1 Introduction

The inspiration for trying to use the Chinese Remainder Theorem (CRT) to compute Fibonacci numbers was not an original idea I had. I was given a paper that claimed to be able to use the CRT to compute Fibonacci numbers. I say claimed due to the fact that it did not contain enough explaination and data to be understandable. This paper was simply known as the Babb's paper.

I began by reading up on the CRT and Number Theory in general. I used a number of websites and books to help me ([1],[9],[10]). I figured after a brief infusion of Number Theory knowledge; I should attempt to create a working example of the process that Babb laid out in his paper. After much tinkering and effort, I got an example to work. With this working example I was able to finally decipher his paper.

This paper is an attempt to explain the ideas and processs behind his thinking in a clearer fashion. Some of the information will mirror the information in the Babb's paper but to prove it to myself I worked out everything that he stated in his paper for myself and so I may tell it in a different fashion then found in his paper. One of the problems with his paper was that there were no proofs provided. I try to show all my work to hopefully reduce confusion.

This paper will also extend the procedures in Babb's paper to paths that were only hinted at by Babb. Then I hope to report my findings in a way that shows some conclusions.

1.1 Theorems and Equations

The following is a list of equations that will be commonly used throughout this paper.

$$f_{n+m}f_r = f_{r+m}f_n - (-1)^r f_{n-r}f_m \qquad Halton\ (1965)\ [7] \quad (1)$$

$$f_{n+1}f_{n-1} = f_n^2 + (-1)^n \qquad Cassini's\ Formula\ [1][7] \quad (2)$$

$$f_{n+k} = f_{n-1}f_k + f_n f_{k+1} \qquad Vorob'ev\ Formula\ [3][7] \quad (3)$$

1.2 The Chinese Remainder Theorem

Let f_n denote the *nth* Fibonacci number of the sequence $f_0 = 0,\ f_1 = 1,\ f_2 = 1,\ f_n = f_{n-1} - f_{n-2}$. The CRT is stated using notation and variable names which will be used consistently throughout this paper.

Theorem 1.1 *Let m_1,m_2,..., m_n be integers greater than 0, relatively prime in pairs, and let r_1,r_2,...,r_n be any integers. Consider the system of congruences*

$$f_x \equiv r_1 \ (mod\ m_1)$$

$$f_x \equiv r_2 \ (mod\ m_2)$$

$$\dots$$

$$f_x \equiv r_n \ (mod\ m_n).$$

Let $M = m_1 m_2 \cdots m_n$ and $M_i = \frac{M}{m_i}$ for i = 1, 2, ..., n. Let y_i be a solution to

$$M_i y_i \equiv 1 \ (mod\ m_i) \qquad for\ i = 1, 2, \dots, n.$$

Then a solution to the original system of congruences is

$$f_x = \sum_{i=1}^{n} M_i y_i r_i \ (mod\ M).$$

This paper will find specific remainders r_i, inverses y_i, and relatively primes m_i values for the CRT such that when f_x is computed, f_x will be the *xth* Fibonacci number.

1.3 Iterative Method to Compute Fibonacci Numbers

```
IterativeFibonacci(x)
{
    a = 0;
    b = 1;
    if(x < 1)
        return 0;
    for i from 2 to x do
    {
        if(a < b)
            a = a + b;
        else
            b = a + b;
    }
    return max(a, b);
}
```

1.3.1 Analysis of Algorithm

The characteristic equation obtained from the Fibonacci numbers difference equation, $f_n = f_{n-1} + f_{n-2}$, is $x^2 - x - 1$, where λ_1 and λ_2 are the two roots defined by

$$\lambda_1 = \frac{1+\sqrt{5}}{2}, \qquad \text{and} \qquad \lambda_2 = \frac{1-\sqrt{5}}{2}.$$

Other algorithms will be compared to this iterative algorithm throughout this paper. A calculation of the number of bit operations is needed to compute a given Fibonacci number. This analysis uses γn to represent the number of bits in f_n where $\gamma = log(\lambda_1) \approx 0.69424$ since f_n is asymptotic to $\frac{\lambda_1^n}{\sqrt{5}}$.

According to a paper by Cull and Holloway [4], the number of bit operations needed to compute f_n using this iterative method is

$$\frac{\gamma n(n-1)}{2}$$

1.4 Modular Recurrences: Fibonacci Numbers

Corollary 1.2 *For any $\alpha \geq 2$, the Fibonacci function modulo α has a periodic orbit for every pair of initial values. [5]*

This periodicity is the motivation for using the CRT. Since the Fibonacci numbers modulo numbers greater then 2 will yield a periodic orbit, then the remainders r will be contained within the orbit. Since the α values will correspond to particular m_i's values in the CRT, these values will

need to be pairwise prime. One limitation of the CRT is that it is only able to compute numbers that are less then M. Therefore, the values of α need to be large enough such that they yield small periods. This will compute larger Fibonacci numbers.

Similar to Babb's paper, I wrote my own program that would find the period lengths of the Fibonacci sequence modulo integers. Also, I found that the Fibonacci sequence modulo Fibonacci numbers yielded the desirable results which Babb discussed. I went on to discover that the cycle length for the Fibonacci sequence modulo f_n was very predictable.

Theorem 1.3 *Given an integer n, let f_n denote the nth Fibonacci number. The sequence given by the Fibonacci sequence modulus f_n, then if n is even, then the cycle length will be 2n and furthermore the sequence is exactly*

$$0, 1, 1, f_3, \ldots, f_{n-1},$$

$$0, f_{n-1}, -f_{n-2}, f_{n-3}, -f_{n-4}, \ldots, -f_4, f_3, -1, 1.$$

if n is odd, then the cycle length will be 4n and furthermore the sequence is exactly

$$0, 1, 1, f_3, \ldots, f_{n-1},$$

$$0, f_{n-1}, -f_{n-2}, f_{n-3}, -f_{n-4}, \ldots, f_4, -f_3, 1, -1$$

$$0, -1, -1, -f_3, \ldots, -f_{n-1},$$

$$0, -f_{n-1}, f_{n-2}, -f_{n-3}, f_{n-4}, \ldots, -f_4, f_3, -1, 1.$$

Proof. The following two equations are used largely throughout this proof.

$$\begin{aligned} f_{n+1} f_{n-1} &= f_n^2 + (-1)^n \text{ (equation 2)} \\ (f_n + f_{n-1}) f_{n-1} &= f_n^2 + (-1)^n \\ f_{n-1}^2 &\equiv (-1)^n \ (mod\ f_n) \text{ (equation 2.1)} \end{aligned}$$

$$\begin{aligned} f_{n+k} &= f_{n-1} f_k + f_n f_{k+1} \text{ (equation 3)} \\ &\equiv f_{n-1} f_k \ (mod\ f_n) \text{ (equation 3.1)} \end{aligned}$$

All terms beyond f_{n-1} will be modulus f_n. So the first n terms will obviously be the sequence $0, 1, 1 \ldots f_{n-1}$. It is given from equation 3.1 that the second n terms will be the corresponding elements from the first n terms multiplied by f_{n-1}. So the second block of terms by the equation will be

$$0, f_{n-1}, \ldots, f_{n-1}^2.$$

Using equation 2.1, if n is even then $f_{n-1}^2 = 1$ and if n is odd then $f_{n-1}^2 = -1$. If n is even then the last term is 1 followed by a zero and then the sequence repeats. Therefore the cycle length is $2n$. Otherwise if n is odd then the last term is -1 followed by a zero. This will result in the negation of the first $2n$ terms. The end of the second $2n$ terms will be 1 followed by a zero and then the sequence repeats and therefore the cycle length will be $4n$.

The actual composition of the terms of the second block of the sequence can be shown to be the sequence $0, f_{n-1}, -f_{n-2}, f_{n-3}, -f_{n-4}, \ldots, -f_4, f_3, -1, 1$. Equation 3.1 will help show this result. The next few terms of the sequence after the first n terms will be $f_{n-1}f_1$, $f_{n-1}f_2$, $f_{n-1}f_3$, $f_{n-1}f_4$, rewritten $f_{n-1}, f_{n-1}, 2f_{n-1}, 3f_{n-1}$. Reducing these terms yields a pattern

$$f_{n-1}$$

$$f_{n-1} = f_n - f_{n-2} = -f_{n-2} \text{ since terms will be modulus } f_n$$
$$2f_{n-1} = f_{n-1} + f_{n-1} = -f_{n-2} + f_{n-2} + f_{n-3} = f_{n-3}$$
$$3f_{n-1} = 2f_{n-1} + f_{n-1} = f_{n-3} + f_n - f_{n-2} = -f_{n-4}.$$

From equation 2, the final few terms of the second block of the sequence will be $f_{n-1}f_{n-4}, f_{n-1}f_{n-3}, f_{n-1}f_{n-2}, f_{n-1}^2$. This time reducing starting at f_{n-1}^2 and assuming that n is even

$$f_{n-1}^2 = 1 \text{ from equation 4}$$

$$f_{n-1}f_{n-2} = f_{n-1}(f_n - f_{n-1}) = -f_{n-1}^2 = -1$$
$$f_{n-1}f_{n-3} = f_{n-1}(f_{n-1} - f_{n-2}) = f_{n-1}^2 - f_{n-1}f_{n-2} = 1 - (-1) = 2$$
$$f_{n-1}f_{n-4} = f_{n-1}(f_{n-2} - f_{n-3}) = f_{n-1}f_{n-2} - f_{n-1}f_{n-3} = -1 - (2) = -3.$$

So the second block of terms is simply the first block in descending order with an alternating sign. ■

1.4.1 Finding Remainders

The nice, predictable form of the period yields an easy way to find the remainders r for the CRT. The first n terms of the sequence is simply the first n Fibonacci numbers and the next n terms is just the first sequence descending from n with an alternating sign. When n is even, the last few terms are 2, -1, 1; which will then repeat the sequence. When n is odd, the last few terms are -2, 1, -1; which will then repeat the sequence with the opposite sign. Then on the second pass the last few terms will be 2, -1, 1; which will then repeat the sequence. This predictable pattern allows, with the knowledge of the first n Fibonacci numbers, for finding the remainder r that corresponds to a given f_n.

Given an x that corresponds to the Fibonacci number that is to be computed, Babb developed four equations to find the remainder of a given f_n. Only the first n Fibonacci numbers are required because the rest of the sequence is just positive or negative versions those numbers. The second equation gives whether or not the xth term falls in an ascending or descending half of the sequence. The third equation gives two things. It shows if the index n is odd or even and if it is odd, it will also calculate if the xth term falls in the first $2n$ sequence or the second $2n$ sequence of the odd sequence. Then finally the fourth equation calculates the actual remainder for a given n. The four equations are:

1. $j = x \ (mod\ n)$

2. $k = \lfloor \frac{x}{n} \rfloor \ (mod\ 2)$

3. $l = (\lfloor \frac{x}{2n} \rfloor \ (mod\ 2)(n\ (mod\ 2))$

4. $r = (-1)^{k(j+1)+l} f_{kn+(-1)^k j}$.

In the second equation, if k is 0 then the term is in the ascending portion of the sequence and if k is 1 then it is in the alternating, descending portion of the sequence. The third equation will be 0 if the index n is even. If n is odd then it will be 0 if it is in the first $2n$ terms of the sequence and 1 if the term is in the second $2n$ terms of the sequence, which are negative versions of the first $2n$ terms. The last equation will yield the remainder and is the combination of j, k, and l.

With this general information it is now possible to handle more specific cases. Babb's paper dealt with a way to calculate Fibonacci numbers using three consecutive Fibonacci terms as the pairwise primes, m_i. The following section will mimic his work and extend his findings further.

2 Three Terms

2.1 Using Three Consecutive Fibonacci Numbers

The approach that Babb pursued consisted of three consecutive Fibonacci numbers n, $n+1$,and $n+2$ as his choices for the relatively prime terms, m_1, m_2, and m_3. The following equation is used to show that these three terms will always be pairwise prime:

$$GCD(f_i, f_j) = f_{GCD(i,j)}.$$

This equation shows that any consecutive terms will be relatively prime. Since any two consecutive numbers have a GCD of 1. Also the maximum value that the greatest common divisor can have between any number n and $n+2$ is 2. This means that for any three consecutive Fibonacci numbers the GCD between any two will be at most f_2 which is 1. So for any n we choose f_n, f_{n+1}, and f_{n+2} will be relatively prime.

Before continuing any further, an estimation what Fibonacci numbers can be calculated with this method would be helpful. The size of the Fibonacci number calculatable is limited by the M value. For three terms, $M = f_n f_{n+1} f_{n+2}$ and so an estimation is needed for this product. To do this, a rounded version of Binet's formula is used,

$$f_n = \frac{1}{\sqrt{5}} \lfloor \alpha^n \rfloor$$

$$\alpha = \frac{1+\sqrt{5}}{2}.$$

Substituting the equation into the epression for M, an estimate is obtained for the maximum Fibonacci number that can be calculated for a given n value.

$$f_n f_{n+1} f_{n+2} = \Big(\frac{1}{\sqrt{5}}\alpha^n\Big)\Big(\frac{1}{\sqrt{5}}\alpha^{n+1}\Big)\Big(\frac{1}{\sqrt{5}}\alpha^{n+2}\Big) = \frac{1}{(\sqrt{5})^3}\alpha^{3n+3} \approx \frac{1}{\sqrt{5}}\alpha^{3n} \approx f_{3n}$$

This means that a n value can be roughly used such that it is a third of the desired x value. To make this procedure more efficient, the following equation is used to calculate the n values:

$$n = \lfloor \frac{x}{3} \rfloor + 1.$$

2.2 Inverse Values

After some work the results that Babb found for the inverse values y_i were confirmed. The inverse values for three consecutive terms simplified to an easy value to compute.

Theorem 2.1 *Given an integer n, let the three pairwise prime values used in the Chinese Remainder Theorem m_1, m_2, and m_3 be f_n, f_{n+1}, and f_{n+2} respectively. Then if $M_i = \frac{m_1 m_2 m_3}{m_i}$ the solution to the equation $M_i y_i \equiv 1 \pmod{m_i}$ for i = 1, 2, and 3 is*

$$y_1 \equiv (-1)^n \pmod{f_n}$$

$$y_2 \equiv (-1)^{n+1} \pmod{f_{n+1}}$$
$$y_3 \equiv (-1)^{n+1} \pmod{f_{n+2}}.$$

Proof. Since n is given, it is possible to simplify the equations. The three equations can be simplified quickly using the fact that $f_n = f_{n-1} + f_{n-2}$ and equation 2

$$f_{n+1} f_{n-1} = f_n^2 + (-1)^n.$$

The equations can then be simplified down to constant time equations. So each equation is dealt with separately. Since the equations are modulo some number, those simplifications are done when possible.

$$\begin{aligned} 1)\ f_{n+2} f_{n+1} y_1 &\equiv 1 \pmod{f_n} \\ f_{n+2} f_{n+1} y_1 &= (f_{n+1} + f_n)(f_n + f_{n-1}) y_1 \\ f_{n+1} f_{n-1} y_1 &= (f_n^2 + (-1)^n) y_1 = (-1)^n y_1 \\ y_1 &\equiv \frac{1}{(-1)^n} \pmod{f_n} \\ y_1 &\equiv (-1)^n \pmod{f_n} \end{aligned}$$

$$\begin{aligned} 2)\ f_{n+2} f_n y_2 &\equiv 1 \pmod{f_{n+1}} \\ f_{n+2} f_n y_2 &= (f_{n+1}^2 + (-1)^{n+1}) y_2 = (-1)^{n+1} y_2 \\ y_2 &\equiv \frac{1}{(-1)^{n+1}} \pmod{f_{n+1}} \\ y_2 &\equiv (-1)^{n+1} \pmod{f_{n+1}} \end{aligned}$$

3) $f_{n+1}f_n y_3 \equiv 1 \ (mod \ f_{n+2})$

$$\begin{aligned} f_{n+1}f_n y_3 &= (f_{n+3}-f_{n+2})(f_{n+2}-f_{n+1})y_3 \\ -f_{n+3}f_{n+1}y_3 &= -(f_{n+2}^2+(-1)^{n+2})y_3 = (-1)^{n+3}y_3 \\ y_3 &\equiv \frac{1}{(-1)^{n+1}} \ (mod \ f_{n+2}) \\ y_3 &\equiv (-1)^{n+1} \ (mod \ f_{n+2}) \end{aligned}$$

■

2.3 Remainder Values

With the use of a program that I wrote in Maple 8 using the four general equations for finding the remainders, I observed that the remainders followed an observable pattern that I later proved.

Theorem 2.2 *Given an integer x that corresponds to a desired Fibonacci number, then let $n = \lfloor \frac{x}{3} \rfloor + 1$ and let $a = x \ (mod \ 3)$ then using the Chinese Remainder Theorem with three consecutive Fibonacci numbers f_n, f_{n+1}, and f_{n+2} the remainders are exactly*

$$r_1 = (-1)^n f_{n-3+a}$$

$$r_2 = (-1)^{n+1} f_{n-5+a}$$

$$r_3 = (-1)^n f_{n-7+a}.$$

Proof. To prove the equations three cases are used, $x = 3i, 3i+1$, and $3i+2$ for any $i = 1, 2, \ldots$ to cover all the possible modulus values. The method used was to compute and simplify each x case.

1. $x = 3i$

$$n = \lfloor \frac{3i}{3} \rfloor + 1 = i+1$$

$$j = 3i \ (mod \ i+1) = i-2$$

$$k = \lfloor \frac{3i}{i+1} \rfloor \ (mod \ 2) = 2 \ (mod \ 2) = 0$$

$$l = \begin{cases} 0 & \text{if i is odd and thus n is even.} \\ 1 & \text{if i is even and thus n is odd.} \end{cases}$$

then to put r_1 into terms of $n = i+1$

$$r_1 = \begin{cases} f_{i-2} \to f_{n-3} & \text{if n is even.} \\ -f_{i-2} \to -f_{n-3} & \text{if n is odd.} \end{cases}$$

$$n = i+2 \text{ simply the } n \text{ value incremented from } r_1$$

$$j = 3i \ (mod\ i+2) = i-4$$

$$k = \lfloor \frac{3i}{i+2} \rfloor \ (mod\ 2) = 2 \ (mod\ 2) = 0$$

$$l = \begin{cases} 1 & \text{if i is odd and thus n is odd.} \\ 0 & \text{if i is even and thus n is even.} \end{cases}$$

then to put r_2 into terms of $n = i+1$

$$r_2 = \begin{cases} -f_{i-4} \rightarrow -f_{n-5} & \text{if n is even.} \\ f_{i-4} \rightarrow f_{n-5} & \text{if n is odd.} \end{cases}$$

$$n = i+3 \text{ simply the } n \text{ value incremented from } r_2$$

$$j = 3i \ (mod\ i+3) = i-6$$

$$k = \lfloor \frac{3i}{i+3} \rfloor \ (mod\ 2) = 2 \ (mod\ 2) = 0$$

$$l = \begin{cases} 0 & \text{if i is odd and thus n is even.} \\ 1 & \text{if i is even and thus n is odd.} \end{cases}$$

then to put r_3 into terms of $n = i+1$

$$r_3 = \begin{cases} f_{i-6} \rightarrow f_{n-7} & \text{if n is even.} \\ -f_{i-6} \rightarrow -f_{n-7} & \text{if n is odd.} \end{cases}$$

2. $x = 3i+1$

   Similar results as in case 1 but since $x = 3i+1$, then the j value will now be incremented by 1 and thus the general result from case 1 but with all the index values increased by 1.

   So this would result in remainder values of f_{n-2}, f_{n-4}, f_{n-6} respectively. Letting $a = x \ (mod\ 3) = 1$, the remainders will be f_{n-3+a}, f_{n-5+a}, and f_{n-7+a} respectively.

3. $x = 3i+2$

   Similar result as well but with $x = 3i+2$ and thus increasing the index by 2 from case 1. So the resulting remainders will be for $a = x \ (mod\ 3) = 2$, f_{n-1}, f_{n-3}, f_{n-5}.

   Therefore, the equation at the end of case 2 applies for the general case for which $a = x \ (mod\ 3)$ and the resulting equation f_{n-3+a}, f_{n-5+a}, and f_{n-7+a}.

■

2.4 Reduction of the CRT General Equation

With the results found for the inverse and remainder values, it was possible to simplify down the general equation given by the CRT. The reduced equation requires only two significant multiples and requires no modulus operation. From the CRT it is known that for three terms the solution will be

$$f_x = r_1 y_1 M_1 + r_2 y_2 M_2 + r_3 y_3 M_3 \ (mod\ M).$$

Plugging in what is known from the two previous theorems, the general equation can be reduced. The following is the result

Theorem 2.3 *Given an integer x that corresponds to the desired Fibonacci number, then let* $n = \lfloor \frac{x}{3} \rfloor + 1$ *then* f_x *can be determined using the following set of equations:*

$$n = \lfloor \frac{x}{3} \rfloor + 1$$

$$k = x \ (mod\ 3)$$

$$l = \lceil \frac{k}{2} \rceil$$

$$m = k \ (mod\ 2)$$

$$f_x = 5 f_n f_{n-1} f_{n-2+k} + (-1)^{n+1} (-2)^{|l-1|} f_{n-1+3m-l}.$$

Proof. To show the final equation, three cases $k = 0$, 1, and 2 need to be considered and simplified down. Before this can happen, the equation can be simplified in general.

$$\begin{aligned} y_i r_i &= < (-1)^{2n} f_{n-3+k}, (-1)^{2n+2} f_{n-5+k}, (-1)^{2n+1} f_{n-7+k} > \\ &= < f_{n-3+k}, f_{n-5+k}, -f_{n-7+k} > \end{aligned}$$

To make things easier , some of the equalities and equations are listed that are used here and case specific ones will appear at the beginning of each case.

Equation 2 gets used frequently throughout this proof with many different index values and rearrangements.

$$f_{n+1} f_{n-1} = f_n^2 + (-1)^n \qquad \text{(equation 2)}$$

1. $k = 0 \ ||\ f_{n+1} f_{n+2} f_{n-3} + f_n f_{n+2} f_{n-5} - f_n f_{n+1} f_{n-7}$
   Using equation 1 with three different values

$$f_{n+2} f_{n-3} = f_n f_{n-1} + 2(-1)^n$$
$$f_{n+2} f_{n-5} = f_{n-1} f_{n-2} + 6(-1)^n$$
$$f_{n+1} f_{n-7} = f_{n-3}^2 + 9(-1)^n$$

$$\begin{aligned} f_{n-3}^2 &= f_{n-3}(f_{n-1}-f_{n-2}) \\ &= f_{n-1}f_{n-3}-f_{n-2}f_{n-3} \\ &= f_{n-1}f_{n-3}-f_{n-2}(f_{n-1}-f_{n-2}) \\ &= f_{n-1}f_{n-3}-f_{n-1}f_{n-2}+f_{n-2}^2 \\ &= f_{n-1}f_{n-3}-f_{n-1}f_{n-2}+f_{n-1}f_{n-3}+(-1)^{n+1} \\ &= f_{n-1}(f_{n-3}-f_{n-2}+f_{n-3})+(-1)^{n+1} \\ &= f_{n-1}(2f_{n-3}-f_{n-2})+(-1)^{n+1} \end{aligned}$$

$$\begin{aligned} f_{n+1} &= f_n+f_{n-1} \\ &= (f_{n-1}+f_{n-2})+(f_{n-2}+f_{n-3}) \\ &= (f_{n-2}+f_{n-3})+2f_{n-2}+f_{n-3} \\ &= 3f_{n-2}+2f_{n-3} \end{aligned}$$

$$f_{n+1}f_{n+2}f_{n-3}+f_nf_{n+2}f_{n-5}-f_nf_{n+1}f_{n-7}$$

$$= f_{n+1}\big(f_nf_{n-1}+2(-1)^n\big)+f_n\big(f_{n-1}f_{n-2}+6(-1)^n\big) - f_n\big(f_{n-3}^2+9(-1)^n\big)$$

$$= f_{n+1}f_nf_{n-1}+f_nf_{n-1}f_{n-2}-f_nf_{n-3}^2 +(-1)^n(2f_{n+1}+6f_n-9f_n)$$

$$= f_n\big[f_{n+1}f_{n-1}+f_{n-1}f_{n-2}-f_{n-3}^2\big]+(-1)^n(2f_{n+1}-3f_n)$$

$$= f_n\big[f_{n-1}(f_{n+1}+f_{n-2})-(f_{n-1}(2f_{n-3}-f_{n-2})+(-1)^{n+1})\big] +(-1)^n(2f_n+2f_{n-1}-3f_n)$$

$$= f_nf_{n-1}\big[f_{n+1}+f_{n-2}-2f_{n-3}+f_{n-2}\big] +(-1)^n(2f_{n-1}-f_n)+(-1)^nf_n$$

$$= f_nf_{n-1}\big[(3f_{n-2}+2f_{n-3})-2f_{n-3}+2f_{n-2}\big] +(-1)^n(2f_{n-1}-f_n+f_n)$$

$$= f_nf_{n-1}\big[5f_{n-2}\big]+(-1)^n(2f_{n-1})$$

$$= 5f_nf_{n-1}f_{n-2}+2(-1)^nf_{n-1}$$

2. $k=1 \;||\; f_{n+1}f_{n+2}f_{n-2}+f_nf_{n+2}f_{n-4}-f_nf_{n+1}f_{n-6}$

Using equation 1 with three different values

$$f_{n+2}f_{n-2}=f_n^2+(-1)^{n+1}$$
$$f_{n+2}f_{n-4}=f_nf_{n-2}+3(-1)^{n+1}$$
$$f_{n+1}f_{n-6}=f_{n-2}f_{n-3}+6(-1)^{n+1}$$

$$\begin{aligned} f_{n-2}f_{n-3} &= f_{n-2}(f_{n-1}-f_{n-2}) \\ &= f_{n-1}f_{n-2}-f_{n-2}^2 \\ &= f_{n-1}f_{n-2}-(f_{n-1}f_{n-3}+(-1)^{n+1}) \\ &= f_{n-1}f_{n-2}-f_{n-1}f_{n-3}+(-1)^n \end{aligned}$$

$f_{n+1}f_{n+2}f_{n-2}+f_nf_{n+2}f_{n-4}-f_nf_{n+1}f_{n-6}$

$$\begin{aligned} &= f_{n+1}\big(f_n^2+(-1)^{n+1}\big)+f_n\big(f_nf_{n-2}+3(-1)^{n+1}\big) \\ &\qquad -f_n\big(f_{n-2}f_{n-3}+6(-1)^{n+1}\big) \\ &= f_{n+1}f_n^2+f_n^2f_{n-2}-f_nf_{n-2}f_{n-3}+(-1)^{n+1}(f_{n+1}+3f_n-6f_n) \\ &= f_n\big[f_{n+1}f_n+f_nf_{n-2}-f_{n-2}f_{n-3}\big]+(-1)^{n+1}(f_n+f_{n-1}-3f_n) \\ &= f_n\big[(f_n+f_{n-1})f_n+(f_{n-1}^2+(-1)^{n-1})-(f_{n-1}(f_{n-2}-f_{n-3})+(-1)^n)\big] \\ &\qquad +(-1)^{n+1}(f_{n-1}-2f_n) \\ &= f_n\big[f_n^2+f_{n-1}(f_n+f_{n-1}-f_{n-2}+f_{n-3})\big] \\ &\qquad +(-1)^{n+1}(f_{n-1}-2f_n+2f_n) \\ &= f_n\big[(f_{n+1}f_{n-1}+(-1)^{n+1})+f_{n-1}(f_n+f_{n-1}-f_{n-2}+f_{n-3})\big] \\ &\qquad +(-1)^{n+1}(f_{n-1}) \\ &= f_n\big[f_{n-1}(f_{n+1}+f_n+f_{n-1}-f_{n-2}+f_{n-3})\big] \\ &\qquad +(-1)^{n+1}(f_{n-1})+(-1)^{n+1}(f_{n-1}+f_n) \\ &= f_nf_{n-1}\big[(f_n+f_{n-1})+(f_{n-1}+f_{n-2})+f_{n-1}-f_{n-2}+(f_{n-1}-f_{n-2})\big] \\ &\qquad +(-1)^{n+1}(f_{n+1}) \\ &= f_nf_{n-1}\big[(f_{n-1}+f_{n-2})+4f_{n-1}-f_{n-2}\big]+(-1)^{n+1}(f_{n+1}) \\ &= f_nf_{n-1}\big[5f_{n-1}\big]+(-1)^{n+1}f_{n+1} \\ &= 5f_nf_{n-1}f_{n-1}+(-1)^{n+1}f_{n+1} \end{aligned}$$

3. $k = 2 \;||\; f_{n+1} f_{n+2} f_{n-1} + f_n f_{n+2} f_{n-3} - f_n f_{n+1} f_{n-5}$
Using equation 1 with three different values

$$f_{n+2} f_{n-1} = f_{n+1} f_n + (-1)^n$$

$$f_{n+2} f_{n-3} = f_n f_{n-1} + 2(-1)^n$$

$$f_{n+1} f_{n-5} = f_{n-2}^2 + 4(-1)^n$$

$$\begin{aligned} f_{n+1}^2 &= f_{n+1}(f_n + f_{n-1}) \\ &= f_{n+1} f_n + fn+1 f_{n-1} \\ &= f_{n+1} f_n + f_n^2 + (-1)^n \end{aligned}$$

$$\begin{aligned} f_{n-2}^2 &= f_{n-2}(f_n - f_{n-1}) \\ &= f_n f_{n-2} - f_{n-1} f_{n-2} \\ &= f_n f_{n-2} - f_{n-1}(f_n - f_{n-1}) \\ &= f_n f_{n-2} - f_n f_{n-1} + f_{n-1}^2 \\ &= f_n f_{n-2} - f_n f_{n-1} + f_n f_{n-2} + (-1)^n \\ &= f_n(f_{n-2} - f_{n-1} + f_{n-2}) + (-1)^n \\ &= f_n(2 f_{n-2} - f_{n-1}) + (-1)^n \end{aligned}$$

$f_{n+1} f_{n+2} f_{n-1} + f_n f_{n+2} f_{n-3} - f_n f_{n+1} f_{n-5}$

$= f_{n+1}\big(f_{n+1} f_n + (-1)^n\big) + f_n\big(f_n f_{n-1} + 2(-1)^n\big)$

$$- f_n\big(f_{n-2}^2 + 4(-1)^n\big)$$

$= f_{n+1}^2 f_n + f_n^2 f_{n-1} - f_n f_{n-2}^2 + (-1)^n(f_{n+1} + 2 f_n - 4 f_n)$
$= f_n\big[f_{n+1}^2 + f_n f_{n-1} - f_{n-2}^2\big] + (-1)^n((f_n + f_{n-1}) - 2 f_n)$
$= f_n\big[(f_{n+1} f_n + f_n^2 + (-1)^n) + f_n f_{n-1} - (f_n(f_{n-2} - f_{n-1} + f_{n-2})\big]$

$$+(-1)^n) + (-1)^n(f_{n-1} - f_n)$$

$= f_n\big[f_n(f_{n+1} + f_n + f_{n-1}) - f_n(2 f_{n-2} - f_{n-1})\big]$

$$+(-1)^n(-f_{n-2}) + (-1)^n f_n + (-1)^{n+1} f_n$$

$$= f_n^2\left[f_{n+1}+f_n+f_{n-1}-2f_{n-2}+f_{n-1}\right]$$
$$+(-1)^n(-f_{n-2}+f_n-f_n)$$
$$= f_n^2\left[(f_n+f_{n-1})+f_n+f_{n-1}-(f_n-f_{n-1})+f_{n-1}-(f_n-f_{n-1})\right]$$
$$+(-1)^n(-f_{n-2})$$
$$= f_n^2\left[5f_{n-1}\right]+(-1)^{n+1}f_{n-2}$$
$$= 5f_nf_nf_{n-1}+(-1)^{n+1}f_{n-2}$$

The results of these three cases are three slightly different equations. The first term of each of the three equations are $5f_nf_{n-1}f_{n-2}$, $5f_nf_{n-1}f_{n-1}$, $5f_nf_{n-1}f_n$ respectively. This term has a nice pattern, letting $k = x \ (mod\ 3)$ the first term becomes $5f_nf_{n-1}f_{n-2+k}$. For the second term there needed to be ways to isolate certain cases so that the right constant is added to the dominant first term. Through experimentation it was possible to find that letting $l = \lceil \frac{k}{2} \rceil$ and $m = k \ (mod\ 2)$ values that helped calculate the left term. Using these variables it was possible to write the solution as one equation with the last term being $(-1)^{n+1}(-2)^{|l-1|}f_{n-1+3m-l}$. The following table checks the equations and results.

x	n	k	l	m	$2^{nd}\ term$
9	even	0	0	0	$(-1)(-2)f_{n-1+3(0)-(0)} = 2f_{n-1}$
10	even	1	1	1	$(-1)(1)f_{n-1+3(1)-(1)} = -f_{n+1}$
11	even	2	1	0	$(-1)(1)f_{n-1+3(0)-(1)} = -f_{n-2}$
12	odd	0	0	0	$(1)(-2)f_{n-1+3(0)-(0)} = -2f_{n-1}$
13	odd	1	1	1	$(1)(1)f_{n-1+3(1)-(1)} = f_{n+1}$
14	odd	2	1	0	$(1)(1)f_{n-1+3(0)-(1)} = f_{n-2}$

This shows that it works for n odd and even and with all three values of k. ■

No modulus operation is needed because the result will always be positive and will always be less then the product $f_nf_{n+1}f_{n+2}$. This is pretty easy to show. Since some small number is being added or subtracted it is only needed to insure that $5f_nf_{n-1}f_{n-2+k} < f_nf_{n+1}f_{n+2}$ or $5f_{n-1}f_{n-2+k} < f_{n+1}f_{n+2}$. Since the largest value that the left side can have with $k = 2$ is $5f_nf_{n-1}$ then it is only needed to insure that $5f_nf_{n-1}$ is less then $f_{n+1}f_{n+2}$.

$$\begin{aligned}
f_{n+2}f_{n+1} &= (f_n+f_{n-1})(2f_n+f_{n-1}) \\
&= 2f_n^2+3f_nf_{n-1}+f_{n-1}^2 \\
&= 2(f_{n+1}f_{n-1}-(-1)^n)+3f_nf_{n-1}+f_{n-1}^2 \\
&= 2f_{n+1}f_{n-1}+3f_nf_{n-1}+f_{n-1}^2-2(-1)^n \\
&= 2(f_n+f_{n-1})f_{n-1}+3f_nf_{n-1}+f_{n-1}^2-2(-1)^n \\
&= 5f_nf_{n-1}+3f_{n-1}^2-2(-1)^n
\end{aligned}$$

Plugging into the inequality yields

$$\begin{aligned} 5f_n f_{n-1} &< f_{n+1} f_{n+2} \\ &< 5f_n f_{n-1} + 3f_{n-1}^2 - 2(-1)^n \\ 0 &< 3f_{n-1}^2 - 2(-1)^n \end{aligned}$$

The right side of the inequality will always be greater then zero, since its some number squared minus at most 2. Therefore, the equation for f_x will always be less then M for a sufficient n value and a modulus operation is not needed no the result.

2.5 Bit Operations

With the reduced equation form it was possible to find, at the end of the last section, an analysis that can be done on it and a calculation of the bit operations that will be needed to find the desired Fibonacci number. As the x values get large, a large number of Fibonacci terms are needed to compute it since n will only roughly reduce x by a third. For example, to get the 10,000 term the first 3,334 terms are needed. So to reduce the number of terms which are needed to calculate the process can be repeated to calculate terms on the calculated n values, like 3,334. Also, the work can be further reduced because at each level of iteration all the terms needed can be reduced down to the two lowest terms of the given level. Then only two iterations of the process are needed for a given level. Doing it this way will reduce the number of terms needed to find logarithmically. This trick will hopefully save time and space.

2.5.1 Algorithm

I constructed a general recursive algorithm to compute the xth Fibonacci number, given a lower limit L that specifies how low the algorithm should recursively reduce down to. The iterative algorithm was slightly altered to return the nth and the $(n+1)th$ term. So a call on this algorithm will return x and $x+1$. So all that would be left is to include a program that calculated the right n value and passed $n-2$ through this algorithm. Then all that would be left to do is to calculate f_x using the returned values and a single pass through the calculation for n, k, l, m, and f_x.

fib3(x)
{
    if($x < L$)
        return(IterativeFibonacci(x));
    calc $nb = \lfloor \frac{x}{3} \rfloor + 1$;
    create array f of size 5;
    $(f_1, f_0) = fib3(nb-2)$;
    for i from 2 to 4 do
        $f_i = f_{i-1} + f_{i-2}$;
    end for;
    for i from 0 to 1 do
        $n = \lfloor \frac{x+i}{3} \rfloor + 1$;

$k = (x+i) \bmod 3$;
$l = \lceil \frac{k}{2} \rceil$;
$m = k \bmod 2$;
$t = 5 f_{n-(nb-2)} f_{n-(nb-1)} f_{n-nb+k} + (-1)^{n+l}(-2)^{|l-1|} f_{n-(nb-1)+3m-l}$;
$a = b$;
$b = t$;
end for;
return(b,a);
}

2.5.2 Result

The bit operations needed at each level for the first couple are the following. At the top level we compute only f_n from three Fibonacci numbers with index values of approximately $\frac{n}{3}$. At each lower level we compute two Fibonacci numbers with approximate index values of n and then each is computed from three Fibonacci numbers with index values of approximately $\frac{n}{3}$. Actually we need to compute three Fibonacci numbers to keep the recursion going, but the third one is computed by addition or subtraction from the two we computed using multiplication. At the top level the bit operations are approximately

$$(\frac{n}{3}\gamma n)(\frac{n}{3}\gamma n) + (\frac{2n}{3}\gamma n)(\frac{n}{3}\gamma n)$$

because we take the product of two numbers with about $\frac{n}{3}\gamma n$ bits to form a number with about $\frac{2n}{3}\gamma n$ bits and multiply that number by another $\frac{n}{3}\gamma n$. This gives $\frac{1}{3}\gamma^2 n^2$ bit operations. For each lower level we have twice as many multiplies but the size of the number is about a third of the previous. So we have for these levels

$$2\gamma^2 n^2 \left[\frac{1}{3^{2a}} + \frac{2}{3^{2a}}\right] \text{ bit operations.}$$

Which gives the generalization

$$\frac{6\gamma^2 n^2}{81}\left[1 + \frac{1}{9} + \frac{1}{9^2} + \cdots\right]$$
$$= \frac{6\gamma^2 n^2}{81} \cdot \frac{9}{8} = \frac{\gamma^2 n^2}{12}.$$

So total we get $\gamma^2 n^2 \left[\frac{1}{3} + \frac{1}{12}\right] = \frac{5}{12}\gamma^2 n^2$ big operations.

3 Two Terms

Instead of using three terms, two terms may result in a simplier, faster result. Going through the same process as the three term case. Like in the three term case, two consecutive Fibonacci terms were used, which from the three term case it was proved that they are pairwise prime.

If the same analysis is used on the M value for two terms using the rounded version of Binet's Formula, an estimation of the maximum Fibonacci number that can be calculated for a given n value.

$$f_n f_{n+1} = \left(\frac{1}{\sqrt{5}}\alpha^n\right)\left(\frac{1}{\sqrt{5}}\alpha^{n+1}\right) = \frac{1}{(\sqrt{5})^3}\alpha^{2n+1} \approx \frac{1}{\sqrt{5}}\alpha^{2n} \approx f_{2n}$$

Thus to make the procedure more efficient, the following equation was used to calculate the n value.

$$n = \lfloor \frac{x}{2} \rfloor + 1$$

3.1 Inverse Values

The reduction of the inverse values for the two term case took on a slightly different form then the three term case. They did not result in the nice ± 1 form, like the three term case. Some nice patterns were found using a Maple program that I wrote.

Theorem 3.1 *Given an integer n, let the two pairwise prime values used in the Chinese Remainder Theorem m_1 and m_2 be f_n and f_{n+1} respectively. Then if $M_i = \frac{m_1 m_2}{m_i}$ the solution to the equation $M_i y_i \equiv 1 \ (mod\ m_i)$ is*

$$(y_1, y_2) = \begin{cases} (f_{n-1}, f_{n-1}) & \text{if n is even.} \\ (f_{n-2}, f_n) & \text{if n is odd.} \end{cases}$$

So for example if $n = 8$ then $y_1 = f_7$ and $y_2 = f_7$ and if $n = 11$ then $y_1 = f_9$ and $y_2 = f_{11}$. This comes from the fact that y_1 uses the value n and y_2 uses the value $n+1$.

Proof. The two general equations

$$f_{n+1} y_1 \equiv 1 \ (mod\ f_n)$$

$$f_n y_2 \equiv 1 \ (mod\ f_{n+1}).$$

The first equation can be reduced down $f_{n+1} y_1 = (f_n + f_{n-1}) y_1 \rightarrow f_{n-1} y_1 \equiv 1\ mod\ f_n$. This form of the first equation is simply the second equation with $n-1$ substituted in for n. So if either equation can be simplified down we can generalize the result. To prove this result, the first equation was broken down into two cases, one where n is even and one where n is odd.

n even: Two basic equations known

$$1 \equiv 1 \ (mod\ f_n)$$

$$f_n^2 \equiv 0 \ (mod\ f_n).$$

It is known that from [1] that if $a \equiv b \ (mod\ m)$ and $a' \equiv b' \ (mod\ m)$, then $a + a' \equiv b + b' \ (mod\ m)$. Using this one on the basic equations yields

$$f_n^2 + 1 \equiv 0 + 1 \ (mod\ f_n).$$

Then since it assumed that n is even then it is known that

$$f_n^2 + (-1)^n \equiv 1 \ (mod \ f_n).$$

and so using Cassini's Formula, equation 3, and substituting it in yields

$$f_{n+1} f_{n-1} \equiv 1 \ (mod \ f_n).$$

From this it is noticable that this is the first equation setting $y_1 = f_{n-1}$. It also known that for y_2, $n+1$ is used so if n is odd then $n+1$ is even and so for and odd n, $y_2 = f_{(n+1)-1} = f_n$.

n odd: The following equation is the result of using equation 1, letting $r = n-2$, $n = n$ and $m = 1$, resulting in

$$f_{n+1} f_{n-2} = f_n f_{n-1} + f_1 f_2 = f_n f_{n-1} + 1.$$

Using the same trick as in the even case to find y_2. The two equations used

$$1 \equiv 1 \ (mod \ f_n)$$

$$f_n f_{n-1} \equiv 0 \ (mod \ f_n).$$

They are then added them together to get

$$f_n f_{n-1} + 1 \equiv 0 + 1 \ (mod \ f_n).$$

Then using the equation at the top of this case and substituting it in yields

$$f_{n+1} f_{n-2} \equiv 0 + 1 \ (mod \ f_n).$$

Which is the first equation again but with $y_1 = f_{n-2}$. Then observing y_2 again, if n is even then $n+1$ is odd and so for an even n, $y_2 = f_{(n+1)-2} = f_{n-1}$. ■

3.2 Remainder Values

With the use of a another program written in Maple 8 using the four general equations for finding the remainders, it was observed that the remainders were predictable.

Theorem 3.2 *Given an integer x that corresponds to a desired Fibonacci number, then let* $n = \lfloor \frac{x}{2} \rfloor + 1$ *and let* $a = x \ mod \ 4$ *then using the Chinese Remainder Theorem with two consecutive Fibonacci numbers* f_n *and* f_{n+1} *as the relatively primes,* m_i*, then the remainders are exactly*

$$< r_1, r_2 > = \begin{cases} < 1, -3 > & \text{if a = 0.} \\ < -1, 2 > & \text{if a = 1.} \\ < -1, 3 > & \text{if a = 2.} \\ < 1, -2 > & \text{if a = 3.} \end{cases}$$

Proof. Four cases need to be worked out, two cases for n even and two cases for n odd or $4i, 4i+1, 4i+2$, and $4i+3$. Just using the set of equations found for finding the remainder in the sequence should suffice in finding the goal. For all four cases $k = 1$ and $l = 0$.

case 1: $4i$

$$n_1 = 2i+1 \qquad j_1 = 4i \ (mod\ 2i+1) = 2i-1$$
$$r_1 = (-1)^{2i} f_{2i+1-(2i-1)} = f_2 = 1$$
$$n_2 = 2i+2 \qquad j_2 = 4i \ (mod\ 2i+2) = 2i-2$$
$$r_2 = (-1)^{2i-1} f_{2i+2-(2i-2)} = -f_4 = -3$$

case 2: $4i+1$

$$n_1 = 2i+1 \qquad j_1 = 4i+1 \ (mod\ 2i+1) = 2i$$
$$r_1 = (-1)^{2i+1} f_{2i+1-(2i)} = -f_1 = -1$$
$$n_2 = 2i+2 \qquad j_2 = 4i+1 \ (mod\ 2i+2) = 2i-1$$
$$r_2 = (-1)^{2i} f_{2i+2-(2i-1)} = f_3 = 2$$

case 3: $4i+2$

$$n_1 = 2i+2 \qquad j_1 = 4i+2 \ (mod\ 2i+2) = 2i$$
$$r_1 = (-1)^{2i+1} f_{2i+2-(2i)} = -f_2 = -1$$
$$n_2 = 2i+3 \qquad j_2 = 4i+2 \ (mod\ 2i+3) = 2i-1$$
$$r_2 = (-1)^{2i} f_{2i+3-(2i-1)} = f_4 = 3$$

case 4: $4i+3$

$$n_1 = 2i+2 \qquad j_1 = 4i+3 \ (mod\ 2i+2) = 2i+1$$
$$r_1 = (-1)^{2i+2} f_{2i+2-(2i+1)} = f_1 = 1$$
$$n_2 = 2i+3 \qquad j_2 = 4i+3 \ (mod\ 2i+3) = 2i$$
$$r_2 = (-1)^{2i+1} f_{2i+3-(2i)} = -f_3 = -2$$

■

3.3 Reduction of the CRT General Equation

With everything that was calculated, an attempt can be made to reduce the general equation given by the CRT. The equation reduces down nicely like in the three term case only that it only needs a single significant multiply per iteration and again no modulus operation.

Theorem 3.3 *Given an integer x that corresponds to a desired Fibonacci number, then let* $n = \lfloor \frac{x}{3} \rfloor + 1$ *then* f_x *can be determined using the following set of equations:*

$$n = \lfloor \frac{x}{2} \rfloor + 1$$

$$k = \big(x \ (mod\ 2)\big) + 1$$

$$l = \lceil \frac{x}{2} \rceil$$

$$f_x = f_n\big(2^m f_{n-m} + (-1)^k f_{n-2}\big) + (-1)^l.$$

Proof. By looking at this case the same way as the three term case, it is possible to prove the equation reduction by looking at the four remainder cases.

Each of the four remainder cases correspond to a certain value of n. For $r = < -1, 3 >$ and $< 1, -2 >$, the n value will be even. For $r = < 1, -3 >$ and $< -1, 2 >$, the n value will be odd. This even and odd will give the set of y_i values that will be used for each case.

1. $r = < -1, 3 >$ and n even $\ ||\ f_{n+1}(-1)(f_{n-1}) + f_n(3)(f_{n-1})$

$$\begin{aligned} -f_{n+1}f_{n-1} + 3f_nf_{n-1} &= -(f_n^2 + (-1)^n) + 3f_nf_{n-1} \\ &= f_n(3f_{n-1} - f_n) - 1 \\ &= f_n(2f_{n-1} - f_{n-2}) - 1 \end{aligned}$$

2. $r = < 1, -2 >$ and n even $\ ||\ f_{n+1}(1)(f_{n-1}) + f_n(-2)(f_{n-1})$

$$\begin{aligned} f_{n+1}f_{n-1} - 2f_nf_{n-1} &= f_n^2 + (-1)^n - 2f_nf_{n-1} \\ &= f_n(f_n - 2f_{n-1}) + 1 \\ &= f_n(f_{n+1} + f_n - 2f_{n-1}) + 1 \\ &= f_n(f_{n+1} - f_{n-1} + f_{n-2}) + 1 \\ &= f_n(f_n + f_{n-2}) + 1 \end{aligned}$$

The term inside the parentheses of the third line in the above work is obviously negative and so $f_{n+1}f_n$ is added to the whole thing to make it positive, since the equation is modulus $f_{n+1}f_n$.

The following is an equation that will be used throughout the next two parts

$$f_{n+1}f_{n-2} = f_nf_{n-1} + 1.$$

3. $r = < 1, -3 >$ and n odd $\;||\; f_{n+1}(1)(f_{n-2}) + f_n(-3)(f_n)$

$$\begin{aligned} f_{n+1}f_{n-2} - 3f_n^2 &= f_n f_{n-1} + 1 - 3f_n^2 \\ &= f_n(f_{n-1} - 3f_n) + 1 \\ &= f_n(2f_{n+1} + f_{n-1} - 3f_n) + 1 \\ &= f_n(2f_n + 2f_{n-1} + f_{n-1} - 3f_n) + 1 \\ &= f_n(3f_{n-1} - f_n) + 1 \\ &= f_n(2f_{n-1} - f_{n-2}) + 1 \end{aligned}$$

The term inside the parentheses of the third line in the above work is also negative and so $2f_{n+1}f_n$ is added to the whole thing for the same reasoning.

4. $r = < -1, 2 >$ and n odd $\;||\; f_{n+1}(-1)(f_{n-2}) + f_n(2)(f_n)$

$$\begin{aligned} -f_{n+1}f_{n-2} + 2f_n^2 &= -f_n f_{n-1} - 1 + 2f_n^2 \\ &= f_n(2f_n - f_{n-1}) - 1 \\ &= f_n(f_n + f_{n-2}) - 1 \end{aligned}$$

■

As in the three term case, the two term case will require no modulus operation. The two cases we need to check are if $f_n f_{n+1} > f_n(f_n - f_{n-4})$ and if $f_n f_{n+1} > f_n(f_n + f_{n-2})$, ignoring the ± 1. This is easy to show after dividing by f_n. The equations reduce down to

$$f_{n+1} > f_n - f_{n-4} \rightarrow \; f_n + f_{n-1} > f_n - f_{n-4} \rightarrow \; f_{n-1} > -f_{n-4}$$

$$f_{n+1} > f_n + f_{n-2} \rightarrow \; f_n + f_{n-1} > f_n + f_{n-2} \rightarrow \; f_{n-1} > f_{n-2}.$$

So it is now known that the equation for f_x for two terms will always be positive and less then M.

3.4 Bit Operations

The same problem arises as in the three term case. It will actually be slightly more significant since for two terms the x value will only be reduced by a half. To improve the results again, the procedure will be called recursively on all previous terms. This will cut the number of terms logarithmically again.

3.4.1 Algorithm

The same information applies to this algorithm as did in the fib3 algorithm with L being the lower limit.

fib2(x)
{
    if($x < L$)
        return(IterativeFibonacci(x));
    calc $nb = \lfloor \frac{x}{2} \rfloor + 1$;
    create array f of size 5;
    $(f_1, f_0) = fib2(nb-2)$;
    for i from 2 to 4 do
        $f_i = f_{i-1} + f_{i-2}$;
    end for;
    for i from 0 to 1 do
        $n = \lfloor \frac{x+i}{2} \rfloor + 1$;
        $k = ((x+i) \ mod \ 2) + 1$;
        $l = \lceil \frac{x+i}{2} \rceil$;
        $m = (x+i+1) \ (mod \ 2)$;
        $t = f_{n-(nb-2)}(2^m f_{n-(nb-2)-m} + (-1)^k f_{n-nb}) + (-1)^l$;
        $a = b$;
        $b = t$;
    end for;
    $return(b, a)$;
}

3.4.2 Results

This calculation mimics the three term example that was worked out. The bit operations followed a similar trend in the two term case as they did for the three term case. There was a nice pattern that could be reduced down to a sum of a geometric series. The number of bit operations turned out to be

$$\Big[\Big(\frac{1}{2}\sum_{k=0}^{\infty}(\frac{1}{4})^k\Big) - \frac{1}{4}\Big]\gamma^2 n^2 = \frac{5}{12}\gamma^2 n^2.$$

4 Four Terms

Using four consecutive Fibonacci terms proved to be more complicated then two and three terms.

The first problem to arise was finding four consecutive Fibonacci numbers for a given n value. Just to be thorough each case will be listed.

$$GCD(f_n, f_{n+1}) = f_{GCD(n,n+1)} \rightarrow f_1 = 1$$

$$GCD(f_n, f_{n+2}) = f_{GCD(n,n+2)} \rightarrow f_1 or f_2 = 1 or 1$$
$$GCD(f_n, f_{n+3}) = f_{GCD(n,n+3)} \rightarrow f_1 or f_2 or f_3 = 1 or 1 or 2$$

So if the n is a multiple of 3 then there will be a problem because the first and the last term will have a common factor of 2. A way to avoid this is to insure that the n value will never be an exact multiple of 3 the following equation was used

$$n = \lfloor \frac{x}{4} \rfloor + 1$$

$$n = n + 2 - (n \ (mod\ 3))$$

These equations will insure that n will always be equal to $3i+2$ for some $i = 0, 1, 2, \ldots$.

4.1 Inverse Values

Knowing that n will be equal to $3i+2$ a reduced form of the inverse equations using a Maple program yielded these observable patterns. These equations have yet to be formally proven.

Proposition 4.1 *Given an $n = 3i+2$ for some $i = 0, 1, 2, \ldots$, let the four pairwise prime values used in the Chinese Remainder Theorem m_1, m_2, m_3, and m_4 be f_n, f_{n+1}, f_{n+2}, and f_{n+3}respectively. Then if $M_i = \frac{m_1 m_2 m_3 m_4}{m_i}$ the solution to the equation $M_i y_i \equiv 1 \ (mod\ m_i)$ is*

$$y_1 = y_4 = \frac{1}{2} f_{n+1}$$
$$y_2 = y_3 = f_n.$$

Proof. Currently the only proof of this is running the Maple program using a large number of input examples. ■

4.2 Remainder Values

While knowing that $n = 3i+2$ for some $i = 0, 1, 2, \ldots$, using another Maple program a repeating pattern emerged after taking values of $x \geq 80$. The following table lists the pattern.

x	*remainders*	x	*remainders*
80	$-f_{12}, -f_{16}, -f_{20}, -f_{24}$	86	$-f_6, -f_{10}, -f_{14}, -f_{18}$
81	$f_{11}, f_{15}, f_{19}, f_{23}$	87	f_5, f_9, f_{13}, f_{17}
82	$-f_{10}, -f_{14}, -f_{18}, -f_{22}$	88	$-f_4, -f_8, -f_{12}, -f_{16}$
83	$f_9, f_{13}, f_{17}, f_{21}$	89	f_3, f_7, f_{11}, f_{15}
84	$-f_8, -f_{12}, -f_{16}, -f_{20}$	90	$-f_2, -f_6, -f_{10}, -f_{14}$
85	$f_7, f_{11}, f_{15}, f_{19}$	91	f_1, f_5, f_9, f_{13}

As tested this pattern holds and thus the remainders can be treated as constants and will not contribute significant multiplies when calculating the number of bit operations that are needed to find a particular f_x.

4.2.1 Results

A reduced form of the general equation that required only three significant multiplications was not found.

One problem that I can see with using four consecutive terms is the fact that the number of bit operations needed for a single iteration will start out too large. Assuming that given an equation which only requires the multiplication of four $\left(\frac{1}{4}\right)\gamma$ bit numbers and that there are no other operations that will add any significant amount to the bit operations, the number of bit operations needed for a single iteration is $\frac{3}{8}\gamma^2 n^2$. This calculation below mimics the three term example that was worked out. It contains the calculations but omits the explanation.

5 Timing

Using Maple, I wrote programs for all the algorithms listed in this paper, the iterative, the two term and the three term methods. I also wrote a program that would time the running of each program given various sizes of Fibonacci numbers they were to compute. The timing program would run a given method a number of times on a particular input size and then average the times that were calculated. It would then repeat the process with the next input size. I chose to vary the input size two ways, the first was a uniform step size of 1,000 and the second was an exponential input increase by powers of two. Data for the exponential timing is included to give perspective. It is apparent that the iterative method is much slower after a sufficiently large index value.

Results of timing programs with exponential step sizes (2^{Exp}).

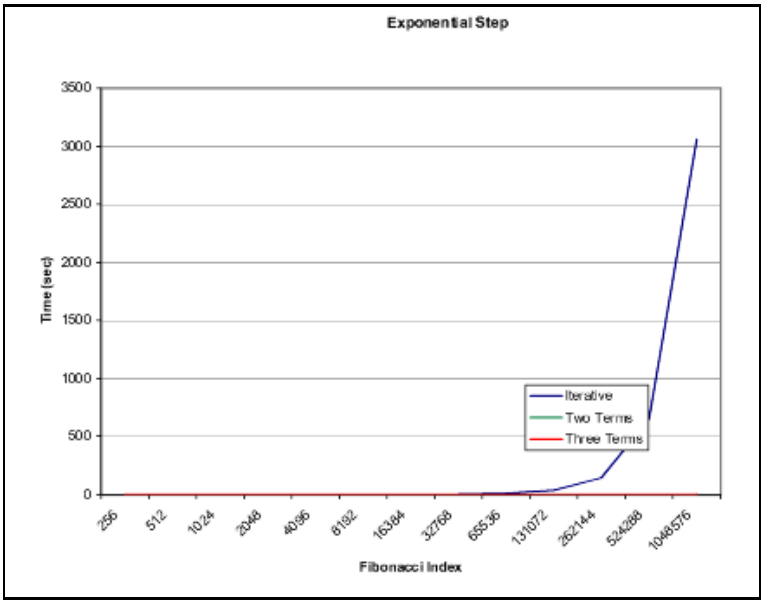
Exp	$FibIt$	$Fib2$	$Fib3$
8	0.002085	0.002435	0.00179
9	0.005105	0.00303	0.0022
10	0.01223	0.003745	0.002585
11	0.0402	0.00451	0.003337
12	0.106615	0.005445	0.00405
13	0.261125	0.007345	0.005945
14	0.778185	0.011085	0.01085
15	2.624305	0.020575	0.02379
16	9.53832	0.04708	0.061565
17	36.428915	0.124955	0.17064
18	141.515	0.347255	0.50022
19	647.397	1.01283	1.47754
20	3060.648	3.0075	4.4224

It looks as though the two term method is actually faster then the three term method. I was going to attempt to program another method which used Lucas numbers to calculate Fibonacci numbers and compare that one also. [5] I ran out of time and could not get the program to work correctly.

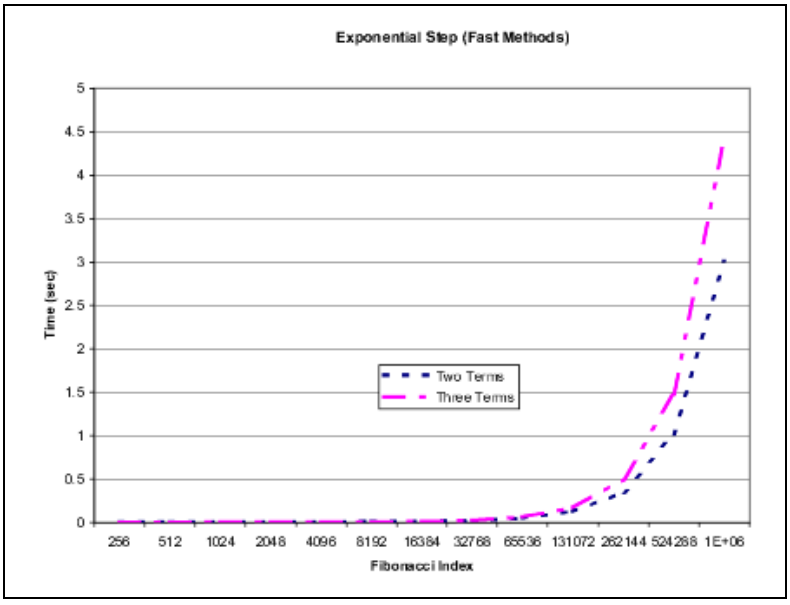
If the visible trend continues then other methods like four or five consecutive terms may not result in quicker terms. Unless these terms are able to be reduced in some manner to reduce the number of bit operations altogether.

One surprising observation is that the two and three term methods seem to increase by a multiple of three and not the expected multiple of 4. Since these are n^2 methods, doubling the input size each time should increase the running time by about a factor of 4 each time. The multiple of three may come from the fact that I used Maple and it has some ways to reduce the time needed. One way to check this would be to try and program the algorithms in another language and time them again.

At the end of this section is two graphs, one with all three data sets and one with only the two term and three term methods. In the first graph, the sharply increasing line is the iterative method and the other two methods hang around the x-axis. The second graph compares the times for the two term and three term methods. It seems as though the three term is generally slower then the two term method. In both graphs, the x-axis is labelled by the Fibonacci index that was computed and the y-axis is labelled with the time in seconds.



Comparison of the running times for the Iterative, Two Term and Three Term Methods for computing a given Fibonacci index



Comparison of the running times for the Two Term and Three Term Methods for computing a given Fibonacci index

6 Conclusion

With the investigation of the Chinese Remainder Theorem, I had hoped that it would result in methods that would beat fastest known methods. In the end, both the two and three term methods resulted in nice equations that did not require the CRT. They were both only dependent on knowing a number of lower Fibonacci numbers and being able to calculate some minor equations and then the final calculation for the desired Fibonacci number.

Unfortunately the bit operations seemed to come out in both cases as $\left(\frac{5}{12}\right)\gamma^2 n^2$. This was the same result that Cull and Holloway[5] found for their method of calculation. Another problem is the results in another paper[6] by Daisuke Takahashi that claims that he had found a way to speed up the method in Cull and Holloway. So one thing to check is to attempt to program his method and compare running times that way.

Another point that could be investigated is seeing how changing the lower limit, L, in the algorithms affect the running times. It may be worth while to find a way to optimize that as well. A simple comparison of when the iterative method performs slower then the other methods would yield a better lower limit then the one that I arbitrarily chose.

It would also be interesting to get the Lucas method working and to compare running times. Since the bit operations come out to be the same in the end, finding out which runs faster in practice would be an interesting outcome.

Obviously the four term section can use a lot of work to actually see if my observations are true. There are a number of other ways to reduce the general equation and maybe with some luck a nice result can be found.

Then work on five consecutive terms could be attempted. The way one calculates the n value will need to be very specific since only one out of every three values will work. Five terms may yield very nicely reduced forms for the remainders and inverses.

Instead of looking at consecutive terms, finding clever ways to compute non-consecutive terms may result in faster methods of computation. I have not thought about ways of choosing them as of yet but one must consider ways which will yield convenient reductions or nice simplifications. The problem with five terms is the same as four. Unless the size of the multiplications can be reduced it seems as though it will start out too slow to end up being any faster.

This investigation did produce some nice results. It was surprising that the bit operations for both methods resulted in the same coefficient of $\frac{5}{12}$. Even more surprising was that $\frac{5}{12}$ was the same coefficient found by Cull and Holloway[5], using a completely separate method for computation. This coefficient may come from the basic approach that all n^2 methods use. It would be interesting to see if other variations of the CRT would result in the same coefficient.

7 References

1. Vanden Eynden, Charles. *Elementary Number Theory second edition*. The McGraw-Hill Companies Inc. New York, New York, 2001.

2. Knuth, Donald. *The Art of Computer Programming* (*Vol*. 1). Addison-Wesley, Reading, Massachusetts, 1975.

3. Vorob'ev, N.N. *Fibonacci Numbers*. Pergamon Press, New York, New York, 1961.

4. Cull, Paul; Flahive, Mary; Robson, Robby. *Difference Equations: From Rabbits to Chaos.* "Chapter 8: Modular Recurrences". Springer-Verlag, 2003. (To appear)

5. Cull, Paul and Holloway, James L. *Computing Fibonacci Numbers Quickly*. Information Processing Letter 32, 1989.

6. Takahashi, Daisuke. *A fast algorithm for computing large Fibonacci numbers*. Information Processing Letter 75, 2000.

7. Koshy, Thomas. *Fibonacci Lucas Numbers with Applications*. John Wiley & Sons, Inc., New York, New York, 2001.

8. Babb's paper. (Copy gotten from Paul Cull).

9. *Chinese Remainder Theorem*. [website] http://mathworld.wolfram.com/.

10. *Chinese Remainder Theorem*. [website] http://www.wikipedia.org/wiki/.