Automatic Generation of Generating Functions for Enumerating Matchings

By *Shalosh B. EKHAD* and *Doron ZEILBERGER*

Exclusively published in the **Personal Journal of Shalosh B. Ekhad and Doron Zeilberger**:
`http://www.math.rutgers.edu/~zeilberg/pj.html` . Written: April 29, 2011.

Very Important: This article comments on the Maple package
`http://www.math.rutgers.edu/~zeilberg/tokhniot/KamaShidukhim`. Lots of sample input and output can be gotten from the "front" of this article:
`http://www.math.rutgers.edu/~zeilberg/mamarim/mamarimhtml/shidukhim.html` .

Per Hakan Lundow (`http://www.theophys.kth.se/~phl/Text/1factors2.ps.gz`), and Frans Faase (`http://www.iwriteiam.nl/Cpaper.zip`) devised efficient algorithms for computing *generating functions* for the number of *matchings* (both perfect, and not-necessarily perfect) in *grid graphs*, namely the Cartesian product $\mathcal{P}_n \times \mathcal{P}_m$ where $\mathcal{P}_n$ is the path of length n, and more generally, $\mathcal{P}_n \times G$, for *any* graph G. In this article we first reproduce, in Maple, what they did, and further generalize their work to more general infinite families of graphs.

For a graph G, let $S_1(G)$ and $S_2(G)$ be, respectively, the number of *perfect matchings* (sets of pairwise vertex-disjoint edges that cover all the vertices) and (all) *matchings* (sets of pairwise vertex-disjoint edges). It is well-known and fairly easy to see (and is implied by the algorithm outlined below, that is implemented in `KamaShidukhim`) that for any *fixed* positive integer m, the generating functions

$$\sum_{n=0}^{\infty} S_1(\mathcal{P}_m \times \mathcal{P}_n) z^n \quad , \quad \sum_{n=0}^{\infty} S_2(\mathcal{P}_m \times \mathcal{P}_n) z^n$$

are both *rational functions* of z. The Maple package `KamaShidukhim` automatically computes these rational function for *any* inputted numeric m. See procedure `GFrect(m,z)` and `GFrectMD(m,z)` of `KamaShidukhim`.

In fact we do something much more general. For *any* graph G, `KamaShidukhim` can (explicitly!) compute the rational generating functions

$$\sum_{n=0}^{\infty} S_1(\mathcal{P}_n \times G) z^n \quad , \quad \sum_{n=0}^{\infty} S_2(\mathcal{P}_n \times G) z^n \quad .$$

See procedures `GFG(G,m,z)` and `GFGmd(G,m,z)` of `KamaShidukhim`.

In fact we do something *enormously* more general! For *any* graph G, on m vertices, and for *any* bipartite (m,m) graph C, let $M_n(G,C)$ be the graph on mn vertices where the edges among

$$1+im \, , \, 2+im \, , \, \dots \, , \, m+im \quad ,$$

for $i = 0, \ldots, n-1$, mimic the graph G , and in addition the edges between

$$1 + im \, , \, 2 + im \, , \, \ldots \, , \, m + im$$

and

$$1 + (i+1)m \, , \, 2 + (i+1)m \, , \, \ldots \, , \, m + (i+1)m$$

$(0 \leq i < n-1)$ mimic the edges of C, given as a set of (up to m^2) ordered pairs $\{[\alpha, \beta]\}$. $[\alpha, \beta] \in C$ means that there is an edge between vertex $\alpha + im$ and vertex $\beta + (i+1)m$ for $0 \leq i < n-1$. Note that when C is the monogamy bipartite graph $\{[1,1], \ldots, [m,m]\}$, where Mr i is connected to Mrs i (but no cheating!), then $M_n(G, C)$ reduces to the Cartesian product $G \times \mathcal{P}_n$.

`KamaShidukhim` can (explicitly!) compute the rational functions (of z):

$$\sum_{n=0} S_1(M_n(G,C)) z^n \quad , \quad \sum_{n=0} S_2(M_n(G,C)) z^n .$$

See procedures `GFt(G,C,z)` and `GFtMD(G,C,z)` of `KamaShidukhim`.

The Method

Of course we use the *transfer matrix method*. When we do match-making, we must first decide whom amongst those vertices of $M_n(G, C)$ that live on the "bottom floor" ("oldest") copy of G would be connected to each other and how to decide on these intra-generational pairings within that oldest level. Having done that, we have to decide how to match the remaining, not-yet-matched vertices (still at the oldest copy of G) to some vertices on the next floor, via the edges of the copy of C. After these inter-generational matchings have been decided, some vertices of the second copy of G (on the second level of $M_n(G, C)$) are already committed to a relationship to someone older, but the remaining vertices can be either matched to someone in their own generation, or to someone in the next-generation-copy of G, etc. So it naturally emerges that we have to tackle the more general problem of enumerating "chopped matchings" where the bottm copy of G has a prescribed subset of vertices that are already matched.

As the computer does it, it dynamically builds the *set of states*, a certain collection of subsets of $\{1, \ldots, m\}$, and there is no need for human "ingenuity" to "figure-out" the set of states.

The computer also, *all by itself*, figures out the *transition matrix*. Then it *automatically* sets-up the system of linear equations (involving the variable z), and *automatically* goes on to solve them, *symbolically!*, and at the end gets the desired quantity, the generating function of the state $\{1, ..., m\}$. Please see the Maple source-code of procedures `GFt(G,C,z)` and `GFtMD(G,C,z)` of `KamaShidukhim` for more details.

Shalosh B. Ekhad and Doron Zeilberger, Department of Mathematics, Hill Center-Busch Campus, Rutgers University, 110 Frelinghuysen Rd, Piscataway, NJ 08544, USA.

Email: `[c/o zeilberg, zeilberg]@math.rutgers.edu` .