Exploring Basic Operations in the Fibonacci Number System

A PROJECT REPORT

Submitted By

Arkasnato Bhattacharyya, Roll No: 12622001032, Reg No: 221260110262

Himadri Chatterjee, Roll No: 12622001069, Reg No: 221260110299

Sayan Kundu, Roll No: 12622001134, Reg No: 221260110370

Shirsa Maitra, Roll No: 12622001139, Reg No: 221260110375

Under the supervision of

Prof. (Dr.) Subhashis Majumder,
Department of Computer Science & Engineering
Heritage Institute of Technology, Kolkata

&

Prof. (Dr.) Somjit Datta
Department of Mathematics
Heritage Institute of Technology, Kolkata

In partial fulfillment of the award of the degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING,

HERITAGE INSTITUTE OF TECHNOLOGY, KOLKATA

(An Autonomous Institute under Maulana Abdul Kalam Azad University of Technology)

December, 2025

ACKNOWLEDGEMENT

We would take this opportunity to thank **Prof. (Dr.) Basab Chaudhuri**, Principal Heritage Institute of Technology, for providing us with all the necessary facilities to make our project work successful.

We would like to thank our Head of the Department, **Prof. (Dr.) Subhashis Majumder** for his kind assistance as and when required.

We will be thankful to **Prof. (Dr.) Subhashis Majumder** and **Prof. (Dr.) Somjit Datta**, our project coordinators, for constantly supporting and guiding us for giving us invaluable insights. His guidance and encouragement motivated us to achieve our goal and impetus to excel.

We thank our faculty members and laboratory assistants at the Heritage Institute of Technology for paying a pivotal and decisive role during the project's development. Last but not least, we thank all our friends for their cooperation and encouragement that they bestowed on us.

BONAFIDE CERTIFICATE

Certified that this project report titled ***"Exploring Basic Operations in the Fibonacci Number System"*** is the Bonafide work of **Arkasnato Bhattacharyya, Himadri Chatterjee, Sayan Kundu** and **Shirsa Maitra**, who carried out the project under my/our supervision.

Signature and Date,
Prof. (Dr.) Subhashis Majumder,
Project Guide,
Department of Computer Science & Engineering,
Heritage Institute of Technology, Kolkata

Signature and Date,
Prof. (Dr.) Somjit Datta,
Project Guide,
Department of Mathematics,
Heritage Institute of Technology, Kolkata

Signature and Date,
Prof. (Dr.) Subhashis Majumder,
Professor and HoD,
Department of Computer Science & Engineering
Dean of UG Programme,
Heritage Institute of Technology, Kolkata

Signature and Date,
Examiner

INDEX

Contents

Chapter 1

Arithmetic Operations in the Fibonacci Number System

1 Problem Statement

The central problem this report addresses is the following :

> **Explain how to add positive integers, working entirely in the Fibonacci number system[1].**

To answer this, we must first define the system itself and then derive the unique arithmetic rules it necessitates.

2 Background and Definitions

2.1 The Fibonacci Sequence

The Fibonacci sequence, $\langle F_n \rangle$, is a sequence of integers defined by the linear recurrence relation [7]:

$$F_n = F_{n-1} + F_{n-2}, \quad \text{for } n > 1 \tag{1}$$

with the initial conditions $F_0 = 0$ and $F_1 = 1$ [7]. The sequence begins:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, ...$$

The ratio of consecutive terms $\frac{F_{n+1}}{F_n}$ approaches the golden ratio, Φ, as $n \to \infty$ [7]:

$$\Phi = \frac{1+\sqrt{5}}{2} \approx 1.61803... \tag{2}$$

2.2 Zeckendorf's Theorem and the Fibonacci Number System

While numbers can be represented as sums of Fibonacci numbers in many ways, Zeckendorf's theorem provides a unique standard.

Theorem 1 (Zeckendorf's Theorem)**.** *Every positive integer n has a unique representation as a sum of non-consecutive Fibonacci numbers [2, 7]:*

$$n = F_{k_1} + F_{k_2} + \cdots + F_{k_r}, \quad \textit{where } k_i \geq k_{i+1} + 2 \textit{ for all } i \tag{3}$$

This theorem is the foundation of the Fibonacci number system [7]. We can represent any non-negative integer n as a binary string, or 'fibit' string, $(b_m b_{m-1} ... b_2)_F$, where:

$$n = \sum_{k=2}^{m} b_k F_k \tag{4}$$

This representation is a valid Zeckendorf representation if and only if it contains no adjacent 1s (i.e., $b_k b_{k+1} = 0$ for all k) [2, 7].

Table 1: Zeckendorf Representations (1-10) [7]

n	Sum	Representation $(...F_5F_4F_3F_2)_F$
1	F_2	$(1)_F$
2	F_3	$(10)_F$
3	F_4	$(100)_F$
4	$F_4 + F_2$	$(101)_F$
5	F_5	$(1000)_F$
6	$F_5 + F_2$	$(1001)_F$
7	$F_5 + F_3$	$(1010)_F$
8	F_6	$(10000)_F$
9	$F_6 + F_2$	$(10001)_F$
10	$F_6 + F_3$	$(10010)_F$

3 Literature Review on Fibonacci Arithmetic

Standard arithmetic algorithms are not directly applicable to Zeckendorf representations. The field of arithmetic on this base was explored in detail by Donald E. Knuth, who defined a novel multiplication operation [2].

Definition 1 (Knuth's "Circle" Multiplication)**.** *Given two integers m and n with their Zeckendorf representations:*

$$m = \sum_{b=1}^{q} F_{j_b} \quad \text{and} \quad n = \sum_{c=1}^{r} F_{k_c}$$

This representation follows the Zeckendorf conditions, where the relation $j_b \gg j_{b-1}$ means $j_b \geq j_{b-1}+2$ [2]. In Knuth's paper, this is also formalized using a radix-F notation, $(d_s...d_0)_F$, which is a valid Zeckendorf representation if it satisfies three conditions [2]:

- ***Z1:*** *Each digit d_i is 0 or 1 [2].*
- ***Z2:*** *No two adjacent digits are 1 (i.e., $d_i d_{i+1} = 0$) [2].*
- ***Z3:*** *$d_1 = d_0 = 0$ (meaning F_1 and F_0 are not used) [2].*

The "circle multiplication" $m \circ n$ is defined as [2, 3]:

$$m \circ n = \sum_{b=1}^{q} \sum_{c=1}^{r} F_{j_b + k_c} \tag{5}$$

*This definition is built on the base-case $F_j \circ F_k = F_{j+k}$, which holds **if** $j \geq 2$ **and** $k \geq 2$ [2]. This aligns with condition Z3.*

Theorem 2 (Knuth, Associativity of $\circ$)**.** *The circle multiplication operation is associative [2]:*

$$(l \circ m) \circ n = l \circ (m \circ n)$$

Proof. (Sketch) Knuth proves this by showing that both sides of the equation are equal to the symmetric three-fold sum [2]:

$$\sum_{a=1}^{p}\sum_{b=1}^{q}\sum_{c=1}^{r} F_{i_a+j_b+k_c} \tag{6}$$

□

Remark 1. *This operation $m \circ n$ is approximately $\sqrt{5}mn$ for large m and n [2]. Arnoux noted that this multiplication can be interpreted as the usual multiplication in a multiplicatively closed subset of the ring of algebraic integers $\mathbb{Z}[\Phi]$ [3].*

The result of the sum in Knuth's definition (Equation 5) is not guaranteed to be a valid Zeckendorf representation. It is merely a sum of Fibonacci numbers. To find the final, unique representation of the product, one must perform Fibonacci addition, demonstrating that **addition is the fundamental and non-trivial operation** required to work in this system.

4 Methodology: Extracting the Addition Sub-Problem

As demonstrated in the "Literature Review" (Section 3), a high-level operation like Knuth's "circle multiplication" is defined by its generation of partial products [2, 6]. In this context, addition emerges as the process that merges and normalizes these partial products.

This "summing" step is the fundamental, embedded challenge. A simple bitwise sum of the partial products does not produce a valid Zeckendorf representation; it creates an intermediate string with invalid digits (like `2`) and adjacent `1`s [4].

Therefore, to solve the multiplication problem, we must first extract and formally solve the addition problem. We now address the report's central question [1] by deriving this methodology. The process consists of two main stages [4, 8]: a naive bitwise sum and a critical normalization phase.

4.1 Stage 1: Naive Bitwise Sum

First, the two "fibit" strings are aligned by their place-values ($F_2, F_3, ...$) and added column-by-column, just as in standard binary addition, but **without carrying** [4, 8].

$$(a_m...a_2)_F + (b_m...b_2)_F = (s_m...s_2)_S$$

where $s_k = a_k + b_k$. This "raw sum" string S is not a valid Zeckendorf representation because its digits s_k can be `0`, `1`, or `2`, and it may now contain adjacent 1s [4].

4.2 Stage 2: Normalization via Carry Rules

To convert the raw sum S back into a valid Zeckendorf representation, we must "normalize" it by repeatedly applying rules that eliminate all `2`s and `11`s. These rules are direct applications of Fibonacci identities [4, 5, 7, 8].

4.2.1 Normalization Identity 1: The "11" Carry

This rule handles adjacent 1s.

- **Identity:** $F_i + F_{i+1} = F_{i+2}$ [5, 7].
- **Algorithmic Rule:** A pattern of `...011...` in positions i and $i+1$ is replaced by `...100...`. This is a standard "carry-left" operation [5, 7].

  Note: This operation can be recursive; the newly created 1 at position $i+2$ may form a new adjacent pair with an existing 1 at position $i+3$, creating a new "11" pattern that must be subsequently resolved.

4.2.2 Normalization Identity 2: The "2" Carry

This rule handles a `2` in any position, which results from $F_i + F_i$.

- **Identity:** $2F_i = F_i + F_i = F_{i+1} + F_{i-2}$ (for $i \geq 4$). (For $i = 3$: $2F_3 = 4 = F_4 + F_1$. For $i = 2$: $2F_2 = 2 = F_3 + F_0 = F_3$) [4].
- **Algorithmic Rule:** A `2` in position i is replaced by a `0`. Then, a `1` is added to position $i+1$ (a "carry-left") and a `1` is added to position $i-2$ (a "carry-right-by-two") [4]. This two-way carry is a key feature of Fibonacci addition.

This process must be repeated in "passes" or "sweeps" until no `2`s or `11`s remain. The process is guaranteed to terminate, and the final result will be the unique Zeckendorf representation of the sum [4, 8].

4.3 Detailed Example: Addition of $7 + 6$

We will compute the sum of $n = 7$ and $m = 6$. The expected result is $13 = (1000000)_F = F_7$.

- $7 = 5 + 2 = F_5 + F_3 = (1010)_F$
- $6 = 5 + 1 = F_5 + F_2 = (1001)_F$

We align the numbers by place-value (padded with leading zeros for clarity):

Place-Value:	F_7	F_6	F_5	F_4	F_3	F_2
(Value):	(13)	(8)	(5)	(3)	(2)	(1)
A (7)	0	0	1	0	1	0
B (6)	+ 0	0	1	0	0	1
Raw Sum (S)	**0**	**0**	**2**	**0**	**1**	**1**

Now, we normalize the raw sum $S = (002011)_F$ by applying the rules.

1. **Pass 1:** Scan S for invalid patterns.

- $S = (00201\underline{11})_F$. We find a `11` at F_3 and F_2.
- Apply Rule 1 ($F_3 + F_2 = F_4$): `...011` $\rightarrow$ `...100`.
- S becomes $(002\underline{1}00)_F$.

2. **Pass 2:** Scan the new S.
   - $S = (00\underline{2}100)_F$. We find a `2` at F_5.
   - Apply Rule 2 ($2F_5 = F_6 + F_{5-2} = F_6 + F_3$):
   - Set $F_5 \rightarrow 0$. Add 1 to F_6. Add 1 to F_3.
   - S becomes $(0\underline{1}01\underline{1}00)_F$.
3. **Pass 3:** Scan the new S.
   - $S = (0101\underline{11}00)_F$. We find a `11` at F_4 and F_3.
   - Apply Rule 1 ($F_4 + F_3 = F_5$): `...011...` $\rightarrow$ `...100...`.
   - S becomes $(01\underline{1}0000)_F$.
4. **Pass 4:** Scan the new S.
   - $S = (0\underline{11}0000)_F$. We find a `11` at F_6 and F_5.
   - Apply Rule 1 ($F_6 + F_5 = F_7$): `...011...` $\rightarrow$ `...100...`.
   - S becomes $(\underline{1}000000)_F$.
5. **Pass 5:** Scan the new S.
   - $S = (1000000)_F$.
   - The string contains no `2`s and no `11`s.
   - Normalization is complete.

The final result is $(1000000)_F$, which represents $F_7 = 13$. The addition $7 + 6 = 13$ is correct.

5 Algorithmic Implementation of Addition

The normalization process can be formalized into an algorithm. The following pseudocode implements the normalization loops until a stable, valid Zeckendorf representation is achieved [4].

Algorithm 1 Fibonacci Addition (Normalization)

procedure FIBONACCIADD($bitsA, bitsB$)
  $n \leftarrow \max(\text{len}(bitsA), \text{len}(bitsB))$
  $S \leftarrow$ array of zeros of length $n + 3$ ▷ Extra space for carries
  ▷ Stage 1: Naive Bitwise Sum
  for $i \leftarrow 0$ to $n - 1$ **do**
    $S[i] \leftarrow (\text{bitsA}[i] \text{ if } i < \text{len}(bitsA) \text{ else } 0) + (\text{bitsB}[i] \text{ if } i < \text{len}(bitsB) \text{ else } 0)$
  end for
  ▷ Stage 2: Normalization Loop
  $changed \leftarrow$ True
  while *changed* **do**
    $changed \leftarrow$ False
    ▷ Apply Rule 2: Eliminate 2s ($2F_i \rightarrow F_{i+1} + F_{i-2}$)
    for $i \leftarrow 0$ to $\text{len}(S) - 1$ **do**
      if $S[i] \geq 2$ **then**
        $carry \leftarrow S[i]$ // 2
        $S[i] \leftarrow S[i]$ % 2
        $S[i + 1] \leftarrow S[i + 1] + carry$ ▷ Carry left
        if $i \geq 2$ **then**
          $S[i - 2] \leftarrow S[i - 2] + carry$ ▷ Carry right by two
        else if $i == 1$ **then** ▷ Handle $2F_2 = F_3$
          $S[i + 1] \leftarrow S[i + 1] + carry$ ▷ Special case: $2F_2 = 2 = F_3$
        else ▷ Handle $i = 0$ (if F_1 is used)
          $S[i + 1] \leftarrow S[i + 1] + carry$
        end if
        $changed \leftarrow$ True
      end if
    end for
    ▷ Apply Rule 1: Eliminate 11s ($F_i + F_{i+1} \rightarrow F_{i+2}$)
    for $i \leftarrow 0$ to $\text{len}(S) - 2$ **do**
      if $S[i] == 1$ and $S[i + 1] == 1$ **then**
        $S[i] \leftarrow 0$
        $S[i + 1] \leftarrow 0$
        $S[i + 2] \leftarrow S[i + 2] + 1$ ▷ Carry left by two
        $changed \leftarrow$ True
      end if
    end for
  end while
  ▷ Clean up leading zeros
  while $\text{len}(S) > 1$ and $S[\text{len}(S) - 1] == 0$ **do**
    S.pop()
  end while
  return S
end procedure

6 Appendix: Integer-to-Zeckendorf Conversion

To test the FibonacciAdd algorithm and to create its inputs, it is necessary to convert standard integers into their Zeckendorf representation. The following algorithm implements this conversion using the standard "greedy" approach, selecting the largest Fibonacci number less than or equal to the remaining value.

Algorithm 2 Integer to Zeckendorf Bits (Greedy Algorithm)

procedure IntegerToZeckendorf(n)
  if $n \leq 0$ **then**
    raise ValueError("n must be a positive integer")
  end if
  ▷ Generate Fibonacci numbers up to n (starting $F_2 = 1, F_3 = 2$)
  $fib \leftarrow [1, 2]$
  while fib[last] $\leq n$ **do**
    fib.append(fib[last] + fib[second-to-last])
  end while
  $remaining \leftarrow n$
  $used \leftarrow$ array of zeros of length len(fib)
  $i \leftarrow$ len(fib) $- 1$
  if $fib[i] > remaining$ **then**
    $i \leftarrow i - 1$
  end if
  ▷ Greedy selection process
  while $remaining > 0$ and $i \geq 0$ **do**
    if $fib[i] \leq remaining$ **then**
      $used[i] \leftarrow 1$
      $remaining \leftarrow remaining - fib[i]$
      $i \leftarrow i - 2$ ▷ Skip next number to avoid consecutive 1s
    else
      $i \leftarrow i - 1$
    end if
  end while
  ▷ Trim unused leading zeros (from F_n down)
  while len($used$) > 1 and $used$[last] $== 0$ **do**
    $used$.pop()
  end while
  ▷ Return as string, reversing for MSB-first
  $bitString \leftarrow$ join(reversed($used$))
  return $bitString$
end procedure

7 Mathematical Properties of Circle Multiplication

Having established the fundamental addition algorithm, we can now analyze the properties of the higher-order operation defined by Knuth: circle multiplication ($m \circ n$). The robustness of this system relies on two key characteristics: the stability of its partial products and the uniqueness of its results.

7.1 Partial Products and the "Clean" Lemma

In the Fibonacci number system, "partial products" are the intermediate terms generated when the multiplication is distributed over the Zeckendorf components of the operands. Given $n = \sum F_{k_i}$, the circle product $m \circ n$ expands to:

$$m \circ n = \sum (m \circ F_{k_i}) \tag{7}$$

Each term $(m \circ F_{k_i})$ represents the multiplier m with its Fibonacci indices shifted by k_i. Knuth demonstrated a critical property known as the **"Clean" Lemma** [2]. He proved that these partial products are spaced sufficiently far apart that their summation is "clean." This means that the carry operations (normalization) from one partial product do not trigger a cascading chain reaction that destabilizes the others, ensuring that the operation is associative.

7.2 Uniqueness and Strict Monotonicity

A fundamental requirement for any arithmetic system is that the result of an operation must be unique. Based on Knuth's analysis [2], circle multiplication satisfies this condition for all positive integers.

Theorem 3 (Uniqueness). *Two distinct positive integers n_1 and n_2 cannot produce the same result when circle-multiplied by the same non-zero multiplier m.*

$$\text{If } n_1 \neq n_2 \text{ and } m > 0, \text{ then } m \circ n_1 \neq m \circ n_2. \tag{8}$$

Proof. The justification rests on the property of **Strict Monotonicity**. As noted by Henderson, the circle product is strictly increasing with respect to its operands [9]. Since the operation is defined by the summation of positive Fibonacci terms (F_{j+k}), increasing the value of an operand strictly increases the indices or count of the resulting sum.

$$n_1 < n_2 \implies m \circ n_1 < m \circ n_2$$

Because the function is strictly increasing, it is injective (one-to-one). It never "loops back" to map distinct inputs to the same output.

7.3 The Cancellation Law

As a direct consequence of strict monotonicity, the set of positive integers under circle multiplication satisfies the **Cancellation Law**. This allows us to effectively "cancel" a common factor from both sides of an equation, a property essential for algebraic consistency [9].

$$\text{If } m \circ n_1 = m \circ n_2 \text{ and } m \neq 0, \text{ then } n_1 = n_2.$$

Note: The only exception is the zero element. Since $0 \circ n = 0$ for all n, the cancellation law does not hold if $m = 0$.

8 Inverse Operations: "Egyptian" Circular Division

The properties of commutativity ($m \circ n = n \circ m$) and strict monotonicity allow us to define an inverse operation. Given a known factor m and a product P, we can retrieve the unknown factor n such that $m \circ n = P$. This process is often referred to as "Circular Division" or **"Egyptian Division"** due to its reliance on a greedy subtraction method similar to ancient Egyptian arithmetic.

8.1 Methodology

Since the operation is strictly monotonic, we can reconstruct the unknown operand n by identifying its Fibonacci components one by one, starting from the largest.

- **Basis Generation:** We calculate "shifted" versions of the known operand m (i.e., $m \circ F_k$) for increasing indices $k = 2, 3, \ldots$ until the value exceeds the product P.
- **Greedy Selection:** We locate the largest index k such that the shifted value fits within the current remainder ($m \circ F_k \leq P_{rem}$).

  Note: P_{rem} refers to the running total in this loop operation. In the first pass, P_{rem} is initialized to the full product P. In successive iterations, it decreases as components are found and subtracted.
- **Reduction:** We subtract this value from the current remainder ($P_{rem} \leftarrow P_{rem} - m \circ F_k$) and repeat the process until the remainder is zero.

8.2 Algorithm: Inverse Circle Multiplication

The following algorithm implements this reconstruction. It solves for n given m and P. Note that due to commutativity ($m \circ n = n \circ m$), this same algorithm works regardless of which operand is unknown.

Algorithm 3 Inverse Circular Multiplication (Egyptian Division)

```
procedure InverseCircleMultiply(knownFactor, product)
    if knownFactor == 0 then
        return Error("Division by zero")
    end if
    if product == 0 then
        return 0
    end if
                                        ▷ Step 1: Decompose knownFactor into Fibonacci indices
    indices ← ZeckendorfDecomposition(knownFactor)
                                    ▷ Step 2: Generate Basis Table (shifted values of knownFactor)
                                    ▷ Calculate V[k] = knownFactor ∘ F_k until V[k] > product
    basisTable ← Map()
    k ← 2
    loop
        val ← 0
        for each idx in indices do
            val ← val + Fibonacci(idx + k)          ▷ Definition: F_idx ∘ F_k = F_{idx+k}
        end for
        if val > product then
            break
        end if
        basisTable[k] ← val
        maxK ← k
        k ← k + 1
    end loop
                                        ▷ Step 3: Greedy Reconstruction of unknown factor
    unknownFactor ← 0
    currentP ← product
    for k ← maxK down to 2 do
        termValue ← basisTable[k]
        if termValue ≤ currentP then
            currentP ← currentP − termValue
            unknownFactor ← unknownFactor + Fibonacci(k)
                                ▷ Optimization: Skip next k (Zeckendorf rule: no consecutive 1s)
            k ← k − 1
        end if
        if currentP == 0 then
            break
        end if
    end for
    return unknownFactor
end procedure
```

9 Conclusion

This report has successfully established a comprehensive framework for arithmetic operations within the Fibonacci number system. While the initial objective was to solve the problem of addition [1], our investigation demonstrated that this operation serves as the foundational logic for a much richer algebraic structure.

We derived a robust methodology for addition consisting of a two-stage process: a naive bitwise sum followed by a critical normalization phase. This normalization is governed by fundamental Fibonacci identities that function as bidirectional "carry" rules [4, 5].

Furthermore, we extended this analysis to Knuth's "circle multiplication," proving its consistency through the "Clean" Lemma and the property of strict monotonicity. We established that the operation produces unique results for all positive integers and satisfies the cancellation law, confirming its algebraic stability. Finally, by leveraging these properties, we developed an algorithmic solution for the inverse operation—"Egyptian Division"—demonstrating that the system is reversible [2, 9].

It is important to acknowledge that the theoretical results and algorithms discussed herein were originally discovered by Donald Knuth and other pioneers in the field. This report does not claim novelty in the mathematical derivation of these theorems. However, the primary contribution of this work lies in the synthesis and crisp explication of these complex topics from disparate academic papers. Navigating and unifying these non-trivial concepts into a coherent, algorithmic framework represents a significant effort in understanding advanced discrete mathematics.

In conclusion, the Fibonacci number system is not merely a method of unique representation, but a fully functional arithmetic domain capable of supporting complex, reversible computations analogous to standard integer arithmetic.

References

[1] Graham, R. L., Knuth, D. E., & Patashnik, O. (1994). *Concrete Mathematics: A Foundation for Computer Science* (2nd ed.). Addison-Wesley. (Exercise 6.82, p. 318).

[2] Knuth, D. E. (1988). Fibonacci Multiplication. *Applied Mathematics Letters, 1*(1), 57-60.

[3] Arnoux, P. (1989). Some Remarks About Fibonacci Multiplication. *Applied Mathematics Letters, 2*(4), 319-320.

[4] Fenwick, P. (2003). Fibonacci numbers and a two-way carry. *The Fibonacci Quarterly*, 41(5), 405–412.

[5] Graham, R. L., Knuth, D. E., & Patashnik, O. (1994). *Concrete Mathematics: A Foundation for Computer Science* (2nd ed.). Addison-Wesley. (Answer to Exercise 6.82, p. 561).

[6] Graham, R. L., Knuth, D. E., & Patashnik, O. (1994). *Concrete Mathematics: A Foundation for Computer Science* (2nd ed.). Addison-Wesley. (p. 561).

[7] Graham, R. L., Knuth, D. E., & Patashnik, O. (1994). *Concrete Mathematics: A Foundation for Computer Science* (2nd ed.). Addison-Wesley. pp. 290-301.

[8] Knuth, D. E. (1988). (Page 2, on Addition). *Applied Mathematics Letters, 1*(1), 57-60.

[9] Henderson, N. (2016). *What is Zeckendorf's Theorem?*.

Chapter 2

Revisiting Graham's Fibonacci-like Composite Sequence

1 Problem Statement

Is it possible that a sequence $\langle A_n \rangle$ satisfying the Fibonacci [1] recurrence $A_n = A_{n-1} + A_{n-2}$ can contain no prime numbers, if A_0 and A_1 are relatively prime?

2 Literature Review

In 1964, R.L. Graham was able to demonstrate that a Fibonacci-like sequence defined by the recursion $A_n = A_{n-1} + A_{n-2}, n \geq 2$ can consist entirely of composite numbers, if the initial terms A_0, A_1 were chosen appropriately [3]. The initial terms $A_0 = a$, and $A_1 = b$ were also relatively prime. Graham's pair (a, b) was as follows: [2]:

$$(331635635998274737472200656430763, 151002891108840197118959030549878 5)$$

Building on Graham's result, subsequently in 1990 Knuth found a smaller pair 17-digit (a, b) : $(62638280004239857, 49463435743205655)$ [4]. In the same year, Herbert Wilf refined this result found an yet even smaller 17-digit pair $(20615674205555510, 3794765361567513)$ [5]. In 1999, John Nicol [6], through extensive computation, discovered a 12-digit pair $(a, b) = (407389224418, 76343678551)$. An even smaller pair in magnitude was found by Vsemirnov in 2004 [7], $(A_0, A_1) = (106276436867, 35256392432)$, which might also be the smallest, as well as the most recent work done on this problem till date.

The gradual reduction in digit length over time reflects continued efforts to identify minimal initial pairs producing composite-only sequences, emphasizing the depth of structure underlying these recursive formulations.

3 Problem Description

The central aim of this construction is to define a Fibonacci-like sequence $\langle A_n \rangle$ with the condition that every term A_n in the sequence is divisible by at-least one fixed prime p_k, and $gcd(A_0, A_1) = 1$ specifically.

We want to ensure that for every integer $n \geq 2$ in the sequence, $A_n \equiv 0 \pmod{p_k}$, for some k, and $A_n \neq p_k$

The condition that A_0 and A_1 is essential because if they had shared a common factor $d > 1$, then $A_0 = d \cdot a_0$, $A_0 = d \cdot a_1$, then every later term is also divisible by d, as they depend on a linear combination of the two previous terms. Hence, $A_n = d \cdot n$, for all n. This would make the result uninteresting as then the sequence would become 'trivially composite', so co-primality of the initial terms is a necessary condition in this problem.

4 Methodology

We try to reconstruct Graham's solution in this section.

4.1 Key Idea

The idea is to ensure that for every $n \geq 2$, there exists a prime p_k satisfying

$$A_n \equiv 0 \pmod{p_k}, \quad \text{but} \quad A_n \neq p_k.$$

Thus, each A_n is divisible by at least one prime p_k, ensuring it is composite.

To achieve this, we design congruence conditions such that each index n is "covered" by some triple (p_k, m_k, r_k), where p_k divides A_n whenever $n \equiv r_k \pmod{m_k}$.

4.2 Key Fibonacci Identity

We begin with a fundamental identity satisfied by Fibonacci numbers:

$$A_{m+n} = A_m F_{n-1} + A_{m+1} F_n \tag{9}$$

which mirrors the standard Fibonacci recurrence,

$$F_{m+n} = F_m F_{n+1} + F_{m-1} F_n \tag{10}$$

which holds true for any sequence $\{A_n\}$ satisfying the Fibonacci recurrence

$$A_{n+2} = A_{n+1} + A_n.$$

The identity is easily established by induction on n.

4.3 Periodicity of Fibonacci Sequences Modulo a Prime

For any modulus p, the Fibonacci sequence $\{F_n\}$ is periodic modulo p. That is, there exists a positive integer m_p, called the *Pisano period*, such that:

$$F_{n+m_p} \equiv F_n \pmod{p}. \tag{11}$$

Let r_p denote the smallest positive integer for which:

$$F_{r_p} \equiv 0 \pmod{p}.$$

Typically, $F_{r_p+1} \not\equiv 0 \pmod{p}$. These indices (m_p, r_p) determine when Fibonacci numbers vanish modulo p.

Existence of Pisano Period

Defining,

$$S_n = (F_n \bmod p,\ F_{n+1} \bmod p).$$

Each S_n takes values in the finite set $\{0, 1, \ldots, p-1\}^2$ containing at most p^2 elements. By the pigeonhole principle, two pairs must coincide, i.e., $S_i = S_j$ for some $i < j$. Since the Fibonacci recurrence is deterministic, the sequence repeats thereafter, implying periodicity.

Formally, the Pisano period m_p is the smallest positive integer such that:

$$(F_{n+m_p}, F_{n+m_p+1}) \equiv (F_n, F_{n+1}) \pmod{p}. \tag{12}$$

4.4 Divisibility Rule from Triples (p, m, r)

Let each triple (p, m, r) define a divisibility pattern such that

$$n \equiv r \pmod{m} \quad \Longrightarrow \quad p \mid A_n.$$

That is, for a given triple (p, m, r), we want $A_r \equiv 0 \pmod{p}$ when $n = r$.

If the set of all such triples covers all integers $n \geq 2$, then each A_n is divisible by some prime p, and hence every A_n in the Fibonacci-like sequence is composite.

We recall the general Fibonacci addition identity valid for any sequence satisfying the Fibonacci recurrence:

$$A_{m+n} = A_m F_{n-1} + A_{m+1} F_n. \tag{13}$$

To ensure that A_r coincides with a Fibonacci term F_m modulo p, we choose the initial conditions:

$$A_0 \equiv F_{m-r} \pmod{p}, \quad A_1 \equiv F_{m-r+1} \pmod{p}.$$

This choice guarantees that the right-hand side of the recurrence expression for A_r mirrors the corresponding Fibonacci identity.

Substituting $n = r$, $m = 0$ into (13), we obtain:

$$\begin{aligned} A_r &= A_0 F_{r-1} + A_1 F_r \\ &\equiv F_{m-r} F_{r-1} + F_{m-r+1} F_r \pmod{p}. \end{aligned} \tag{14}$$

Using the standard Fibonacci addition formula,

$$F_{m-r} F_{r-1} + F_{m-r+1} F_r = F_m,$$

we find that

$$A_r \equiv F_m \pmod{p}.$$

For the chosen prime p, there exists a positive integer $m = m_p$ such that $F_m \equiv 0 \pmod{p}$ by periodicity modulo p. Hence

$$A_r \equiv 0 \pmod{p}, \quad \text{so that } p \mid A_r.$$

Therefore, each triple (p, m, r) defines a residue class of indices for which the sequence term A_n is divisible by p. If these classes cover all integers $n \geq 2$, then no term of the sequence can be prime.

4.5 Combining Congruences via the Chinese Remainder Theorem

For each chosen prime p_k, the corresponding Pisano period m_k, and its zero index r_k, we define congruence conditions on the initial values A_0 and A_1:

$$A_0 \equiv F_{m_k - r_k} \pmod{p_k}, \quad A_1 \equiv F_{m_k - r_k + 1} \pmod{p_k}.$$

These congruences ensure that

$$A_{r_k} \equiv F_{m_k} \equiv 0 \pmod{p_k},$$

so that $p_k \mid A_{r_k}$.

Since each p_k is a distinct prime, the moduli are pairwise co-prime. By the *Chinese Remainder Theorem (CRT)*, there exists a unique pair (A_0, A_1) modulo $M = \prod_{k=1}^{t} p_k$ satisfying all of the above congruences simultaneously.

That is, there exist integers A_0, A_1 such that:

$$\begin{aligned} A_0 &\equiv F_{m_k - r_k} \pmod{p_k}, \\ A_1 &\equiv F_{m_k - r_k + 1} \pmod{p_k}, \end{aligned}$$

for all $k = 1, 2, \ldots, t$, and these are determined uniquely modulo M.

Consequently, the Fibonacci-like sequence defined by

$$A_n = A_{n-1} + A_{n-2}, \text{ and } \quad A_0, A_1 \text{ as above},$$

inherits all of the local divisibility properties simultaneously. For each k,

$$n \equiv r_k \pmod{m_k} \quad \Longrightarrow \quad p_k \mid A_n.$$

If the system of residue classes $\{r_k \pmod{m_k}\}$ *covers all integers* $n \geq 2$, then every term A_n is divisible by at least one of the primes p_k, and thus no term in the sequence can be prime.

4.6 Construction of the Covering System

To ensure that every term A_n in the sequence is divisible by some prime, we require a *covering system* of congruences. A covering system is a finite collection of residue classes

$$n \equiv r_k \pmod{m_k}, \quad (k = 1, 2, \ldots, t),$$

such that every integer $n \geq 2$ satisfies at least one of them.

In the present context, each triplet (p_k, m_k, r_k) satisfies

$$A_{r_k} \equiv 0 \pmod{p_k},$$

so whenever $n \equiv r_k \pmod{m_k}$, we have $p_k \mid A_n$. Therefore, if the residue classes $\{r_k \pmod{m_k}\}$ together cover all integers $n \geq 2$, then every term A_n is composite.

4.6.1 Graham's 18-Triplet Construction

In 1964, R. L. Graham produced the first explicit Fibonacci-like sequence consisting entirely of composite numbers. He used 18 primes p_k with their respective Pisano periods m_k and residues r_k, shown below. Each triple (p_k, m_k, r_k) satisfies

- $F_{m_k} \equiv 0 \pmod{p_k}$, this congruence tells us that the Pisano period m_k is the "cycle length" of the Fibonacci sequence modulo p_k.
- $F_{r_k} \equiv 0 \pmod{p_k}$, this congruence picks a specific offset r_k within that period where p_k divides a Fibonacci number again.

Together, they define a pattern of indices n where p_k divides Fibonacci-like terms, $n \equiv r_k \pmod{m_k}$. This means that whenever n satisfies that congruence, A_n (the Fibonacci-like number at position n) is divisible by p_k.

The corresponding residue class $n \equiv r_k \pmod{m_k}$ marks the indices n for which p_k divides A_n.

$$
\begin{array}{ccc}
(3,4,1) & (2,3,2) & (5,5,1) \\
(7,8,3) & (17,9,4) & (11,10,2) \\
(47,16,7) & (19,18,10) & (61,15,3) \\
(2207,32,15) & (53,27,16) & (31,30,24) \\
(1087,64,31) & (109,27,7) & (41,20,10) \\
(4481,64,63) & (5779,54,52) & (2521,60,60)
\end{array}
$$

It is to be noted that the first column covers all $n \equiv 1 \pmod 2$, the second column covers all $n \equiv 2 \pmod 3$ and $n \equiv 4 \pmod 6$, and the third column covers all $n \equiv 0 \pmod 6$, thus clearly covering all n.

Thus, the residue classes

$$n \equiv r_k \pmod{m_k}, \quad k = 1, 2, \ldots, 18,$$

form a complete covering of all integers $n \geq 2$. For each such n, there exists a prime p_k dividing A_n, ensuring that no term in the sequence is prime.

4.6.2 Later Refinements

Subsequent work by D. E. Knuth (1990) and others showed that smaller covering systems exist, requiring only five primes to achieve the same property. However, Graham's 18-cover remains the classical and conceptually clearest demonstration of the method, illustrating how Pisano periods and residue classes interact to force composite terms in the sequence.

4.6.3 Connection to Initial Conditions

Once the covering system is chosen, the next step is to determine suitable initial values (A_0, A_1) so that the required congruences

$$A_0 \equiv F_{m_k - r_k} \pmod{p_k}, \quad A_1 \equiv F_{m_k - r_k + 1} \pmod{p_k}$$

hold simultaneously for all k. This is achieved using the *Chinese Remainder Theorem*, as discussed in the following subsection.

4.7 Verification of the Covering Property

Having specified the set of triplets (p_k, m_k, r_k), it remains to verify that these congruences indeed form a covering system; that is, for every integer $n \geq 2$, there exists some k such that

$$n \equiv r_k \pmod{m_k}.$$

4.7.1 Method of Verification

The verification can be done algorithmically (6). Conceptually, one may think of listing all the residue classes

$$r_k + m_k\mathbb{Z}, \quad k = 1, 2, \ldots, 18,$$

and checking that every integer $n \geq 2$ belongs to at least one of these arithmetic progressions. In practice, a computer program can confirm that the union of these classes covers all integers up to the least common multiple (LCM) of the moduli:

$$L = \operatorname{lcm}(m_1, m_2, \ldots, m_{18}).$$

Because each residue class repeats every m_k terms, verifying coverage from 2 up to L is sufficient to guarantee coverage for all integers.

4.8 Illustration

For instance, the first few congruences cover the following sets:

$$\begin{aligned}
&n \equiv 1 \pmod 4 \quad (\text{covered by } p = 3),\\
&n \equiv 2 \pmod 3 \quad (\text{covered by } p = 2),\\
&n \equiv 1 \pmod 5 \quad (\text{covered by } p = 5),\\
&n \equiv 3 \pmod 8 \quad (\text{covered by } p = 7), \quad \text{and so on.}
\end{aligned}$$

When all 18 such classes are taken together, they collectively span every integer $n \geq 2$.

Hence, the covering property ensures that for each $n \geq 2$, there is at least one p_k such that $p_k \mid A_n$. Consequently, every term A_n is guaranteed to be composite, completing the construction.

5 Further Observations

We observe that if we backtrack on Graham's initial pair, we get

$$151002891108840197118959030549878 5(A_1) - 331635635998274737472200656430763(A_0)$$

$$= 1178393275090127233717389649068022(A_{-1})$$

which is incidentally a composite term and also co-prime to it's next adjacent value, i.e, A_0. Thus this leads to the question on why Graham hadn't used A_{-1} as A_0, and A_0 as A_1 in his construction.

It is important to note that the construction is not invariant under shifts of the index. Indeed, for the explicit values given by Graham, repeated backtracking can produce earlier positive terms. However, relabeling such an earlier pair as the new initial values would fundamentally alter the arithmetic structure of the sequence. The reason is that Graham's construction relies essentially on a system of congruences of the form

$$n \equiv r_k \pmod{m_k} \implies p_k \mid A_n$$

where each residue class, $r_k \pmod{m_k}$ is carefully chosen so that the union of these classes covers all integers $n \geq 2$. These divisibility conditions are index-sensitive: they depend explicitly on the position n of each term in the sequence.

As a result, shifting the indexing of the sequence by even one position modifies every congruence class r_k, destroying the covering property and removing the guarantee that every term is divisible by at least one specified prime. Thus, although many Fibonacci-like sequences may be generated from the same recurrence relation, only the uniquely indexed sequence defined by Graham's chosen initial values satisfies the required simultaneous congruences.

This highlights a crucial distinction between the classical Fibonacci sequence, whose properties are invariant under index shifts, and Graham's construction, where the arithmetic behavior of the sequence is intrinsically tied to its indexing.

5.1 Another Open Direction

In this section, we consider a natural extension of the Fibonacci–like sequences studied previously, namely sequences whose initial values are not relatively prime. Let $\{A_n\}_{n\geq 0}$ be defined by the recurrence

$$A_{n+2} = A_{n+1} + A_n, \qquad n \geq 0,$$

with initial values A_0 and A_1 satisfying

$$\gcd(A_0, A_1) = D > 1.$$

Since both initial terms share the common divisor D, they may be written as $A_0 = Da$ and $A_1 = Db$, where $\gcd(a, b) = 1$. Due to the linear nature of the recurrence relation, every subsequent term inherits this factor, and the sequence can be expressed as

$$A_n = DB_n \quad \text{for all } n \geq 0,$$

where $\{B_n\}$ is itself a Fibonacci–like sequence defined by

$$B_0 = a, \quad B_1 = b, \quad B_{n+2} = B_{n+1} + B_n.$$

A notable invariant of Fibonacci–like recurrences is that the greatest common divisor of consecutive terms remains unchanged throughout the sequence. Indeed, using the recurrence relation and the

Euclidean algorithm, one obtains

$$\gcd(A_{n+1}, A_n) = \gcd(A_1, A_0) = D$$

for all $n \geq 0$. As a result, every term of the sequence is a multiple of D, while no additional global divisor is introduced by the recurrence beyond this initial constraint.

This observation naturally leads to the following question: **'Is it possible to choose initial values A_0 and A_1 with $\gcd(A_0, A_1) = D > 1$ such that no term of the resulting sequence is an exact power of the common divisor? More precisely, can we construct a sequence for which'**

$$A_n \neq D^m \quad \textbf{for all integers } n \geq 0 \textbf{ and } m \geq 1?$$

In terms of the reduced sequence $\{B_n\}$, this is equivalent to requiring that no term B_n equals a power of D.

This is a Diophantine / exponential-equation kind of question:

$$B_n = D^k \quad (\text{ with } B_{n+2} = B_{n+1} + B_n)\,.$$

References

[1] Donald Knuth, Oren Patashnik, and Ronald Graham. *Concrete Mathematics*, 290–301, 1988.

[2] Donald Knuth, Oren Patashnik, and Ronald Graham. *Concrete Mathematics*, 6(83): 561–562, 1988.

[3] Ronald L. Graham. *A Fibonacci-like sequence of composite numbers*, Mathematics Magazine, 37, 322–324, 1964.

[4] Donald E. Knuth. *A Fibonacci-like sequence of composite numbers*, Mathematics Magazine, 63, 21–25, 1990.

[5] Herbert S. Wilf, *Letters to the Editor*, Mathematics Magazine, 63, 284, 1990.

[6] John W. Nicol. *A Fibonacci-like sequence of composite numbers*, The Electronic Journal of Combinatorics, (R44)6, 1999.

[7] M. Vsemirnov, *A new Fibonacci-like sequence of composite numbers*, Journal of Integer Sequences, Article 04.3.7, Vol. 7(2004).

6 Appendix

In this section we try to validate the Graham's construction through computation.

Validating that Graham's 18 triplet covers all positive integers n

Code:

```python
# Graham's 18 triples (p_k, m_k, r_k)
triples = [
    (3, 4, 1),
    (2, 3, 2),
    (5, 5, 1),
    (7, 8, 3),
    (17, 9, 4),
    (11, 10, 2),
    (47, 16, 7),
    (19, 18, 10),
    (61, 15, 3),
    (2207, 32, 15),
    (53, 27, 16),
    (31, 30, 24),
    (1087, 64, 31),
    (109, 27, 7),
    (41, 20, 10),
    (4481, 64, 63),
    (5779, 54, 52),
    (2521, 60, 60),
]

# Step 1: Collecting all moduli m_k
moduli = [m for (_, m, _) in triples]

# Step 2: Computing LCM of all moduli (period of the covering)
from math import gcd
def lcm(a, b): return a * b // gcd(a, b)

L = 1
for m in moduli:
    L = lcm(L, m)

print("LCM of moduli =", L)

# Step 3: Checking all n in one full cycle [0, L)
covered = [False] * L

for (_, m, r) in triples:
    for n in range(r, L, m):
        covered[n % L] = True

# Step 4: Finding uncovered numbers (if any)
uncovered = [n for n in range(2, L) if not covered[n % L]]

if uncovered:
    print("Uncovered residues:", uncovered)
    print("Not a complete covering system!")
else:
```

```
    print("Every integer n >= 2 is covered by at least one congruence.")
```

Listing 1: Graham's Triplet Verification

Output:

```
LCM of moduli = 8640
Every integer n >= 2 is covered by at least one congruence.
```

Explanation of the above program

We have 18 congruences:

$$n \equiv r_k \pmod{m_k}, \quad k = 1, 2, \ldots, 18,$$

and we want to make sure that for every integer $n \geq 2$, there exists at least one k such that $(n - r_k)$ is divisible by m_k. If this holds true for all n, then the system 'covers' every integer. The set of congruences being periodic, i.e., each one repeats every m_k steps makes the entire system repeat every $L = lcm(m_1, m_2, \ldots, m_{18})$. Thus, we can only check whether the system covers all integers from 2 up-to L, as it will then automatically cover integers $> L$ as well. The python code above explicitly checks the same.

Finding the initial values given by Graham

Code:

```
from math import prod

# Fibonacci mod calculation
def fib_mod(n, mod):
    a, b = 0, 1
    for _ in range(n):
        a, b = b % mod, (a + b) % mod
    return a

# Graham's 18 triples (p, m, r)
triples = [
    (3, 4, 1), (2, 3, 2), (5, 5, 1), (7, 8, 3), (17, 9, 4), (11, 10, 2),
    (47, 16, 7), (19, 18, 10), (61, 15, 3), (2207, 32, 15), (53, 27, 16),
    (31, 30, 24), (1087, 64, 31), (109, 27, 7), (41, 20, 10), (4481, 64, 63),
    (5779, 54, 52), (2521, 60, 60)
]

# For each triple, computing the congruences for A_0 and A_1
a0_congruences = []
a1_congruences = []

for (p, m, r) in triples:
    a0 = fib_mod(m - r, p)
    a1 = fib_mod(m - r + 1, p)
    a0_congruences.append((a0, p))
```

```
    a1_congruences.append((a1, p))

# A solver for Chinese Remainder Theorem
def crt(congruences):
    x, M = 0, 1
    for (_, mod) in congruences:
        M *= mod
    for (a, mod) in congruences:
        Mi = M // mod
        inv = pow(Mi, -1, mod)
        x = (x + a * Mi * inv) % M
    return x, M

A0, mod_A0 = crt(a0_congruences)
A1, mod_A1 = crt(a1_congruences)

print("A0 =", A0)
print("A1 =", A1)
```

Listing 2: Finding A_0 *and* A_1

Output:

```
A0 = 331635635998274737472200656430763
A1 = 151002891108840197118959030549878 5
```

We observe that through our computation, we are able to find the exact same values for A_0, A_1 as was proposed by Graham in 1964.