Partitions of Fibonacci numbers into distinct Fibonacci parts

Jason d'Eon

December 6, 2022

Below is described a problem revolving around some results I came across on Fibonacci numbers. Included are relevant theorems and their proofs, as well as an implemented solution.

1 Reference

Robbins, Neville. "Fibonacci Partitions." *The Fibonacci Quarterly,* Vol. *34.4* (1996): pp. 306–313.

2 Problem Statement

We let F_n be the nth Fibonacci number, where $F_1 = 1$ and $F_2 = 1$. Define $p(F_n)$ to be the number of ways to partition F_n into distinct Fibonacci parts, that is, where the summands are Fibonacci numbers. For example, $p(F_2) = 1$ and $p(F_{10}) = 5$.

Further we define

$$S(n) = \sum_{k=2}^{n} p(F_k)$$

You are given $S(123) = 3782$ and $S(12\,345) = 38\,099\,756$.

Find $S(123\,456\,789)$.

3 Solution

First, we obtain a useful lemma.

Lemma 1. $\sum_{k=2}^{n} F_k = F_{n+2} - 2$.

Proof. We proceed by induction on n. Note that the statement holds for $n = 2$, since $F_2 = 1 = 3-2 = F_4-2$. Now suppose the statement holds for some m, that is $\sum_{k=2}^{m} F_k = F_{m+2} - 2$. Then by the definition of Fibonacci numbers, $\sum_{k=2}^{m+1} F_k = F_{m+1} + F_{m+2} - 2 = F_{m+3} - 2$. Therefore the statement holds for $m+1$. □

The problem is made trivial once the following (slightly surprising) theorem is proved.

Theorem 2. $p(F_n) = \lfloor n/2 \rfloor$ *when* $n \geq 2$.

Proof. We apply induction once more. Trivially, $p(F_2) = 1 = \lfloor 2/2 \rfloor$ and $p(F_3) = 1 = \lfloor 3/2 \rfloor$. Now suppose $p(F_m) = \lfloor m/2 \rfloor$ for some $m \geq 4$. By Lemma 1, we know that the sum of the distinct F_k up to $k = m-2$ would only be $F_m - 2$, so a nontrivial partition of F_m must contain F_{m-1}. Since $F_m = F_{m-1} + F_{m-2}$, we see that the number of nontrivial partitions of F_m must equal to that of the number of partitions of F_{m-2}. Then taking into account the trivial partition, $p(F_m) = 1 + p(F_{m-2}) = 1 + \lfloor (m-2)/2 \rfloor = \lfloor m/2 \rfloor$. $\square$

4 Code (Python)

With the above results, the code is very straightforward.

```python
# pe_fibo.py
from functools import cache
import time

@cache
def fibo(n):
    if n==1 or n==2: return 1
    return fibo(n-1) + fibo(n-2)

def p_of_f(k):
    return k//2

def s(n):
    return sum([p_of_f(k) for k in range(2,n+1)])

t0 = time.time()
print(s(123))
print(s(12345))
print(s(123456789))
print("Execution time: %.4fs" % (time.time() - t0))
```

Output:

```
$ python pe_fibo.py
3782
38099756
3810394687547630
Execution time: 15.6884s
```

5 Notes

Any method that iterates all the partitions would be expected to fail due to size of the solution. However, this problem is not very difficult: even if the solver cannot produce the above result, some brute-force experimentation may lead them to guess the pattern of $p(F_n)$, $\{1, 1, 2, 2, 3, 3, \ldots\}$. Nevertheless, I think it is a cute result that many solvers would be inspired to look into further upon solving.