Using Rota's Umbral Calculus to Enumerate Stanley's P-Partitions

Shalosh B. EKHAD and Doron ZEILBERGER[1]

Dedicated to Two Combinatorial Giants: Gian-Carlo Rota (April 27, 1932- April 18, 1999) ZTz"L and Richard Stanley (b. June 23, 1944) ShLIT"A on their 75th and 63rd birthdays, respectively.

Abstract. Gian-Carlo Rota's powerful Umbral Calculus is used to enumerate large families of Richard Stanley's P-Partitions.

Compositions and Partitions

A *composition* with n parts is a vector of non-negative integers $(a_1, \ldots, a_n)$. Defining the **weight** of such a composition to be $x_1^{a_1} \ldots x_n^{a_n}$, we easily get that the *weight-enumerator*, alias **generating function**, of **all** compositions into n parts is:

$$\sum_{a_1 \geq 0, \ldots, a_n \geq 0} x_1^{a_1} \cdots x_n^{a_n} =$$

$$\left(\sum_{a_1=0}^{\infty} x_1^{a_1} \right) \cdots \left(\sum_{a_n=0}^{\infty} x_n^{a_n} \right) = \frac{1}{1-x_1} \cdots \frac{1}{1-x_n} \quad .$$

Leaving the generating function in terms of $x_1, \ldots, x_n$ is really displaying the whole (infinite!) set of compositions into n parts. Setting all the x_i's to be q we get that the generating function of $c_n(m) :=$ number of compositions of m into exactly n non-negative parts, $\sum_{m=0}^{\infty} c_n(m) q^m$, equals $(1-q)^{-n}$.

A *partition* with n parts is an integer-vector $(a_1, \ldots, a_n)$ with $0 \leq a_1 \leq a_2 \leq \ldots \leq a_n$. Defining the **weight** of such a partition to be $x_1^{a_1} \ldots x_n^{a_n}$, we easily get that the *weight-enumerator*, alias **generating function**, of **all** partitions into n parts is:

$$\sum_{0 \leq a_1 \leq a_2 \leq \ldots \leq a_n} x_1^{a_1} \cdots x_n^{a_n} = \sum_{0 \leq a_1 \leq a_2 \leq \ldots \leq a_n} (x_1 \cdots x_n)^{a_1} (x_2 \cdots x_n)^{a_2 - a_1} (x_3 \cdots x_n)^{a_3 - a_2} \cdots x_n^{a_n} =$$

$$= \left(\sum_{a_1=0}^{\infty} (x_1 \cdots x_n)^{a_1} \right) \left(\sum_{a_2 - a_1 = 0}^{\infty} (x_2 \cdots x_n)^{a_2 - a_1} \right) \left(\sum_{a_3 - a_2 = 0}^{\infty} (x_3 \cdots x_n)^{a_3 - a_2} \right) \cdots \left(\sum_{a_n=0}^{\infty} (x_n)^{a_n} \right)$$

$$= \frac{1}{1 - x_1 \cdots x_n} \cdot \frac{1}{1 - x_2 \cdots x_n} \cdots \frac{1}{1-x_n} = \frac{1}{(1 - x_1 \cdots x_n)(1 - x_2 \cdots x_n) \cdots (1 - x_n)} \quad .$$

[1] Department of Mathematics, Rutgers University (New Brunswick), Hill Center-Busch Campus, 110 Frelinghuysen Rd., Piscataway, NJ 08854-8019, USA. `zeilberg@math.rutgers.edu` , `http://www.math.rutgers.edu/~zeilberg`.
First version: June 12, 2007. Accompanied by Maple package `RotaStanley` downloadable from Zeilberger's website. Supported in part by the NSF.

AMS subject classification: 05A15, 05A40.

Key words: Umbral Calculus, P-Partitions, Symbolic Computation.

Leaving the generating function in terms of $x_1, \ldots, x_n$ is really displaying the whole (infinite!) set of partitions into n parts. Setting all the x_i's to be q we get that the generating function of $p_n(m) :=$ number of partitions of m into exactly n non-negative parts, $\sum_{n=0}^{\infty} p_n(m) q^m$, equals $1/((1-q)(1-q^2)\ldots(1-q^n))$.

Stanley's P-Partitions

Compositions and Partitions are the two **extremes** of non-negative integer arrays, where in the former there is **no order** at all imposed, while in the latter there is a **total** order.

In 1972, Richard Stanley[S1], standing on the shoulders of Percy MacMahon[M] and Don Knuth[K], defined a whole new class of objects that bridges between these two extremes. He coined them *P-Partitions*. They concern *partially-ordered sets*, called *posets* for short.

Recall that a poset P is a finite set of vertices, that we will name $1, 2, \dots, n$, together with a *partial-order* $\leq_P$, satisfying the *axioms* $x \leq_P y$ and $y \leq_P x$ implies $x = y$ and $x \leq_P y$ and $y \leq_P z$ implies $x \leq_P z$. The labeling is called *natural* if $x \leq_P y$ implies $x \leq y$.

We will represent a naturally-labeled poset with n vertices by a sequence of n sets $[S_1, \dots, S_n]$ where S_i is the set of $j \in P$ such that $j < i$ and $j \leq_P i$. Of course $S_1 = \emptyset$ always. For example, for the empty order (yielding compositions) we have $P = [\emptyset, \emptyset, \dots, \emptyset]$, for the total order (yielding partitions) we have $P = [\emptyset, \{1\}, \{1, 2\}, \dots, \{1, 2, \dots, n-1\}]$, for the "diamond" (or two-dimensional unit cube) we have $P = [\emptyset, \{1\}, \{1\}, \{1, 2, 3\}]$ while for the three-dimensional unit cube, we have

$$P = [\emptyset, \{1\}, \{1\}, \{1\}, \{2, 3\}, \{1, 3\}, \{1, 2\}, \{1, 2, 3, 4, 5, 6, 7\}] \quad .$$

Given a poset P on $\{1, \dots, n\}$, a P-partition is a vector of non-negative integers $(a_1, \dots, a_n)$ such that $i \leq_P j$ implies $a_i \leq a_j$. For example, if $P = [\emptyset, \{1\}, \{1\}, \{1, 2, 3\}]$, then $(0, 2, 4, 5)$ and $(0, 4, 2, 5)$ are both P-partitions but $(0, 4, 2, 3)$ is not.

Once again we define the **weight** of a P-partition $(a_1, \dots, a_n)$ to be $x_1^{a_1} \cdots x_n^{a_n}$, and given *any* naturally-labeled poset P, we would like to compute the *weight-enumerator*, alias **generating-function**, of all its P-partitions.

Richard Stanley[S1] (nicely described in sec. 4.4 of the classic [S2]) reasons as follows. [2]

Let S_n be the set of permutations on $\{1, 2, \dots, n\}$.

For any permutation π we define the set A_π of vectors of non-negative integers $(a_1, \dots, a_n)$ such that $a_{\pi_1} \leq a_{\pi_2} \leq \dots \leq a_{\pi_n}$ and **in addition** $a_{\pi_i} < a_{\pi_{i+1}}$ whenever $\pi_i > \pi_{i+1}$.

It is easy to see that the set of non-negative integer vectors of length n, $\mathcal{N}^n$, can be partitioned into a *disjoint* union

$$\mathcal{N}^n = \bigcup_{\pi \in S_n} A_\pi \quad .$$

A permutation $\pi = \pi_1 \dots \pi_n$ is *compatible* with P if $i < j$ and $i \leq_P j$ implies that i lies to the left of j in π.

[2] Note that Stanley defines P-partitions to be order-reversing, while we require that they'll be order-preserving, so to pass from his convention to ours one has to reverse the poset.

For example, for $P = [\emptyset, \{1\}, \{1\}, \{1,2,3\}]$, there are only two compatible permutations: 1234 and 1324.

The weight-enumerator of each individual A_π is easy to compute. It is

$$\sum (x_{\pi_1})^{a_{\pi_1}} \cdots (x_{\pi_n})^{a_{\pi_n}}$$

where the sum ranges over all $(a_1, \dots, a_n)$ in $\mathcal{N}^n$ such that $a_{\pi_1} \leq a_{\pi_2} \leq \dots \leq a_{\pi_n}$ *and* whenever $\pi_i > \pi_{i+1}$, $a_{\pi_i} < a_{\pi_{i+1}}$. These are easily summable geometrical series that the computer can do by itself:

$$h_\pi(x_1, \dots, x_n) := \frac{M_\pi(x_1, \dots, x_n)}{(1 - x_{\pi_1} \cdots x_{\pi_n})(1 - x_{\pi_2} \cdots x_{\pi_n}) \cdots (1 - x_{\pi_n})} \quad ,$$

for some easily-computable monomial $M_\pi(x_1, \dots, x_n)$.

Clearly the set of P-partitions is the disjoint union of the A_π's for the π compatible with P, so the weight-enumerator of all P-partitions is simply the sum of the h_π's for the π's that are compatible with P.

Once again the generating function $F_P(x_1, \dots, x_n)$ contains *all* the information about the P-partitions of P, since the set of monomials in its Maclaurin expansion is in an obvious one-to-one correspondence with the set of P-partitions of P.

If one is only interested in the number of P-partitions of m, then it is the coefficient of q^m in the uni-variate rational function $f_P(q) := F_P(q, q, q, \dots, q)$ obtained by setting all the x_i's to q.

The Exponential Curse

Stanley's approach works well for small posets. But once the number of vertices exceeds 10, the set of compatible permutations is too large, and it becomes hopeless to compute F_P and hence f_P directly. But, in mathematics as well as in life, if things get too complicated, it is sometimes possible to **decompose** them into smaller components, by finding some operation of **gluing** or *grafting*.

Grafting

Let P be a poset , with n vertices, naturally labeled $\{1, 2, \dots, n\}$, and let Q be another poset, with m vertices, naturally labeled $\{1, 2, \dots, m\}$. Suppose that the subposet of P induced by its *last* k vertices is isomorphic to the subposet of Q induced by its *first* k vertices. Then we can form a **new** poset, the k-graft of P and Q, let's call it R and denote it by $g(P, Q; k)$, by first promoting the labels of Q by $n - k$, so that its labels will be $\{n - k + 1, \dots, n - k + m\}$ rather than the original $\{1, \dots, m\}$, and then *identifying* the last k vertices of P with the first k vertices of Q.

An example of Grafting

If $P = [\emptyset, \{1\}, \{1\}, \{1,2,3\}, \{1,2,3\}]$ and $Q = [\emptyset, \{1\}, \{1\}, \{1,2,3\}, \{1,2,3\}, \{1,2,3,4,5\}]$, and $k = 3$, the induced subposet of the first 3 vertices of Q is $[\emptyset, \{1\}, \{1\}]$, while that of the last 3 vertices of P is $[\{1\},\{1,2,3\},\{1,2,3\}]$ with $\{1,2\}$ removed, that gives $[\emptyset,\{3\},\{3\}]$, and "normalizing", we get $[\emptyset,\{1\},\{1\}]$. The reader can easily convince itself that

$$g(P,Q;3)=[\emptyset,\{1\},\{1\},\{1,2,3\},\{1,2,3\},\{1,2,3,4,5\},\{1,2,3,4,5\},\{1,2,3,4,5,6,7\}] \quad .$$

If you know the generating functions for P-partitions of P and Q, can you compute that of Their Graft?

Not directly! But let's try and ponder how a typical P-partition of the graft R looks like. We can split the decision of how to construct it into two *phases*. First we construct a P-partition of P, and then we look at the last k vertices of P and think **how to extend them** to the remaining $m-k$ vertices of Q. So we need a more **general** notion than the generating function for P-partition, that weight-enumerates those P-partitions whose first k vertices have already been assigned values.

For the sake of simplicity, let's first assume that the grafting region, of the common k vertices (the last of P and the first of Q), are *totally-ordered*. From the point of view of Q, its first k vertices are already committed, and it remains to decide the fate of the remaining $m-k$ vertices. So for each weakly-increasing vector of integers $(a_1,\dots,a_k)$ we need

$$S_Q(a_1,\dots,a_k)(x_{k+1},\dots,x_m)=\sum_{q=(a_1,\dots,a_k,q_{k+1},\dots,q_m)} {x_{k+1}}^{q_{k+1}}{x_{k+2}}^{q_{k+2}}\cdots {x_m}^{q_m} \quad ,$$

where the sum is over all P-partitions $q=(q_1,\dots,q_m)$, of Q, for which $q_1=a_1,\dots,q_k=a_k$. And surprise! The computer can compute it just as easily as it computed F_Q, and the answer will be a rational function whose denominator is a polynomial in $x_{k+1},\dots,x_m$ but its numerator is a polynomial in $x_{k+1},\dots,x_m$ as well as in 'symbolic' monomials formed from them.

Note that the $(a_1,\dots,a_k)$ do not have to be numerical but can stay *symbolic*. It $(a_1,\dots,a_k)$ is a symbolic weakly-increasing sequence of non-negative integers, then there is an **explicit** formula for $S_Q(a_1,\dots,a_k)(x_{k+1},\dots,x_m)$, that the computer can easily find by summing (iterated) *symbolic* infinite geometrical series. Combining, we have

$$F_R(x_1,\dots,x_{m+n-k})=\sum_{0\le a_1\le a_2\le\dots\le a_k}\ \sum_{(\mathrm{p}_1,\dots,\mathrm{p}_{n-k},\mathrm{a}_1,\dots,\mathrm{a}_k)\ \text{is a P-Partition of P}}$$

$$x_1^{p_1}\cdots x_{n-k}^{p_{n-k}}x_{n-k+1}^{a_1}\cdots x_n^{a_k}\cdot S_Q(a_1,\dots,a_k)(x_{n+1},\dots,x_{n+m-k}) \quad . \qquad (GianCarlo)$$

Gian-Carlo Rota's Umbral Miracle Comes to the Rescue

Gian-Carlo Rota's Umbral Calculus [RR] was already exploited in [Z1-5]. Here we will describe how to use it for computing generating functions for P-partitions. We refer the readers to [Z1] for background about Umbral operators.

Going back to Eq. (GianCarlo), it is now natural to define an *operator* U_Q, on the ring of formal power series of $(y_1,\dots,y_k)$ by defining it on monomials by:

$$U_Q(y_1^{a_1}\cdots y_k^{a_k}):=S_Q(a_1,\dots,a_k)(x_{n+1},\dots,x_{n+m-k}) \quad ,$$

and *extend it by linearity.* It turns out that U_Q is an *Umbral operator*, i.e. of the form

$$f(y_1,\dots,y_k) \to \sum_{j\in J} R_j(x_{n+1},\dots,x_{m+m-k}) f(m_1^{(j)},\dots,m_k^{(j)}) \quad ,$$

where J is a certain finite index-set, R_j are rational functions of their arguments, and $m_1^{(j)},\dots,m_k^{(j)}$ are some specific *monomials* in $(x_{n+1},\dots,x_{m+n-k})$. The beauty is that the computer can figure out this rather abstract object, an *Umbral operator*, **all by itself**, using a certain **data structure** described in [Z1]. Going back to $(GianCarlo)$, we get, by the **linearity** of U_Q

$$F_R(x_1,\dots,x_{m+n-k}) = \sum_{0\le a_1\le a_2\le\dots\le a_k} \quad \sum_{(p_1,\dots,p_{n-k},a_1,\dots,a_k)\ is\ a\ P-Partition\ of\ P}$$

$$x_1^{p_1}\cdots x_{n-k}^{p_{n-k}} x_{n-k+1}^{a_1}\cdots x_n^{a_k}\, U_Q(y_1^{a_1}\cdots,y_k^{a_k})$$

$$= U_Q\left[\sum_{0\le a_1\le a_2\le\dots\le a_k} \quad \sum_{(p_1,\dots,p_{n-k},a_1,\dots,a_k)\ is\ a\ P-Partition\ of\ P} x_1^{p_1}\cdots x_{n-k}^{p_{n-k}}(y_1x_{n-k+1})^{a_1}\cdots(y_kx_n)^{a_k}\right]$$

$$= U_Q\left[F_P(x_1,\dots,x_{n-k}\,,\,y_1x_{n-k+1}\,,\,\dots\,,\,y_kx_n)\right] \quad .$$

So in order to compute the generating function for P-partitions of the k-graft of P and Q (assuming for now that the k common vertices are totally ordered), we need to ask P for its F_P, but from Q we need more, namely, the Umbral operator U_Q, with input variables $y_1,\dots,y_k$, and output variables $x_{k+1},\dots,x_m$, that we later have to shift (in order to accommodate the re-labeling implied by the graft) to $x_{n+1},\dots,x_{n+m-k}$. Fortunately, computer algebra systems (Maple in our case) can compute this *operator* almost as easily as computing the *rational function* F_Q.

What if the Interface is Not Totally Ordered?

Then one must find all the linear extensions of the induced common poset of P's last k vertices and Q's first k vertices. First, we must replace F_P by a vector of rational functions, each of whose components correspond to one of the linear extensions of the intersection. Then for each of these linear extensions, we must find its own Umbral operator, getting a vector of Umbral operators. Then, in order to get F_R, we take the "dot product" so-to-speak. If Q will be later grafted to yet another poset, as part of a chain (see below), then we must also look at all the possible "output states", getting an **Umbral matrix** connecting the input states to the output states. Once again the computer does if all by itself. For the technical/formal details, we refer the reader to the Maple package `RotaStanley`, where this is implemented by procedure `PPumbraMatrix`.

Chains of Grafted Posets

Every poset P can be naturally described (usually in numerous ways) as a chain of elementary posets $[P_1,P_2,\dots,P_M]$ together with a sequence of positive integers $[k_1,\dots,k_{M-1}]$ that describes the interfaces: the last k_1 vertices of P_1 are identified with the first k_2 vertices of P_2, ..., the last k_{M-1} vertices of P_{M-1} are identified with the first k_{M-1} vertices of P_M. Of course we assume the compatibility conditions. We will denote the resulting poset by

$$G([P_1,P_2,\dots,P_M],[k_1,\dots,k_{M-1}]) \quad .$$

In the case of a ranked poset, it is especially transparent, since each of the component posets consists of two successive ranks.

In order to compute the generating function for enumerating P-partitions of $G([P_1, P_2, \dots, P_M], [k_1, \dots, k_{M-1}])$, we view it as iterated grafts, and iterate what we did before for the graft of two posets.

For large posets P, it would be hopeless to compute F_P directly. But as long as the "intersection-sizes" k_i's do not get too big, we can compute, with the Umbral method, the generating functions for enumerating very large posets. In other words, we can handle very tall posets, as long as they are skinny enough.

Enumerating Bounded P-Partitions

To get the generating function for P-partitions all of whose parts are $\leq M$, say, which now is a *polynomial* rather than a rational function, simply graft one last "claw" poset to the maximal vertices of P, and calling this new vertex L, look at the coefficient of x_L^M.

Enumerating Families of P-partitions

So far we outlined an *algorithm* for computing generating functions for *specific* posets. But, often we are interested in one-parameter or several-parameter families. For example, MacMahon's box theorem about plane partitions is about the P-partitions of the $m \times n$ rectangle with *symbolic* m and n. For every specific (not too large), n, `RotaStanley` can find F_P for rather large m, but what about *symbolic* m and n.

Now, MacMahon was extremely lucky that he got a 'nice' answer in terms of m **and** n. Getting something so explicit for a **two**-parameter family of posets is very unlikely to ever happen again, and testifies, that in some sense, MacMahon's result is trivial. By hindsight, it was extremely naive of him to expect that there would be a 'nice' answer for *solid partitions*, jumping to conclusions from the utterly trivial one-dimensional, and moderately trivial two-dimensional results.

However, with the Umbral method, one can still hope to get 'nice' (in a generalized sense) results for one-parameter families of posets, namely those that are iterated-grafting of an input poset P to itself.

So given a poset P, let's assume that the subposet induced by the last k vertices is isomorphic to the subposet induced by the first k vertices. Then define the n-th power $P^{(n)}$ to be

$$P^{(n)} := G([P, P, P, \dots, P], [k, k, \dots, k]) \quad ,$$

where there n P's and $n-1$ k's.

The computer can find the *self-Umbra* U from the 'input' first k vertices to the 'output' last k vertices. The catalytic variables are $x_1, \dots, x_k$, and one can finds an *Umbral* recurrence

$$F_n(q; x_1, \dots, x_k) = U[F_{n-1}(q; x_1, \dots, x_k)] \quad . \qquad (UmbralRecurrence)$$

This is a certain functional-recurrence equation, and if there is some conjectured expression in closed form or any other tractable description (for example, as a solution of linear recurrence equation with polynomial coefficients in (q^n, q)), then it can be easily proved (automatically!) by verifying this functional recurrence. Finally, to get the actual generating function, one plugs $x_1 = \ldots = x_k = 1$, getting $f_n(q) = F_n(q; 1, 1, \ldots, 1)$, that has the same "niceness status" as F_n, or, if in luck, even nicer. But even if we are not lucky, $(UmbralRecurrence)$, is an "answer" in the Wilfian sense, since it allows us to compute things in polynomial times, alas in $O(n^{k+1})$ rather than $O(n)$ steps, due to the k "catalytic" variables. [See procedure `UmbralRecurrenceSingle` in `RotaStanley`] .

The Maple Package RotaStanley

All of this is implemented in our Maple package `RotaStanley` downloadable from the webpage of this article

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

where one can also find some sample input and output. Readers who have Maple can generate lots of further output on their own.

A Challange: Interfacing with MacMahon's Omega Operator Method

If one wants to compute the generating function for a *specific*, random poset, with no discernible structure, then it is probably prefarable to use the MacMahon-Andrews-Paule-Riese Omega method.

MacMahon's Omega operator method was ingeniously resurrected and implemented by George Andrews, Peter Paule, and Axel Riese (See [APR1] and its sequels (e.g. [APR2])), and made orders-of-magnitude *faster* by Guoce Xin[X], who realized that continuous mathematics was just a red herring that slowed things down.

But it seems that our approach is superior for *families* that are "powers" as described above. It is true that in some simple cases, one can use the Omega method to do several special cases by computer, look at the pattern, and discover, by human "ingenuity" a functional equation that, if in luck, may be solved.

The advantage of our approach is that no humans are needed, and one can find the functional recurrence completely automatically.

Alas, once the "atomic" poset gets larger, our naive approach, that looks at all the compatible permutations for the atomic poset, also becomes intractable. In other words, if the atoms are small, we can handle very large molecules built from them, but if the atoms are big, we run out of time and memory.

But, the Omega method, that was so succesful in computing *generating functions* for enumerating P-partitions, should, almost as easily, be able to automatically compute the Umbral operators of the present approach. I am sure that *combining* the Umbral approach with the Omega method is likely to handle larger and larger posets and families, both "numerically" and symbolically. I leave this as a challenge to the readers.

REFERENCES

[APR1] G. Andrews, Peter Paule, Axel Riese, MacMahon's Partition Analysis III: The Omega Package, Europ. J. Comb. 22 (2001) 887-904.

[APR2] G. Andrews, P. Paule, A.Riese, MacMahon's Partition Analysis VIII: Plane Partition Diamonds, Adv. Appl. Math. 27 (2001) 231-242.

[K] D.E. Knuth, A note on solid partitions, Math. Comp. 24(1970), 955-962.

P.A. MacMahon, "Combinatory Analysis", 2 vols., Cambridge University Press, 1915 and 1916; reprinted in one volume by Chelsea, New York, 1960.

[RR] S. M. Roman and G.-C. Rota, The Umbral Calculus, Advances in Mathematics 27 (1978), 95-188.

[S1] R. Stanley, "Ordered Structures and partitions", Memoirs of the Amer. Math. Soc. no. 119 (1972).

[S2] R. Stanley, "Enumerative Combinatorics", vol. 1, Wadsworth, Monterey, 1986. Reprinted by Cambridge University Press.

[X] G. Xin, A fast algorithm for MacMahon's Partition Analysis, Elec. J. Combinatorics 11 (2004), no. 1, R58 (20pp).

[Z1] D. Zeilberger, The Umbral Transfer-Matrix Method: I. Foundations, J. Comb. Theory Ser. A 91, 451-463, (Rota memorial issue) (2000).

[Z2] D. Zeilberger, The Umbral Transfer-Matrix Method. II. Counting Plane Partitions, Personal J. of Ekhad and Zeilberger, http://www.math.rutgers.edu/~zeilberg/pj.html, (2002).

[Z3] D. Zeilberger, The Umbral Transfer-Matrix Method. III. Counting Animals, New York J of Mathematics 7(2001), 223-231.

[Z4] D. Zeilberger, The Umbral Transfer-Matrix Method. IV. Counting Self-Avoiding Polygons and Walks, Elec. J. Comb. 8(1)(2001), (22 pages) R28.

[Z5] D. Zeilberger, The Umbral Transfer-Matrix Method. V. The Goulden-Jackson Cluster Method for Infinitely Many Mistakes, INTEGERS, 2 (2002), (10 pages), A5 .