Project Euler, Problem 2:
Even Fibonacci Numbers

John Butler

1 Problem Description

Find the sum of all $F_n < 4,000,000$ such that F_n is even, where

$$F_n = \begin{cases} 0 & \text{for } n = 0 \\ 1 & \text{for } n = 1 \\ F_{n-1} + F_{n-2} & \text{for } n \geq 2 \end{cases}$$

2 Theorems whose results aid in my solution

2.1 Lemma: $\forall n \in \mathbb{N}$, $\exists k \in \mathbb{N} : n = 2k$ or $n = 2k + 1$

case $n = 0$	case $n = 1$
$0 = 2 \cdot 0$	$1 = 0 + 1$
	$1 = 2 \cdot 0 + 1$
Let $k = 0$	Let $k = 0$
$0 = 2k$	$1 = 2k + 1$

Assume $n - 1 = 2k$, or $n - 1 = 2k + 1$

case $n - 1 = 2k$	case $n - 1 = 2k + 1$
$n = 2k + 1$	$n = 2k + 2$
	$n = 2(k + 1)$
	Let $k' = k + 1$
	$n = 2k'$

2.2 Lemma: o is odd $\iff \exists n \in \mathbb{N} : o = 2n + 1$, and e is even $\iff \exists n \in \mathbb{N} : e = 2n$

Suppose e is even.

$\iff e$ is divisible by 2

$\iff \frac{e}{2} = n$ for some $n \in \mathbb{N}$

$\iff e = 2n$

Suppose o is odd.

$\iff o$ is not divisible by 2

$\iff o$ is not even.

$\iff o \neq 2n$

$\iff o = 2n + 1$ by lemma 1

2.3 Lemma: An odd number plus an even number equals an odd number, and an odd number plus an odd number equals an even number

Consider an odd number plus an even number.

$$\begin{aligned} &\iff (2n_1+1)+(2n_2) \text{ for some } n_1, n_2 \in \mathbb{N} \\ &=2n_1+2n_2+1 \\ &=2(n_1+n_2)+1 \\ \text{Let } n_3 &=n_1+n_2 \\ &=2n_3+1 \end{aligned}$$

which is odd by lemma 2.

$\therefore$ an odd plus an even equals an odd $\square$

Now consider an odd number plus an odd number.

$$\begin{aligned} &\iff (2n_1+1)+(2n_2+1) \text{ for some } n_1, n_2 \in \mathbb{N} \\ &=2n_1+2n_2+1+1 \\ &=2n_1+2n_2+2 \\ &=2(n_1+n_2+1) \\ \text{Let } n_3 &=n_1+n_2+1 \\ &=2n_3 \end{aligned}$$

which is even by lemma 2.

$\therefore$ an odd plus an odd equals an even $\square$

2.4 Theorem: only every third Fibonacci number is even

Proof by induction:
case $n \in \{0, 1, 2\}$

$F_0 = 0$	$F_1 = 1$	$F_2 = F_1 + F_0$
	$F_1 = 0 + 1$	
$F_0 = 2 \cdot 0$	$F_1 = 2 \cdot 0 + 1$	$F_2 = F_1 + 0$
Let $k_1 = 0$	Let $k_2 = 0$	
$\therefore F_0 = 2k_1$	$\therefore F_1 = 2k_2 + 1$	$F_2 = F_1$
$\therefore F_0$ is even	$\therefore F_1$ is odd	$\therefore F_2$ is odd

Assume F_{n-3} is even, F_{n-2} is odd, and F_{n-1} is odd

$F_n = F_{n-1} + F_{n-2}$	$F_{n+1} = F_n + F_{n-1}$	$F_{n+2} = F_{n+1} + F_n$
$F_n =$ odd + odd	$F_{n+1} =$ even + odd	$F_{n+2} =$ odd + even
$F_n =$ even	$F_{n+1} =$ odd	$F_{n+2} =$ odd $\square$

3 Application to Code

Now that we know that every third Fibonacci number is even, we don't have to worry about evenness. Instead, since we know that $F_0 = 0$ is the first Fibonacci number which is even, we can add F_0 to our sum, then compute F_1, F_2, and then add $F_3 = 2$ to our sum, and so on. Granted, checking if a number is even is computationally light.

Additionally, F_0 doesn't affect our sum, so we can start with $F_3 = 2$; Project Euler doesn't even mention F_0 or F_1.

4 Furthur Optimizations

For this section, assume that n is the index of the maximum Fibonacci number we must execute, and L is the limit $(4,000,000)$ Considering the definition, it calculating the Fibonacci numbers would seem to be most natural to use a top-down recurrsive algorithm, but this would be incredibly slow, $O(2^n)$ for *each* number, meaning we end up with a time complexity of $O(n2^n)$. It doesn't help that this is the example used by many introductory programming courses for recursion, so this is what many people write first. There are of course optimizations you can apply, such as providing a memo so we don't have to recompute F_{n-1} and F_{n-2}, or any previous numbers, but that has linear space complexity. Not only that, but once don't even need to store F_{n-3}, since we're always only computing the next number.
The algorithm I'm using uses a bottom up approach, which has an overall space complexity of $O(1)$ and a time complexity of $O(n)$. The idea is to store only F_{n-1} and F_{n-2}, so that we can compute F_n. After that we shift over what we're storing, so that we forget F_{n-2} and keep F_n and F_{n-1} in order to calculate F_{n+1}.