Counting Clean Words According to the Number of Their Clean Neighbors

Shalosh B. EKHAD and Doron ZEILBERGER

In fond memory of Marko Petkovšek (1955-2023), a great summer and enumerator

Preface

Our good friend and collaborate, Marko Petkovšek ([PWZ]), passed away on March 23, 2023, and we already wrote a eulogy [Z], and donated to the Online Encyclopedia of Integer Sequences in his memory (See `https://oeisf.org/donate` and search for Petkovsek). However we believe that we can do more than that to commemorate Marko. We looked through his list of publications, and found the delightful article [KMP] by Marko, joint with Sandi Klavžar and Michel Mollard, and realized that the beautiful methodology that they used to solve one very *specific* enumeration problem is applicable to a wide class of enumeration problems of the same flavor. More important, since Marko was such an authority in *symbolic* computation, we decided to **implement** the method, and wrote a Maple package

`https://sites.math.rutgers.edu/~zeilberg/tokhniot/Marko.txt` ,

that can very fast answer these kind of questions. In particular as we will soon see, Theorem 1.1 of [KMP] can be gotten (in its equivalent form in terms of generating functions stated as $f(x, y)$ on top of p. 1321) by typing

`WtEs( {0,1 },{[1,1]},y,x,3);` .

Our Maple package, `Marko.txt`, gives, in 0.057 seconds, the answer

$$-\frac{x^2y^2 - x^2y - xy - 1}{x^3y^2 - x^3y - x^2y - xy + 1} \quad .$$

The Problem Treated so Nicely by Klavžar, Mollard, and Petkovšek

There are 2^n vertices in the n-dimensional unit cube $\{0, 1\}^n$ and every such vertex has **exactly** n neighbors (i.e. vertices with Hamming distance 1 from it). The **Fibonacci lattice** consists of those vertices whose 01 vector **avoids** two consecutive 1s, in other words of words in the alphabet $\{0, 1\}$ avoiding as a **consecutive subword** the two-letter word 11. Such words are called **Fibonacci words**, and there are, not surprisingly, F_{n+2} of them (why?).

Each such word has n neighbors, but some of them are not Fibonacci words. The question answered so elegantly in [KMP] was:

For any given n and k, How many Fibonacci words of length n are there that have exactly k Fibonacci neighbors? Calling this number $f_{n,k}$, [KMP] derived an explicit expression for it, that is equivalent to the generating function (that they also derived)

$$f(x,y) = \sum_{n,k\geq 0} f_{n,k}\, x^n\, y^k = -\frac{x^2y^2 - x^2y - xy - 1}{x^3y^2 - x^3y - x^2y - xy + 1} \quad .$$

They also considered the analogous problem for **Lucas words** that consists of Fibonacci words where the first and last letter can't both be 1. This problem is also amenable to far-reaching generalization, but will not be handled here.

The general Problem

Input:

• A finite alphabet A (In the [KMP] case $A = \{0, 1\}$).

• A finite set of words M, (of the same length) in the alphabet A. (In the [KMP] case M is the singleton set $\{11\}$).

Definition: A word in the alphabet A is called **clean** if it does not have, as *consecutive substring*, any of the members of M.

In other words writing $w = w_1 \dots w_n$, a word is **dirty** if there exists an i such that $w_i w_{i+1} \dots w_{i+k-1} \in M$. For example if $A = \{1, 2, 3\}$ and $M = \{123, 213\}$, then 12212312 is dirty while 111222333 is clean.

To get the set of clean words of length `n` in the alphabet `A` and set of 'mistakes' `M`, type, in `Marko.txt`,

`CleanWords(A,M,n);` .

For example, to get the Fibonacci words of length 3 type:

`CleanWords({ 0,1 },{ [1,1] } , 3);`, getting:

$\{[0, 0, 0], [0, 0, 1], [0, 1, 0], [1, 0, 0], [1, 0, 1]\}$.

The problem of the straight enumeration of clean words is handled very efficiently via the Goulden-Jackson cluster algorithm [NZ], but it is not suitable for the present problem of *weighted* enumeration.

Definition: Two words of the same length in the alphabet A are neighbors if their Hamming distance is 1, in other words, $u = u_1 \dots u_n$ and $v = v_1 \dots v_n$ are neighbors if there exists a location r such $u_i = v_i$ if $i \neq r$ and $u_r \neq v_r$.

For example if $A = \{1, 2, 3\}$, the set of neighbors of 111 is

$$\{211, 311, 121, 131, 112, 113\} \quad .$$

Obviously every word of length n in the alphabet A has $n \cdot (|A|-1)$ neighbors.

However, if w is a clean word, some of its neighbors may be dirty, so if there is one *typo*, it can become dirty, and that would be embarrassing (Oops, *embarrassing* is already dirty). While the word, *duckling* is clean, not all its neighbors are clean.

To see the number of clean neighbors of a word `w` in the alphabet `A` and set of mistakes `M`, type

`NCN(w,A,M);`

Output: Having fixed the (finite) alphabet A, and the finite set of forbidden substrings M (all of the same length), let $f_{n,k}$ be the number of clean words in the alphabet A of length n having k clean neighbors. Compute the bi-variate generating function

$$f(x,y) := \sum_{n,k \geq 0} f_{n,k} x^n \, y^k \quad .$$

It would follow from the algorithm (inspired by the methodology of [KMP], but vastly generalized) that this is always a **rational function** of x and y.

This is implemented in procedure

`WtEs(A,M,y,x,MaxK)`,

where `MaxK` is a 'maximum complexity parameter'. See the beginning of this article for the case treated in [KMP]. For a more complicated example, where a word is clean if it avoids the substrings 000 and 111, type

`WtEs( { 0,1 }, { [1,1,1],[0,0,0] },y,x,5);`

getting, immediately:

$$\frac{2x^5y^4 - 4x^5y^3 + 2x^5y^2 - 2x^4y^3 + 4x^4y^2 - 2x^4y - y^2x^3 + 2x^3y - 4x^2y^2 - x^3 + 2x^2y + x^2 - 2xy + x - 1}{y^2x^3 - x^3 + x^2 + x - 1} .$$

If you want to keep track of the individual letters, rather than just the length, use the more general procedure

`WtEg(A,M,x,y,t,MaxK)`.

Reverse-Engineering the beautiful Klavžar-Mollard-Petkovšek Proof and Vastly Generalizing It

In fact, the authors of [KMP] proved their results in two ways, and only the second way used *generatingfunctionology*. Even that part argued directly in terms of the (double) sequence $f_{n,k}$ itself, and only at the **end of the day**, took the (bi-variate) generating function.

A more efficient, and *streamlined*, approach is to forgo the actual bi-sequence and operate **directly** with weight-enumerators. Let $\mathcal{C}(A, M)$ be the ('infinite') set of words in the alphabet A, avoiding, as consecutive substrings, the members of M, and for each word w in $\mathcal{C}(A, M)$, define the **weight**, $Weight(w)$ by

$$Weight(w) = x^{length(w)} \, y^{NCN(w)} \quad .$$

For example, for the original case of $A = \{0, 1\}$ and $M = \{11\}$,

$$Weight(10101) \; = \; x^5 \, y^3 \quad .$$

We are interested in the **weight-enumerator**

$$f(x, y) := Weight(\mathcal{C}(A, M)) = \sum_{w \in \mathcal{C}(A,M)} Weight(w) \quad .$$

Once you have it, and you are interested in a specific $f_{n,k}$, all you need is to take a Taylor expansion about $(0, 0)$ and extract the coefficient of $x^n y^k$.

Let $\mathcal{C}(A, M)^{(i)}$ be the subset of $\mathcal{C}(A, M)$ of words of length i, and pick a positive integer k. For any word $v \in \mathcal{C}(A, M)^{(k)}$, let $\mathcal{C}_v(A, M)$ be the set of words in $\mathcal{C}(A, M)$ of length $\geq k$ that start with v. Obviously

$$\mathcal{C}(A, M) = \bigcup_{i=0}^{k-1} \mathcal{C}(A, M)^{(i)} \;\cup \bigcup_{v \in \mathcal{C}(A,M)^{(k)}} \mathcal{C}(A, M)_v \quad .$$

We can decompose $\mathcal{C}(A, M)_v$ as follows

$$\mathcal{C}(A, M)_v \; = \; \bigcup_{a \in A} \mathcal{C}(A, M)_{va} \quad ,$$

where, of course $\mathcal{C}(A, M)_{va}$ is empty if appending the letter a turns the clean v into a dirty word. Now, writing $v = v_1 \dots v_k$, and for $a \in A$ the computer verifies whether the difference

$$NCN(v_1 \dots v_k a w) - NCN(v_2 \dots v_k a w)$$

is always the same, for any $v_1 \dots v_k a w \in \mathcal{C}(A_M)_{va}$. The way we implemented it is to test it for sufficiently long words, and then in *retrospect* have the computer check it 'logically', by looking the at the difference in the number of clean neighbors that happens by deleting the first letter v_1. Let's call this constant quantity $\alpha(v, a)$.

It follows that we have a system of $|\mathcal{C}(A, M)^{(k)}|$ equations with $|\mathcal{C}(A, M)^{(k)}|$ unknowns.

$$Weight(\mathcal{C}(A, M)_v) \; = \sum_{\substack{a \in A \\ va \in \mathcal{C}(A,M)}} x y^{\alpha(v,a)} Weight(\mathcal{C}(A, M)_{v_2 \dots v_{k-1} a}) \quad .$$

After the computer algebra system (Maple in our case) automatically found all the $\alpha(v,a)$, and set up the system of equations, we kindly asked it to `solve` it, getting certain rational functions of x and y. **Finally**, our object of desire, $f(x,y)$, is given by

$$Weight(\mathcal{C}(A,M)) \; = \sum_{i=0}^{k-1} Weight(\mathcal{C}(A,M)^{(i)}) \; + \sum_{v \in \mathcal{C}(A,M)^{(k)}} Weight(\mathcal{C}(A,M)_v) \quad .$$

This is implemented in procedure `WtEs(A,M,y,x,MaxK)`.

If you also want to keep track of the individual letters, having the variable t take care of the length, the equations are

$$Weight(\mathcal{C}(A,M)_v) \; = \sum_{\substack{a \in A \\ va \in \mathcal{C}(A,M)}} x_{v_1} t y^{\alpha(v,a)} Weight(\mathcal{C}(A,M)_{v_2 \dots v_{k-1} a}) \quad .$$

This is implemented in procedure `WtEg(A,M,x,y,t,MaxK)`.

Sample output

• If you want to see the bi-variate generating functions for words in the alphabet $\{0,1\}$, avoiding i consecutive occurrences of 1, for $2 \leq i \leq 6$, see

`https://sites.math.rutgers.edu/~zeilberg/tokhniot/oMarko1.txt` .

Note that the original case was $i = 2$.

• If you want to see the bi-variate generating functions for words in the alphabet $\{0,1\}$, avoiding i consecutive occurrences of 1, **and** i consecutive occurrences of 0, for $3 \leq i \leq 6$, see

`https://sites.math.rutgers.edu/~zeilberg/tokhniot/oMarko2.txt` .

• If you want to see all such generating functions (still with BINARY words) for all possible SINGLE patterns of length 3,4,5 (up to symmetry), look at:

`https://sites.math.rutgers.edu/~zeilberg/tokhniot/oMarko3.txt` .

The front of this article contains numerous other output files, but you dear reader, can generate much more!

Conclusion

The value of the article [KMP], that inspired the present article, is not so much with the actual result, that in hindsight, thanks to our Maple package, is trivial, but in the human-generated ideas and **methodology** that enabled one of us to generalize it to a much more general framework.

References

[KMP] Sandi Klavžar, Michel Mollard and Marko Petkovšek, *The degree sequence of Fibonacci and Lucas cubes*, Discrete Math. **311** (2011) 1310-1322.
`https://www-fourier.ujf-grenoble.fr/~mollard/soumis/DegSeqSubmit.pdf`

[NZ] John Noonan and Doron Zeilberger, *The Goulden-Jackson Cluster Method: Extensions, Applications, and Implementations*, J. Difference Eq. Appl. **5** (1999), 355-377.
`https://sites.math.rutgers.edu/~zeilberg/mamarim/mamarimhtml/gj.html` .

[PWZ] Marko Petkovšek, Herbert S. Wilf, and Doron Zeilberger, *"A=B"*, A.K. Peters, 1996.
`https://www2.math.upenn.edu/~wilf/AeqB.pdf` .

[Z] Doron Zeilberger, *Marko Petkošvšek (1955-2023), My A=B Mate* , Personal Journal of Shalosh B. Ekhad and Doron Zeilberger, April 14, 2023.
`https://sites.math.rutgers.edu/~zeilberg/mamarim/mamarimhtml/mpm.html` .

Shalosh B. Ekhad, c/o D. Zeilberger, Department of Mathematics, Rutgers University (New Brunswick), Hill Center-Busch Campus, 110 Frelinghuysen Rd., Piscataway, NJ 08854-8019, USA.
Email: `ShaloshBEkhad at gmail dot com` .

Doron Zeilberger, Department of Mathematics, Rutgers University (New Brunswick), Hill Center-Busch Campus, 110 Frelinghuysen Rd., Piscataway, NJ 08854-8019, USA.
Email: `DoronZeil at gmail dot com` .

April 21, 2023.