In How Many Ways Can n (Straight) Men and n (Straight) Women Get Married, if Each Person Has Exactly k Spouses?

Shalosh B. EKHAD and Doron ZEILBERGER[1]

More boringly, how many (n, n) (labeled) bipartite graphs are there that are regular of degree k? Even more boringly, how many $n \times n$ $0-1$ matrices are there with exactly k 1's in each row and each column?

This is a venerable problem in combinatorics. For $k = 1$ it goes back to Rabbi Levi Ben Gerson's *classic* classic, *Sepher Ma'asei Choshev*, from almost 800 years ago. For $k = 2$ and $k = 3$ it is treated in Louis Comtet's *modern* classic, *Advanced Combinatorics* (section 6.3), from almost 40 years ago. Comtet called them *multipermutations*. They even make a brief appearance in Richard Stanley's *postmodern* classic, *Enumerative Combinatorics*, from almost seven years ago (Cor. 5.5.11).

Ira Gessel, in his seminal paper "*Symmetric Functions and P-Recursiveness*", J. Comb. Theory, Series A, **53** (1990), 257-285, proved that these sequences, for any fixed k, are P-recursive.

The asymptotics of these sequences, let's call them $\{P_k(n)\}$, for *any fixed* k, was determined by C. J. Everett, Jr. and P.R. Stein in their article "*The asymptotic number of integer stochastic matrices*", Discrete Mathematics **1** (1971), 33-72. In an amazing tour-de-force, the Asymptotics Wizard Rodney Canfield, and the Creative Debunker Brendan McKay (who, in addition to his debunking activities is an equally great asymptotician) extended the range of (k, n) much further. See their paper: *Asymptotic enumeration of 0-1 dense matrices with equal row-sums and equal column-sums*, Elec. J. Combinatorics, **12(1)** (2005), R29. In that paper, they also designed an efficient algorithm for computing these numbers *exactly* for *small* (and not-so-small!) values of k and n.

The sequences $P_k(n)$, for small values of k, are in Sloane. The case $k = 2$ is $A001499$, while the case $k = 3$ is $A001501$. Both of these have already made it to the **first** (paper) edition. The cases $k = 4$ and $k = 5$ are $A058528$ and $A075754$ respectively.

Our goal is modest. We keep k small. After all, we are not King Solomon. But we would like to know *exactly*, (not asymptotically!), how many ways it can be done with n very large, say $n = 10000$.

Traditionally, in order to tackle such a problem, one would *think*, for each k, about the structure of the set to be counted. Then, *by hand*, derive a *recurrence*, then, *by hand* program the recurrene in Fortran or C, and finally crank out the first 10000 terms.

---

[1] Department of Mathematics, Rutgers University (New Brunswick), Hill Center-Busch Campus, 110 Frelinghuysen Rd., Piscataway, NJ 08854-8019, USA. `zeilberg at math dot rutgers dot edu` ,
`http://www.math.rutgers.edu/~zeilberg` . First version: Dec. 29, 2006. This version: Jan. 11, 2007. Accompanied by Maple packages `Bipartite` and `LatinRectangles` downloadable from Zeilberger's website. Supported in part by the NSF.

It is easy to see that the quantity of interest is the coefficient of $x_1^k \cdots x_n^k$ in $e_k(x_1, \dots, x_k)^n$, where $e_k(x_1, \dots, x_n)$ is the elementary symmetric function of degree k. A natural approach would be to use the obvious recurrence

$$e_k(x_1, \dots, x_n) = e_k(x_1, \dots, x_{n-1}) + x_n e_{k-1}(x_1, \dots, x_{n-1}) \quad ,$$

raise it to the n^{th} power, and collect the coefficient of x_n^k. Alas this forces us to consider expressions of the form $e_k^{n-k} e_{k-1}^k$, that in turn, force us to consider expressions of the form ${e_k}^{a_k} e_{k-1}^{a_{k-1}} e_{k-2}^{a_{k-2}}$, and before we know it, we have to consider expression of the form ${e_k}^{a_k} e_{k-1}^{a_{k-1}} \cdots {e_1}^{a_1}$. So let's be more broad-minded, and define $F_k(n; a_1, \dots, a_k)$ to be the coefficient of $x_1^k \cdots x_n^k$ in

$$e_k(x_1, \dots, x_n)^{a_k} e_{k-1}(x_1, \dots, x_n)^{a_{k-1}} \cdots e_1(x_1, \dots, x_n)^{a_1} \quad , \tag{1}$$

where, of course, $a_1 + 2a_2 + \dots + ka_k = kn$. Our object of desire is $F_k(n; 0, \dots, 0, n)$, but we have to put up with the $O(n^{k-1})$ extra uninvited guests.

The next step is to derive a *recurrence* for $F_k(n; a_1, \dots, a_k)$. Plug-in

$$e_i(x_1, \dots, x_n) = e_i(x_1, \dots, x_{n-1}) + x_n e_{i-1}(x_1, \dots, x_{n-1}) \quad ,$$

for $i = 1 \dots k$ (with $e_0 = 1$) into (1), expand with respect to x_n, and extract the coefficient of x_n^k. This would be a linear combination, with coefficients that are polynomials in $(a_1, \dots, a_k)$, of terms like $F_k(n-1; a_1 - \alpha_1, \dots, a_k - \alpha_k)$, for various shifts $(\alpha_1, \dots, \alpha_k)$. In other words, we have a *linear partial recurrence operator with polynomial coefficients* $\mathcal{P}_k(a_1, \dots, a_k, A_1^{-1}, \dots, A_k^{-1})$, (where A_i is the shift operators in a_i: $A_i f(a_i) := f(a_i + 1)$), such that

$$F_k(n; a_1, \dots, a_k) = \mathcal{P}_k(a_1, \dots, a_k; A_1^{-1}, \dots, A_k^{-1}) F_k(n-1; a_1, \dots, a_k) \quad ,$$

and this gives an $O(n^k)$ algorithm to compute the F_k's, in particular, $P_k(n) = F_k(n; 0, 0, \dots, n)$.

The beauty of our approach is that the above operators, for any fixed k, can be found automatically by the first author, once the second author spent half an hour programming the above algorithm for computing $\mathcal{P}_k$. Once the first author finishes the *symbol-crunching*, it goes on to use it for *number-crunching*. So everything, except for the programming, is seamless and *automatic*.

It is just as easy to count $n \times n$ matrices whose entries are non-negative integers $\leq r$. Just use:

$$e_k^{(r)}(x_1, \dots, x_n) = \sum_{j=0}^{r} x_n^j e_{k-j}^{(r)}(x_1, \dots, x_{n-1}) \quad ,$$

and define $F_k^{(r)}$ analogously, getting the appropriate operators $\mathcal{P}_k^{(r)}$. If $r = k$ then it is the classic problem of counting $n \times n$ matrices with entries that are non-negative integers, whose row- and column- sums are all k.

All of this is implemented in the Maple package `Bipartite` available from the webpage of this article: `http://www.math.rutgers.edu/~zeilberg/mamarim/mamarimhtml/bipartite.html` , where there is ample sample output.

Final Remarks

1. For each fixed k, the above $O(n^k)$ algorithm can be used to get (automatically!), an $O(n)$ algorithm! Use the previous algorithm to crank-out enough terms, and then use Bruno Salvy and Paul Zimmermann's useful and versatile Maple package `gfun` to find a linear recurrence with polynomial coefficients, that we know exists thanks to Ira Gessel, and that can be *a posteriori* proved rigorously within the *holonomic ansatz.* The catch is that the implied constant in the $O(n)$ grows very fast with k. More seriously, *guessing* the recurrence for larger k, takes for ever, since we need so many terms.

2. A more sophisticated symmetric functions technique can be found in the very interesting pioneering paper by Ian Goulden, David Jackson, and J. W. Reilly, "*The Hammond Series of a Symmetric Function and Its Applications to P-recursivenss*", SIAM J. Alg. Disc. Meth. **4** (1983), 179-193. It would be interesting to compare it with our naive method.

3. An analogous approach can be used to find recurrence operators for computing the enumerating sequences for k by n Latin Rectangles. In another remarkable paper, Ira Gessel, "*Counting Latin Rectangles*", Bulletin (New Series) of the Amer. Math. Soc. **16** (1987), 79-82, proved that these too are P-recursive, for each fixed k. However his approach, while very elegant, was *negative*, using inclusion-exclusion (alias Möbius functions).

For the sake of exposition, let's take $k = 3$. The general case can be filled-in by the reader. Let

$$e_n(x_1, \dots, x_n; y_1, \dots, y_n; z_1, \dots, z_n) = \sum x_i y_j z_k \quad ,$$

where the sum is over all ordered triples (i, j, k) of *distinct* integers drawn from $\{1, \dots, n\}$.

The number of 3 by n Latin Rectangles is the coefficient of $(x_1 \cdots x_n)(y_1 \cdots y_n)(z_1 \cdots z_n)$ in

$$e_n(x_1, \dots, x_n; y_1, \dots, y_n; z_1, \dots, z_n)^n \quad .$$

The obvious recurrence for e_n is

$$e_n(\mathbf{x}, \mathbf{y}, \mathbf{z}) = e_{n-1}(\mathbf{x}, \mathbf{y}, \mathbf{z}) + x_n e_{n-1}(\mathbf{y}, \mathbf{z}) + y_n e_{n-1}(\mathbf{x}, \mathbf{z}) + z_n e_{n-1}(\mathbf{x}, \mathbf{y}) \quad . \qquad (Recurrence)$$

Plugging this above, forces us to consider the coefficient of $(x_1 \cdots x_n)(y_1 \cdots y_n)(z_1 \cdots z_n)$ in the product

$$e_n(\mathbf{x}, \mathbf{y}, \mathbf{z})^{a_{123}} e_n(\mathbf{x}, \mathbf{y})^{a_{12}} e_n(\mathbf{x}, \mathbf{z})^{a_{13}} e_n(\mathbf{y}, \mathbf{z})^{a_{23}} \quad ,$$

that, in turn, will force us to consider the coefficient of $(x_1 \cdots x_n)(y_1 \cdots y_n)(z_1 \cdots z_n)$ in

$$e_n(\mathbf{x}, \mathbf{y}, \mathbf{z})^{a_{123}} e_n(\mathbf{x}, \mathbf{y})^{a_{12}} e_n(\mathbf{x}, \mathbf{z})^{a_{13}} e_n(\mathbf{y}, \mathbf{z})^{a_{23}} e_n(\mathbf{x})^{a_1} e_n(\mathbf{y})^{a_2} e_n(\mathbf{z})^{a_3} \quad ,$$

let's call it $F(a_{123}, a_{12}, a_{13}, a_{23}, a_1, a_2, a_3)$. Using (*Recurrence*) and its analog, expanding, and collecting coeff. of $x_n y_n z_n$, we can express it as a linear combination, with coefficients that are polynomials in the a's of $F(a_{123}, a_{12}, a_{13}, a_{23}, a_1, a_2, a_3)$'s with smaller arguments. This gives an effective way to compute these, and in particular our object of desire: $F(n, 0, 0, 0, 0, 0, 0)$.

For general k, we need functions defined on *pseudo-boolean functions* $\{a_T \,|\, T \subset \{1, \dots k\}\}$. If $E[T]$ is the shift operator in a_T, then the required linear recurrence operator is the coeff. of $x_1 \cdots x_k$ in

$$\prod_{T \subset \{1,\dots,k\}} (1 + \sum_{i \in T} x_i E[T \backslash i] E[T]^{-1})^{a_T} \quad ,$$

which for any fixed k, can be computed automatically, and then used to crank out any desired number of terms. For any fixed k, it is "polynomial-time" $O(n^{\alpha_k})$, *but* the α_k is **huge**: $2^k - k$.

This is implemented in the other Maple package accompanying this article, called `LatinRectangles`, also available from the webpage of this article:

`http://www.math.rutgers.edu/~zeilberg/mamarim/mamarimhtml/bipartite.html` ,

where there is some sample output.