The GOULDEN-JACKSON CLUSTER Method For CYCLIC Words

Anne E. EDLIN[1] *and Doron ZEILBERGER*[1]

Abstract: The powerful Goulden-Jackson Cluster Method, that generates generating functions enumerating words that avoid, as factors, a prescribed finite set of "mistakes", is adapted to handle cyclic words (with marked beginnings).

How many n-letter words $w_1 \dots w_n$, in the alphabet $\{A, B, D\}$ avoid the "bad word" BAD?, i.e. for every $i = 1, 2, \dots, n-2$, it may never happen that $w_i = B$, $w_{i+1} = A$, $w_{i+2} = D$. This question is easy. Suppose that you also must avoid ADB and DBA. Then the question is not so easy, and is handled by the powerful Goulden-Jackson method, independently discovered by Guibas and Odlyzko. See [GJ1],[GJ2][GO]. See [NZ] for a detailed exposition and many extensions.

Inspired by the very elegant article [BW], one may ask the following question. Suppose that your words are cyclic, i.e. after the last letter you go back to the beginning, equivalently you are counting n-bead necklaces with a clasp. In this more restrictive problem, one also forbids the following possibility $w_{n-1} = B$, $w_n = A$, $w_1 = D$, as well as the possibility that $w_n = B$, $w_1 = A$, $w_2 = D$. So the words $DAABA$ and $ADAAAB$ that are not bad when viewed as linear words, become bad in the cyclic paradigm.

In order to save bytes (and trees), we will assume that the reader is familiar with [NZ], that is downloadable free of charge from the second author's website.

As in the original Goulden-Jackson method, we can assume, without loss of generality, that no mistake is a factor of another mistake, since if this is the case, we can remove the latter from the list of mistakes, without changing the set of admissible words.

As in the original Goulden-Jackson, consider all words in the alphabet, together with a (possibly empty) subset of its mistakes marked, and assign to this marked word the weight $(-1)^m$, where m is the number of marked mistake.

An in [NZ] we would have zero or more "clusters" usually interspersed with letters that do not belong to any marked mistake. Let L be the cluster generating function for the linear case, as computed in [GJ1] and [NZ]. There are three possibilities.

Case I: The last and first letter of the marked word are not both in any marked mistake. This case is exactly the previous linear case, and the enumerating generating function, in the variable s, is $1/(1 - ks - L)$, where k is the number of letters in our alphabet.

Case II: The last and first letter of the marked word do belong to the same marked mistake, but

[1] Department of Mathematics, Temple University, Philadelphia, PA 19122, USA. `[anne, zeilberg]@math.temple.edu` `http://www.math.temple.edu/~[anne,zeilberg]/` , Feb. 18, 2000. The second author is supported in part by the NSF. This article is accompanied by a Maple package CGJ downloadable from our homepages.

the cluster does not wrap-around itself. In other words the first letter of the cluster (viewed as letter in the underlying word) is strictly after the last letter of that cluster. We can view this situation as a pair consisting of a cluster that is "rooted" at one of its letters, except that the first letter is not allowed to be the root. The generating function for such rooted clusters is $(s\frac{d}{ds} - 1)L$. Now when we remove this distinguished cluster, we are left with a linear marked word, whose generating function was recalled in Case I. So the contribution from Case II is $(s\frac{d}{ds} - 1)L/(1 - ks - L)$.

Case III: There is only one cluster, and all adjacent letters (including the last and the first) share a marked mistake. In this case the cluster "wraps around" itself. For example, if the set of mistakes is $\{BAD, ADB, DBA\}$, and the underlying word is BAD, and all three possible mistakes (consisting of letters 1,2,3 (BAD) , letters 2,3,1 (ADB), and letters 3,1,2 (DBA)) are being marked, belongs to the present case. Even when two out of the three mistakes are marked, we still have a wrap-around cluster, and we are in the present case. If only the mistake that resides in letters 1,2,3 (BAD) is marked, then we are in case I, while, if the only marked mistake is DBA (letters 3,1,2) then we are in case II. We are also in case II when the only marked mistake is ADB (letters 2,3,1).

Let's go back to Case III. We can view a wrap-around cluster as a "cycle", of length at least 2, in the "mistake-digraph". The *mistake-digraph* has vertices labelled by the set of mistakes, and there is a directed edge, between mistake $A = a_1 \dots a_k$ and mistake $B = b_1 \dots b_r$, for each and every time a suffix of A coincides with a prefix of B, i.e. there is a directed edge between A and B for each occurrence of an i, $2 \le i \le k$, such that the word $a_i a_{i+1} \dots a_k$ equals the word $b_1 b_2 \dots b_{k-i+1}$. The *weight* of this edge is $-s^{r-(k-i+1)}$, i.e. s raised to the power "number of letters of B that *stick out*", times -1.

For example there are no edges between BAD and AAA. There in one edge between BAD and DAD, of weight $-s^2$. There are two edges between AAA and itself, of weights $-s$ and $-s^2$. There is one edge between BAD and ADB, of weight $-s$, and one edge between BAD and DBA, of weight $-s^2$.

We claim that any wrap-around cluster corresponds to a cycle in the mistake graph, and the weight of the cluster equals the product of the weights of the participating edges. But these cycles start somewhere, so these are cycles with a marked beginning. We take the starting mistake to be the leftmost mistake that contains both the last letter and the first letter of the underlying word. For example if the set of mistakes if $\{BAD, ADB, DBA\}$, and the underlying word is $BADBAD$, and the marked mistakes are $[6, 1, 2]$ (DBA), $[2, 3, 4]$ (ADB), and $[4, 5, 6]$ (BAD), this gives rise to the cycle $DBA \to ADB \to BAD \to DBA$, of weight $(-s)^2(-s)^2(-s)^2 = -s^6$.

But even that does not uniquely specify the wrap-around cluster. On the last edge of the cycle, that goes from the last mistake to the first mistake, we have to label the "first letter" of the underlying word. This may be any of the "stick-out" letters. For example if the last edge in the cycle goes from 1234 to 34567 (assuming that both 1234 and 34567 are mistakes), whose weight is $-s^3$, because there are three "stick-out" letters: $5, 6, 7$, then we may "root" either 5, 6 or, 7, thereby crowning it the starting letter of the underlying word. In other words, that very last edge should have weight

$-3s^3$ rather than just $-s^3$. In general it has weight $s\frac{d}{ds}(weight)$.

Let's define a matrix A whose rows and columns are labelled by mistakes, by $A[i,j]$:= the sum of the weights of all edges from mistake i to mistake j.

By the definition of matrix product, and its well-known combinatorial interpretation, the weight enumerator of all cycles that start at mistake i and ends at mistake i, with at least two edges, and the last edge marked as above, is the (i,i) entry in the matrix

$$M := \sum_{r=1}^{\infty} A^r \frac{d}{ds} A = A(I-A)^{-1} s \frac{d}{ds} A \quad . \tag{1}$$

We are almost done, except that this gives us some spurious cycles that do not correspond to any wrap-around clusters. Let l_i be the length of the i^{th} mistake. Cycles that start (and hence also end) at the i^{th} mistake, whose weights are $\pm s^m$, for some $m < l_i$, do not correspond to any wrap-around cluster, since the latter should have weight $\pm s^m$, with $m \geq l_i$. So the final step is to remove the first l_i terms from the power-series expansion of M. Let's define

$$Chop_r(\sum_{p=0}^{\infty} a_p s^p) = \sum_{p=r}^{\infty} a_p s^p \quad ,$$

and we get that the contribution to the generating function, from type III, is

$$\sum_{i=1}^{n} Chop_{l_i} M_{i,i} \quad .$$

Combining cases I,II, and III we have

Theorem: The generating function for the set of cyclic words, with a marked beginning, in an alphabet of k letters, avoiding as factors the members of a set of n mistakes, is

$$\frac{1+(s\frac{d}{ds}-1)L}{1-ks-L} + \sum_{i=1}^{n} Chop_{l_i} M_{i,i} \quad ,$$

where L is the cluster generating function computed as in [NZ], and the matrix M is given in (1), and the $n \times n$ matrix $A = A(s)$ is defined as above.

The Maple package `CGJ` (Cyclic Goulden-Jackson), available from either of our homepages, implements this paper. In particular, it easily reproduces the Burstein-Wilf [BW] generating function that enumerates the number of cyclic words, on an alphabet of k letters, that do not have a constant block of length $> w$, in its *full* generality, i.e. for *symbolic* k and w. All you have to do is type `ProveBW();` , and after one nano-second, Maple should respond with `true`.

Burstein and Wilf used their generating function to obtain a very precise exact formula for the actual enumerating sequence. To use Herb Wilf's famous metaphor, they beautifully took the clothes off the clothes-line. We hope that there would be other instances of generating functions, obtainable by our present extension, that would be amenable to an analysis in the style of Burstein and Wilf. In [E], The first-named author intends to generate numerous extensions and applications of the present circle of ideas.

References

[BW] Alexander Burstein and Herbert S. Wilf, *On cyclic strings without long constant blocks*, Fibonacci Quart. **35** (1997), 240-247.

[E] Anne E. Edlin, *Applications and Extensions of the Edlin-Zeilberger Cyclic Adaptation of the Goulden-Jackson Cluster Method*, in preparation.

[GJ1] I. Goulden and D.M. Jackson, *An inversion theorem for cluster decompositions of sequences with distinguished subsequences*, J. London Math. Soc.(2)**20** (1979), 567-576.

[GJ2] I. Goulden and D.M. Jackson, *"Combinatorial Enumeration"*, John Wiley, 1983, New York.

[GO] L.J. Guibas and A.M. Odlyzko, *String overlaps, pattern matching, and non-transitive games*, J. Comb. Theory (ser. A) **30**(1981), 183-208.

[NZ] John Noonan and Doron Zeilberger, *The Goulden-Jackson Cluster Method: Extensions, Applications, and Implementations*, J. Difference Equations and Applications **5** (1999), 355-377.