Counterproof of the Daddy conjecture

Ricossa Sergio

April 28, 2021

0.1 Introduction

In the "Papa Flammy's advent Calendar" youtube series, Flammable Math introduced an interesting conjecture about a particular property of the number 52 presented in the video at the following link: `https://www.youtube.com/watch?v=4Uqd9QexPuA`. He conjectured that this property happens infinitely many times among integers.

In this paper, combining analytical and numerical methods, we disprove this conjecture by showing 52 is unique in exhibiting that property. We also prove a more general statement.

This proof is complemented by numerical calculations. All the important files can be found in the following Github repository: `https://github.com/sbacco/daddynessK`.

0.2 Statement of the conjecture

First off, some definitions:

Definition 1. *Let $n \in \mathbb{N}$ such that $n = \sum_{i=1}^{N} d_i \cdot 10^{N-i}$.*

- *The digit sum of n is: $S(n) := \sum_{i=1}^{N} d_i$*
- *The alternate digit sum of n is: $A(n) := -\sum_{i=1}^{N} (-1)^i d_i$*
- *The concatenation $conc(n_1, n_2, n_3, \dots)$ of a list of numbers n_i is the number obtained by putting the n_i one after the other in base 10:*

We write: $conc(n_1, n_2, n_3, \dots) =: n_1 \circ n_2 \circ n_3 \circ \dots$
For example: $12 \circ 4 \circ 123 = 124123$

Definition 2. *Let $n \in \mathbb{N}$. Let: $\sigma := S(n) + A(n)$.*
Let $\sigma = \prod_{i=1}^{\xi} p_i$ the prime factors decomposition of σ, where p_i are ordered from smallest to greatest.
n is said to be pseudo-daddy if: For some permutation $j(i)$ of the indices, one has:

$$n = p_{j(1)} \circ p_{j(2)} \circ \dots \circ p_{j(\xi)}$$

Although A can be negative, it's easy to check that σ is always positive and divisible by 2. One way to obtain it is to sum up every two digits of n and multiply by 2.
For example: $n := 1234 \Rightarrow \sigma = 2(1 + 3) = 8$
Finally:

Definition 3. *Let $n \in \mathbb{N}$.*
n is said to be daddy if it is pseudo-daddy and if both $S(n)$ and $A(n)$ are primes.

The theorem we will prove is the following:

Theorem 1. *There are exactly 5 pseudo-daddy numbers:* $22, 32, 52, 72, 222$
There is only one daddy number: 52

This is proof that the Daddy conjecture that Flammable Math proposed in his video is wrong, as he proposed there are infinitely many daddy numbers.

It's easy to show that the listed numbers are effectively pseudo-daddy. The second statement is also easy to prove. You can just try to evaluate A and S for the 5 numbers above. Only 52 gives 2 primes.

We will focus on showing the first part in this paper.

0.3 Proof of the theorem

Proof. A crucial component of the proof is the program provided with it. We compiled a C++ program that finds through brute force every pseudo-daddies smaller than 10 Million. As one can verify, we immediately find the above mentioned numbers, but no other number.

We therefore just have to show that there are no pseudo-daddies greater than 10 Million.

Let's choose n to be some N-digits pseudo-daddy number, with:

$$n := d_1 \circ d_2 \circ \cdots \circ d_N \ ; \ d_i \in \{0, 1, \ldots, 9\}$$

We also write, for short:

$$S, A := S(n), A(n)$$

We define the prime factors expansion:

$$\sigma = 2 \prod_{i=1}^{\xi} p_i$$

where we separated the factor of 2 from the rest, since we know σ is even.

One has that:

- If N is even: $\sigma = 2 \sum_{i=1}^{N/2} d_{2i-1} \leq 2 \cdot (N/2) \cdot 9$
- If N is odd: $\sigma = 2 \sum_{i=1}^{(N+1)/2} d_{2i-1} \leq (N+1) \cdot 9$

Indeed, as we vaguely discussed above, adding A to S just takes away the even d_i's and doubles the odd ones. So:

$$\sigma = 2 \prod_{i=1}^{\xi} p_i \leq 9(N+1) \tag{1}$$

Moreover:

$$2 \prod_{i=1}^{\xi} p_i \geq 2 \cdot 2^{\xi}$$

Since prime numbers are all greater than 2.

So, sandwiching σ we get that:

$$2^{\xi+1} \leq 9(N+1) \Rightarrow \xi \leq \log_2 (9(N+1)) - 1 \tag{2}$$

So far, we have just done standard manipulations. Now, we use the daddyness condition:

Let k_i be the number of digits of p_i:

$$k_i \leq 1 + \log_{10} p_i$$

For n to be pseudo-daddy we need (the extra '1' comes from the factor of '2' we extracted that has to be concatenated as well):

$$1 + \sum_{i=1}^{\xi} k_i = N$$

$$\Rightarrow N \leq 1 + \xi + \log_{10}(\prod_{i=1}^{\xi} p_i) \leq \log_2 (9(N+1)) + \log_{10}(\frac{9}{2}(N+1)) =: f(N) \tag{3}$$

Where we used both (1) and(2)

We know that N will 'win' over $f(N)$ as N increases. We plotted the two functions on fig.1. They intersect only once at:

$$N_{max} \approx 7.93$$

For $N > N_{max}$, the inequality (3) is violated. Therefore, there are no pseudo-daddies with more than 7 digits.

The outtake is that we only have to bruteforce our way through the first 10,000,000 positive integers to finish the proof. In this time and age, computers are much better than us at doing just that.

By running the annexed program up to $n = 10,000,000$, we get all the existing pseudo-daddy numbers, listed in the theorem's statement. And that concludes the proof. □

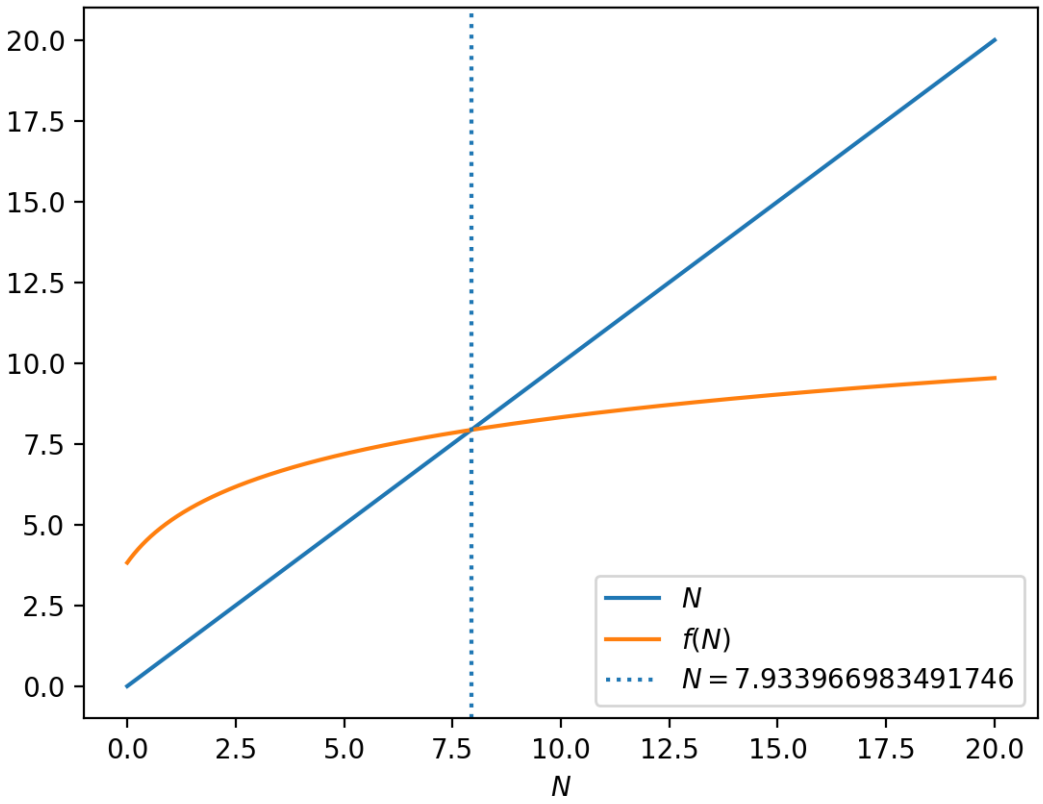

Figure 1: Graphic representation of inequality (3). The inequality is violated when the orange curves goes under the blue curve at $N \approx 7.93$.

0.4 Conclusion

Although this is a valid proof, future improvements are still possible. The 5 only pseudo-daddy numbers are actually quite easy to guess (at least the first four) knowing that they have to contain at least the digit '2' and that $\sigma/2$ being equal to the leftmost digit must be prime on a 2-digit pseudo-daddy.

We are therefore confident that there exist better proofs that don't require testing manually 10,000,000 individual cases.

0.5 Annexes: The c++ code

The following code can be copy pasted and built in c++14.

```
/*This script checks the pseudo-daddyness of numbers.(compiled in c
   ++14)
A number n is daddy if the following conditions are met:

1) The sum of its digits S and the alternating sum of its digits A
   are both primes (requiring A to be positive)
2) Some concatenation of the primes that factor (S+A) is equal to N.

Pseudo daddiess only obey rule 2.

Example: 52 is a daddy number:
S = 5+2 = 7
A = 5-2 = 3
S+A = 7+3 = 10 = 2*5
conc(S+A) = [25 , 52]

The conjecture is that 52 is the only daddy number. This program
   tests up to 10millions, and no other example was found.
*/

#include <iostream>
#include <cmath>
#include <vector>
#include <algorithm>

#define UPTO 10000000
#define MAXN 130 //this is for prime factorisation of S+A, which cant
    go further than 9*2*7 (S+S for n=9999999)

typedef unsigned long num;
typedef std::vector<num> vec;

// C++ class to find prime factorization of a
// number n in O(Log n) time with precomputation
// allowed.
class Primefac
{
public:
       Primefac();
       // A O(log n) function returning primefactorization
       // by dividing by smallest prime factor at every step
       vec operator()(num x);

private:
       // Calculating SPF (Smallest Prime Factor) for every
       // number till MAXN.
       // Time Complexity : O(nloglogn)
```

```
        void sieve();

        vec spf;// stores smallest prime factor for every number
};

long digitsum(num n, bool alternate=false);//sum and alternating sum
   of the digits of n

vec conc(vec lon);//all the concatenations of a list of numbers

//main

int main()
{
        Primefac primefac{};
        for(num n{2} ; n < UPTO ; n++)
        {
                if(!(n%1000000)) std::clog << "Tested␣up␣to␣n␣=␣" << n
                   /1000000 << "␣millions" << std::endl;
                long S{digitsum(n)};
                long A{digitsum(n,true)};

                vec ans{conc(primefac(S+A))};
                for(num i{0} ; i < ans.size() ; i++)
                        if(ans.at(i) == n)
                                std::cout << n << std::endl;
        }

        return EXIT_SUCCESS;
}

//other functions

Primefac::Primefac() {sieve();}

vec Primefac::operator()(num x)
{
    vec ret;
    while (x != 1)
    {
        ret.push_back(spf[x]);
        x = x / spf[x];
    }
    return ret;
}

void Primefac::sieve()
{
    spf.push_back(0); spf.push_back(1);
    for (num i{2}; i<MAXN; i++)
```

```
        // marking smallest prime factor for every
        // number to be itself.
        spf.push_back(i);

    // separately marking spf for every even
    // number as 2
    for (num i{4}; i<MAXN; i+=2)
        spf[i] = 2;

    for (num i{3}; i*i<MAXN; i++)
    {
        // checking if i is prime
        if (spf[i] == i)
        {
            // marking SPF for all numbers divisible by i
            for (num j{i*i}; j<MAXN; j+=i)

                // marking spf[j] if it is not
                // previously marked
                if (spf[j]==j)
                    spf[j] = i;
        }
    }
}

long digitsum(num n, bool alternate)
{
        vec S{};//every digit
        long s{0};//the sum
        num d{0};//the number of digits

        //calculating highest digit position
        if(n == 0) d = 0;
        else d = 1+std::floor(std::log10(n));

        while(d > 0)//isolating digits
        {
                d--;
                S.push_back(std::floor(n/std::pow(10,d)));
                n -= S.at(S.size()-1)*std::pow(10,d);
        }

        for(num i{0} ; i < S.size() ; i++)
        {
                if(alternate) s += std::pow(-1,i)*S.at(i);
                else s += S.at(i);
        }
        return s;
}

vec conc(vec lon)
```

```
{
        vec r{};
        std::sort(lon.begin(),lon.end());
        do{
                r.push_back(0);
                for(num i{0} ; i < lon.size() ; i++)
                {
                        num d{0};
                        if(r.at(r.size()-1) == 0) d = 0;
                        else d = 1+std::floor(std::log10(r.at(r.size()
                           -1)));

                        r.at(r.size()-1) += std::pow(10,d)*lon.at(i);//
                           we concatenate from right to left.
                }
        } while(std::next_permutation(lon.begin(),lon.end()));

        return r;
}
```