Algorithmic Problem-Solving: A Comprehensive Approach to Project Euler Problems 1 and 2

Luis Perez

Problem 1. Summation of multiples of 3 or 5 less than a positive integer n

Description:
Suppose we list all the positive integers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9, the sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 1000, 8456, 19564 and in general find the sum of multiples of 3 or 5 less than any arbitrarily given n.

Definition 1 (Summation). Suppose $m, n \in \mathbb{Z}$, the symbol $\sum_{i=m}^{n} a_i$, describes the sum of all the terms $a_m, a_{m+1}, a_{m+2}, ..., a_n$. Thus

$$\sum_{i=m}^{n} a_i = a_m +, a_{m+1} +, a_{m+2} +, ..., +a_n.$$

We use i as an arbitrary index of the summation, m is the lower limit of the summation or the starting index, and n is the upper limit of the summation or the ending index of the summation.

Theorem 1 (Generalized distributive law of summation). If $a_n, a_{n+1}, a_{n+2}, ...$ and b_n, b_{n+1}, b_{n+2} where $a_k, b_k, c \in \mathbb{R}$. Then the following statement holds for any integer $m \geq n$

$$c \cdot \sum_{i=n}^{m} a_i = \sum_{i=n}^{m} c \cdot a_i$$

Definition 2 (Floor function). Let $\lfloor . \rfloor : \mathbb{R} \to \mathbb{Z}$, then $\lfloor n \rfloor$ is defined as greatest integer less than or equal to a real number n, thus $\lfloor n \rfloor$ is a unique integer such that $\lfloor n \rfloor \leq n \leq \lfloor n \rfloor + 1$.

Theorem 2. For all $n \geq 1 \in \mathbb{Z}^+$,

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + ... + n = \frac{n(n+1)}{2}$$

Proof. We employ induction, let $n = 1$, since $\frac{1(1+1)}{2} = \frac{2}{2} = 1$ and $\sum_{i=1}^{1} i = 1$, the statement is true for $n = 1$. Suppose

$$1 + 2 + 3 + ... + k = \frac{k(k+1)}{2}$$

for some arbitrary positive integer k. We show that

$$1 + 2 + 3 + ... + (k+1) = \frac{k+1((k+1)+1)}{2} = \frac{(k+1)(k+2)}{2}.$$

Thus by the inductive hypothesis

$$\begin{aligned} 1 + 2 + 3 + ... + (k+1) &= 1 + 2 + 3 + ... + k + (k+1) \\ &= \frac{k(k+1)}{2} + (k+1) \\ &= \frac{k(k+1)}{2} + \frac{2(k+1)}{2} \\ &= \frac{k(k+1) + 2(k+1)}{2} \\ &= \frac{(k+1)(k+2)}{2} \end{aligned}$$

Therefore for all $n \geq 1 \in \mathbb{Z}^+$, $\sum_{i=1}^{n} i = 1 + 2 + 3 + ... + n = \frac{n(n+1)}{2}$ ■

Result 1. The quantity of multiples of k less than or equal to n is denoted as

$$\left\lfloor \frac{n}{k} \right\rfloor$$

Proof. Suppose $n, k, m_i \in \mathbb{Z}^+$, the positive multiples of k are as follows $k, 2k, 3k, ..., m_1k, m_2k$. Let $m_1k \leq n \leq m_2k$, where $m_2k = m_1k+1$. Or equivalently when we divide by k

$$m_1 \leq \frac{n}{k} \leq m_2$$

Then m_1 is the greatest integer less than or equal to $\frac{n}{k}$ thus by **Definition 2** (Floor Function) the quantity of multiples of k less than or equal to n is denoted as $\lfloor \frac{n}{k} \rfloor$ ■

Solution: By **Result 1** the quantity of multiples of 3 and 5 less than or equal to n are $\lfloor \frac{n}{3} \rfloor$ and $\lfloor \frac{n}{5} \rfloor$, however the challange states less than n thus n is excluded so this means the quantity of multiples of 3 and 5 less than n are $\lfloor \frac{n-1}{3} \rfloor$ and $\lfloor \frac{n-1}{5} \rfloor$. The summation of multiples of 3 is $3+6+9+12+...+3m$ by **Theorem 1** this is equivalent to $3(1+2+3+4+...+m)$. Suppose m is the quanitity of multiples of 3 less than n, and this also the case for the summation of multiples of 5, a case we must consider is the multiples of both 3 and 5. Therefore the summation of multiples of 3 or 5 less than a given n denoted as a function is

$$f(n) = 3\left(\frac{\lfloor \frac{n-1}{3} \rfloor \left(\lfloor \frac{n-1}{3} \rfloor + 1\right)}{2}\right)+5\left(\frac{\lfloor \frac{n-1}{5} \rfloor \left(\lfloor \frac{n-1}{5} \rfloor + 1\right)}{2}\right)-15\left(\frac{\lfloor \frac{n-1}{15} \rfloor \left(\lfloor \frac{n-1}{15} \rfloor + 1\right)}{2}\right).$$

Algorithm in Python

```
import math
def summation(n):
        return int((n*(n+1))/2)
def mult_k(n,k):
        return math.floor((n)/k)
def f(n):
        return  3*(summation(mult_k(n-1,3)))+
        5*(summation(mult_k(n-1,5)))
        -15*(summation(mult_k(n-1,15)))

Function input                Function output

print(f(10))                  23
print(f(49))                  543
print(f(1000))                233168
print(f(8456))                16687353
print(f(19564))               89301183
```

Problem 2. Even Fibonacci Numbers

Description:
Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...
By considering the terms in the Fibonacci sequence whose values is less than or equal to n, find the sum of the even-valued terms.
The Fibonacci sequence is defined by the recurrence relation

$$\begin{aligned} f_0 &= 0 \\ f_1 &= 1 \\ f_2 &= 1 \\ f_{n+1} &= f_n + f_{n-1} \end{aligned}$$

Computing the nth Fibonacci number can be done using matrix multiplication. Firstly let us define what a matrix is and how matrix multiplication is computed. A matrix is represented as a rectangular array comprised of values. Each matrix has a quantity of rows and columns an $m \times m$ matrix contains m rows and columns. Let A be a $m \times n$ matrix denoted as

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

thus A has m rows and n columns. Typically the values of a matrix are comprised of real numbers ($a_{ij} \in \mathbb{R}$), but complex numbers can also be used, and in our case we will be more specific by using positive integers for computing the nth Fibonacci number. Suppose A is an $m \times n$ matrix and B is an $p \times q$ matrix the resulting matrix from $A \times B$ will be a $m \times q$ matrix C, in order to multiply two matrices the columns of the first matrix n has to equal the rows of the second matrix p. Let

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1q} \\ b_{21} & b_{22} & \cdots & b_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ b_{p1} & b_{p2} & \cdots & b_{pq} \end{bmatrix}.$$

Then

$$A \times B = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1q} \\ b_{21} & b_{22} & \cdots & b_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ b_{p1} & b_{p2} & \cdots & b_{pq} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1q} \\ c_{21} & c_{22} & \cdots & c_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mq} \end{bmatrix}$$

Where

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in} + b_{pj} = \sum_{k=1}^{n} a_{ik}b_{kj}$$

We introduce a theorem which shows that matrix exponentiation can be utilized for computing an nth term of the Fibonacci sequence. Recall the recurrence relation of the Fibonacci sequence.

Theorem 3. Let $A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$

$$A^n = \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix}$$

$\forall n \in \mathbb{Z}^+ \geq 1$

Proof. We prove this proposition by method of induction, let $n = 1$, since $A^1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ and $\begin{bmatrix} f_{1+1} & f_1 \\ f_1 & f_{1-1} \end{bmatrix} = \begin{bmatrix} f_2 & f_1 \\ f_1 & f_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ the statement is true for $n = 1$. Suppose that

$$A^k = \begin{bmatrix} f_{k+1} & f_k \\ f_k & f_{k-1} \end{bmatrix}$$

for some positive integer $k \geq 1$. We show that

$$A^{k+1} = \begin{bmatrix} f_{k+2} & f_{k+1} \\ f_{k+1} & f_k \end{bmatrix}$$

By the inductive hypothesis

$$\begin{aligned} A^{k+1} &= A^k A \\ &= \begin{bmatrix} f_{k+1} & f_k \\ f_k & f_{k-1} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} f_{k+1} + f_k & f_{k+1} \\ f_k + f_{k-1} & f_k \end{bmatrix} \\ &= \begin{bmatrix} f_{k+2} & f_{k+1} \\ f_{k+1} & f_k \end{bmatrix} \end{aligned}$$

Therefore

$$A^n = \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix}$$

$\forall n \in \mathbb{Z}^+ \geq 1$ ■

We introduce an important algorithm called binary exponentiation for efficient computation. Suppose $n, m \in \mathbb{Z}^+$, using the binary expansion of m, $m = (k_{i-1}, ..., k_1, k_0)_2$, n^m can be expressed in the following terms, $n^m = n^{k_{i-1} \cdot 2^{k-1} + ... + k_1 \cdot 2 + k_0}$. Which is equivalent to $n^m = n^{k_{i-1}} \cdot ... \cdot n^{k_1 \cdot 2} \cdot n^{k_0}$. Suppose $n = 5$ and $m = 7$, then the binary expansion of 7 is $(0111)_2$ or $(0 \cdot 2^3) + (1 \cdot 2^2) + (1 \cdot 2^1) + (1 \cdot 2^0)$. Thus instead of computing n^m using 7 or m multiplications that is $5 \cdot 5 \cdot 5 \cdot 5 \cdot 5 \cdot 5 \cdot 5$, to compute n^m we can compute $5^4 \cdot 5^2 \cdot 5^1 = (5^2)^2 \cdot 5^2 \cdot 5 = 25^2 \cdot 25 \cdot 5 = 625 \cdot 25 \cdot 5 = 78125$, which is most efficient as it requires less multiplications. We call this binary exponentiation, this method can also be applied for computing matrices with powers. Thus we will use this method for computing the nth Fibonacci number.

Algorithm 1. Binary exponentiation.

```
Require: $n, m > 1 \in \mathbb{Z}^+$
Ensure: $y = n^m$
  $y \leftarrow 1$
  $N \leftarrow n$
  $M \leftarrow m$
  while $M \neq 0$ do
    $x \leftarrow M \mod 2$
    $M \leftarrow \lfloor \frac{M}{2} \rfloor$
    if $x = 1$ then
      $y \leftarrow y \cdot N$
    end if
    $N \leftarrow N \cdot N$
  end while
  return $y$
```

We implement this algorithm using Python for computing matrix exponentiation.

Program in Python

```
import numpy as np

def binary_matrix_exponentiation(N, m):
        y = np.array([[1, 0], [0, 1]])
        M = m
        while M != 0:
                x = M % 2
                M = M // 2
                if x == 1:
                        y = np.dot(y, N)
                N = np.dot(N, N)
        return y

def nth_Fibonacci(n):
        if(n>92):
                raise ValueError("Cannot compute Fibonacci numbers
                larger than 92 due to integer overflow.")
        A = np.array([[1, 1], [1, 0]])
        return binary_matrix_exponentiation(A, n)[0][0]

for i in range(100):
print(nth_Fibonacci(i))
```

In the Fibonacci sequence, every third number is even and the two numbers between are odd, this pattern repeats indefinitely. Thus we consider the three forms of the a Fibonacci number, due to the periodic nature of the parity of Fibonacci numbers. Recall the recurrence relation defined earlier, given this the forms we examine are f_{3n}, f_{3n+1}, and f_{3n+2}

Theorem 4. For all $n \geq 1 \in \mathbb{Z}$, f_{3n} is even.

Proof. We prove this theorem by induction, let $n = 1$, since $f_{3n} = f_3 = 2$ and 2 is even then the statement is true for $n = 1$. Suppose that f_{3k} is even for some arbitrary positive integer k. Then

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

By the inductive hypothesis f_{3k} is even and $2(f_{3k+1})$ is even by definition. Therefore $\forall n \geq 1 \in \mathbb{Z}$, f_{3n} is even. ■

Theorem 5. For all $n \geq 0 \in \mathbb{Z}$, f_{3n+1}, f_{3n+2} is odd.

Proof. Proof by induction resembles Theorem 4 and is omitted. ■

Corollary 1. All even Fibonacci numbers are only of the form f_{3n} for all $n \geq 1 \in \mathbb{Z}^+$.

Proof. By Theorem 4 f_{3n} is even, and from Theorem 5 f_{3n+1} and f_{3n+2} are always odd.
We show that no Fibonacci numbers of the form f_{3n+1} or f_{3n+2} can be even. This implies that all even Fibonacci numbers must be of the form f_{3n}.
Assume to the contrary that there exists an n such that f_{3n+1} or f_{3n+2} is even. This would contradict Theorem 5, which states that f_{3n+1} and f_{3n+2} are always odd. Therefore, no such n can exist.
Hence, all even Fibonacci numbers must be only of the form f_{3n}, $n \geq 1 \in \mathbb{Z}^+$. ■

We present crucial proposition regarding Fibonacci numbers. While its not entirely certain how this formula was derived from our source it is verifiably true using induction for which we leave as an exercise to the reader (non-trivial proof), for more regarding this result visit the following sequence A130233 for which the following formula is referenced.

Theorem 6. For all $n \geq 1 \in \mathbb{Z}^+$, the quantity of Fibonacci numbers less than or equal to n is denoted as the counting function

$$\pi_\phi(n) = \left\lfloor \log \phi \left(\left(n\sqrt{5} + \sqrt{5n^2+4} \right) \frac{1}{2} \right) \right\rfloor$$

$$\phi = \frac{1+\sqrt{5}}{2}$$

Since the proof is left to the reader we provide a python program for verifying that this formula discussed holds for all numbers less than or equal to 4 million.

Program in Python

```
import math
def pi_phi(n):
        phi = (1 + math.sqrt(5)) / 2
        epsilon = 1e-10  # counteract rounding errors
        return math.floor(math.log((math.sqrt(5)*n +
        math.sqrt(5*n**2 + 4))/2, phi) + epsilon)

def fib_count(n):
        fib_seq = [1, 1]
        while fib_seq[-1] <= n:
                fib_seq.append(fib_seq[-1] + fib_seq[-2])
        return len(fib_seq) - 1

# Verification that the formula holds for numbers up to 4 million
for n in range(1, 4000001):
        formula_result = formula(n)
        if formula_result != pi_phi(n):
        print(f"The formula does not hold for n = {n}")
        break
else:
print("The formula holds for all n up to 4 million")
```

Theorem 7. The summation of n even consecutive Fibonacci numbers is

$$\begin{aligned}\sum_{i=1}^{n} f_{3i} &= \\ f_3 + f_6 + ... + f_{3n} &= (f_4 - f_2) + (f_7 - f_5) + ... + (f_{3n+1} - f_{3n-1}) \\ &= f_{3n+1} - \frac{f_{3n-1} + 1}{2}\end{aligned}$$

$\forall n \geq 1 \in \mathbb{Z}^+$.

Proof. We prove this proposition by method of induction, let $n = 1$, since $f_4 - \frac{f_2+1}{2} = 3 - 1 = 2$, and $\sum_{i=1}^{1} f_{3i} = 2$, the statement is true for $n = 1$. Suppose

$$\sum_{i=1}^{k} f_{3i} = f_{3k+1} - \frac{f_{3k-1} + 1}{2}$$

for some arbitrary positive integer k. We show that

$$\sum_{i=1}^{k+1} f_{3i} = f_{3k+4} - \frac{f_{3k+2} + 1}{2}.$$

Thus by the inductive hypothesis

$$\begin{aligned}
\sum_{i=1}^{k+1} f_{3i} &= \sum_{i=1}^{k} f_{3i} + f_{3(k+1)} \\
&= f_{3k+1} - \frac{f_{3k-1}+1}{2} + f_{3k+3} \\
&= \frac{2(f_{3k+1})}{2} - \frac{f_{3k-1}+1}{2} + \frac{2(f_{3k+3})}{2} \\
&= \frac{2(f_{3k+1}) - f_{3k-1} - 1 + 2(f_{3k+3})}{2} \\
&= \frac{2(f_{3k+1}) - f_{3k+2} - 1 + f_{3k+3}}{2} \\
&= \frac{2(f_{3k+4}) - f_{3k+2} - 1}{2} \\
&= \frac{2(f_{3k+4})}{2} + \frac{-f_{3k+2}-1}{2} \\
&= f_{3k+4} - \left(\frac{-f_{3k+2}-1}{2}\right) \\
&= f_{3k+4} - \frac{f_{3k+2}+1}{2}
\end{aligned}$$

Therefore for all $n \geq 1 \in \mathbb{Z}$, $\sum_{i=1}^{n} f_{3i} = f_{3n+1} - \frac{f_{3n-1}+1}{2}$ ■

Theorem 8. Given $\pi_\phi(n)$, the computing function for the quanitity of Fibonacci numbers less than or equal to n, the amount of even Fibonacci numbers less than or equal to n is $\left\lfloor \frac{\pi_\phi(n)}{3} \right\rfloor$.

Proof. As shown by the recurrence relation

$$\begin{aligned}
f_0 &= 0 \\
f_1 &= 1 \\
f_2 &= 1 \\
f_{n+1} &= f_n + f_{n-1}
\end{aligned}$$

every third Fibonacci number in the sequence is even of the form f_{3n} with previous and subsequent numbers in the sequence being odd of the form f_{3n+1}, and f_{3n+2}. By Theorem 4 and Corollary 1 it follows that a Fibonacci number f_n is even if and only if n is a multiple of 3. Hence, given this periodic parity, one third of Fibonacci numbers less than or equal to n are even.

Let E_n denote the rational approximation of even Fibonacci numbers less than or equal to n. We can express E_n as a function of $\pi_\phi(n)$, the total number of Fibonacci numbers less than or equal to n. That is, $E_n = \frac{\pi_\phi(n)}{3}$.
However, computing E_n produces a rational number, and we require strictly an integer, we employ the floor function. By Definition 2 (Floor function), $\lfloor x \rfloor$ is the greatest integer less than or equal to x. Thus, $\lfloor E_n \rfloor$ is the greatest integer less than or equal to E_n, which computes the exact quantity of even Fibonacci numbers less than or equal to n. Therefore the quantity of even Fibonacci numbers less than or equal to n is $\left\lfloor \frac{\pi_\phi(n)}{3} \right\rfloor$. ■

Solution: By Theorem 8 computing the quantity of even Fibonacci numbers less than or equal to n is $\left\lfloor \frac{\pi_\phi(n)}{3} \right\rfloor$. By Theorem 7 the summation of n even consecutive Fibonacci numbers is $f_{3n+1} - \frac{f_{3n-1}+1}{2}$, given this, we implement this solution in Python.
Algorithmic implementation in Python

```
import math
import numpy as np

def binary_matrix_exponentiation(N, m):
        y = np.array([[1, 0], [0, 1]])
        M = m
        while M != 0:
        x = M % 2
        M = M // 2
        if x == 1:
        y = np.dot(y, N)
        N = np.dot(N, N)
        return y

def pi_phi(n):
        phi = (1 + math.sqrt(5)) / 2
        epsilon = 1e-10  # counteract rounding errors
        return math.floor(math.log((math.sqrt(5)*n +
        math.sqrt(5*n**2 + 4))/2, phi) + epsilon)

def nth_Fibonacci(n):
        if(n>92):
        raise ValueError("Cannot compute an nth Fibonacci number
        larger than 92 due to integer overflow.")
        A = np.array([[1, 1], [1, 0]])
        return binary_matrix_exponentiation(A, n)[0][0]
```