DETERMINANT: OLD ALGORITHMS, NEW INSIGHTS*

MEENA MAHAJAN† AND V VINAY‡

Abstract. In this paper we approach the problem of computing the characteristic polynomial of a matrix from the combinatorial viewpoint. We present several combinatorial characterizations of the coefficients of the characteristic polynomial, in terms of walks and closed walks of different kinds in the underlying graph. We develop algorithms based on these characterizations, and show that they tally with well-known algorithms arrived at independently from considerations in linear algebra.

Key words. Determinant, Algorithms, Combinatorics, Graphs, Matrices

AMS subject classifications. 05A15, 68R05, 05C50, 68Q25

1. Introduction. Computing the determinant, or the characteristic polynomial, of a matrix is a problem which has been studied several years ago from the numerical analysis viewpoint. In the mid 40's, a series of algorithms which employed sequential iterative methods to compute the polynomial were proposed, the most prominent being due to Samuelson [19], Krylov, Leverier; see, for instance, the presentation in [8]. Then, in the 80's, a series of parallel algorithms for the determinant were proposed, due to Csanky, Chistov, Berkowitz [6, 5, 1]. This culminated in the result, shown independently by several complexity theorists including Vinay, Damm, Toda, Valiant [26, 7, 24, 25], that computing the determinant of an integer matrix is complete for the complexity class GapL, and hence computationally equivalent in a precise complexity-theoretic sense to iterated matrix multiplication or matrix powering.

In an attempt to unravel the ideas that went into designing efficient parallel algorithms for the determinant, Valiant studied Samuelson's algorithm and interpreted the computation combinatorially [25]. He presented a combinatorial theorem concerning closed walks in graphs, the correctness of which followed from that of Samuelson's algorithm. This was the first attempt to view determinant computations as graph-theoretic rather than linear algebraic manipulations. Inspired by this, and by the purely combinatorial and extremely elegant proof of the Cayley-Hamilton Theorem due to Rutherford [18] (and independently discovered by Straubing [22]; see [2, 27] for nice expositions and see [3] for related material), Mahajan and Vinay [15] described a combinatorial algorithm for computing the characteristic polynomial. The proof of correctness of this algorithm is also purely combinatorial and does not rely on any linear algebra or polynomial arithmetic.

In this paper, we follow up on the work presented in [25, 22, 15] and present a unifying combinatorial framework in which to interpret and analyse a host of algorithms for computing the determinant and the characteristic polynomial. We first describe what the coefficients of the characteristic polynomial of a matrix M represent as combinatorial entities in the graph G_M whose adjacency matrix is M. We then

*A preliminary version of this paper appeared in the Proceedings of the Sixth Scandinavian Workshop on Algorithm Theory SWAT'98, Vol. **1432** in the Springer-Verlag LNCS series, pp 276–287.

†Institute of Mathematical Sciences, Chennai 600 113, India. (meena@imsc.ernet.in). Part of this work was done when this author was visiting the Department of Computer Science and Automation, IISc, Bangalore.

‡ Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012, India. (vinay@csa.iisc.ernet.in).

consider various algorithms for evaluating the coefficients, and in each case we relate the intermediate steps of the computation to manipulation of similar combinatorial entities, giving combinatorial proofs of correctness of these algorithms.

In particular, in the graph-theoretic setting, computing the determinant amounts to evaluating the signed weighted sum of cycle covers. This sum involves far too many terms to allow evaluation of each, and we show how the algorithms of [19, 5, 6] essentially expand this sum to include more terms i.e., generalizations of cycle covers, which eventually cancel out but which allow easy evaluation. The algorithm in [15] uses clow sequences explicitly; Samuelson's method [19] implicitly uses prefix clow sequences; Chistov's method [5] implicitly uses tables of tour sequences; and Csanky's algorithm [6] hinges around Leverier's Lemma (see, for instance, [8]), which can be interpreted using loops and partial cycle covers. In each of these cases, we explicitly demonstrate the underlying combinatorial structures, and give proofs of correctness which are entirely combinatorial in nature.

In a sense, this paper parallels the work done by a host of combinatorialists in proving the correctness of matrix identities using the graph-theoretic setting. Foata [9] used tours and cycle covers in graphs to prove the MacMohan master theorem; Rutherford and Straubing [18, 22] reproved the Cayley-Hamilton theorem using counting over walks and cycle covers; Garsia [11], Orlin [17] and Tempereley [23] independently found combinatorial proofs of the matrix-tree theorem and Chaiken [4] generalized the proof to the all-minor matrix-tree theorem; Foata [10] and then Zeilberger [27] gave new combinatorial proofs of the Jacobi identity; Gessel [12] used transitive tournaments in graphs to prove Vandermonde's determinant identity. More recently, Minoux [16] showed an extension of the matrix-tree theorem to semirings, again using counting arguments over arborescences in graphs. For beautiful surveys of some of these results, see Zeilberger's paper [27] and chapter 4 of Stanton and White's book on Constructive Combinatorics [20]. Zeilberger ends with a host of "exercises" in proving many more matrix identities combinatorially.

Thus, using combinatorial interpretations and arguments to prove matrix identities has been around for a while. To our knowledge, however, a similar application of combinatorial ideas to interpret, or prove correctness of, or even develop new *algorithms* computing matrix functions, has been attempted only twice before: by Valiant [25] in 1992, and by the present authors in our earlier paper in 1997 [15]. We build on our earlier work and pursue a new thread of ideas here.

This paper is thus a collection of new interpretations and proofs of known results. The paper is by and large self-contained.

2. Matrices, Determinants and Graphs. Let A be a square matrix of dimension n. For convenience, we state our results for matrices over integers, but they apply to matrices over any commutative ring.

We associate matrices of dimension n with complete directed graphs on n vertices, with weights on the edges. Let G_A denote the complete directed graph associated with the matrix A. If the vertices of G_A are numbered $\{1, 2, \ldots, n\}$, then the weight of the edge $\langle i, j\rangle$ is a_{ij}. We use the notation $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

The determinant of the matrix A, $\det(A)$, is defined as the signed sum of all weighted permutations of S_n as follows:

$$det(A) = \sum_{\sigma \in S_n} sgn(\sigma) \prod_i a_{i\sigma(i)}$$

where $sgn(\sigma)$ is the number (modulo 2) of inversions in σ i.e. the cardinality of the set $\{\langle i,j\rangle \mid i<j, \sigma(i)>\sigma(j)\}$ modulo 2.

Each $\sigma \in S_n$ has a cycle decomposition, and it corresponds to a set of cycles in G_A. For instance, with $n=5$, the permutation $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 2 & 3 & 5 & 1 \end{pmatrix}$ has a cycle decomposition (145)(2)(3) which corresponds to 3 cycles in G_A. Such cycles of G_A have an important property: they are all simple (non-intersecting), disjoint cycles; when put together, they touch each vertex exactly once. Such sets of cycles are called cycle covers. Note that cycle covers of G_A and permutations of S_n are in bijection with each other.

We define weights of cycle covers to correspond to weights of permutations. The weight of a cycle is the product of the weights of all edges in the cycle. The weight of a cycle cover is the product of the weights of all the cycles in it. Thus, viewing the cycle cover $\mathcal{C}$ as a set of edges, $w(\mathcal{C}) = \prod_{e\in\mathcal{C}} w(e)$. Since the weights of the edges are dictated by the matrix A, we can write $w(\mathcal{C}) = \prod_{\langle i,j\rangle\in\mathcal{C}} a_{ij}$.

We can also define the sign of a cycle cover consistent with the sign of the corresponding permutation. A cycle cover is even (resp. odd) if it contains an even number (resp. odd) of even length cycles. Equivalently, the cycle cover is even (resp. odd) if the number of cycles plus the number of edges is even (resp. odd). Define the sign of a cycle cover $\mathcal{C}$ to be $+1$ if $\mathcal{C}$ is even, and -1 if $\mathcal{C}$ is odd. Cauchy showed that with this definition, the sign of a permutation (based on inversions) and the sign of the associated cycle cover is the same. For our use, this definition of sign based on cycle covers will be more convenient.

Let $\mathcal{C}(G_A)$ denote the set of all cycle covers in the graph G_A. Then we have

$$det(A) = \sum_{\mathcal{C}\in\mathcal{C}(G_A)} sgn(\mathcal{C})w(\mathcal{C}) = \sum_{\mathcal{C}\in\mathcal{C}(G_A)} sgn(\mathcal{C}) \prod_{\langle i,j\rangle\in\mathcal{C}} a_{ij}$$

Consider the characteristic polynomial of A,

$$\chi_A(\lambda) = \det(\lambda I_n - A) = c_0\lambda^n + c_1\lambda^{n-1} + \ldots + c_{n-1}\lambda + c_n$$

To interpret these coefficients, consider the graph $G_A(\lambda)$ whose edges are labeled according to the matrix $\lambda I_n - A$. The coefficient c_l collects part of the contribution to $\det(\lambda I_n - A)$ from cycle covers having at least $(n-l)$ self loops. (A self loop at vertex k now carries weight $\lambda - a_{kk}$.) This is because a cycle cover with i self loops has weight which is a polynomial of degree i in λ. For instance, with $n=4$, consider the cycle cover $\langle 1,4\rangle, \langle 2,2\rangle, \langle 3,3\rangle, \langle 4,1\rangle$ in $G_A(\lambda)$. This has weight $(-a_{14})(\lambda - a_{22})(\lambda - a_{33})(-a_{41})$, contributing $a_{14}a_{22}a_{33}a_{41}$ to c_4, $-a_{14}a_{41}(a_{22}+a_{33})$ to c_3, $a_{14}a_{41}$ to c_2, and 0 to c_1.

Following notation from [22], we consider partial permutations, corresponding to partial cycle covers. A partial permutation σ is a permutation on a subset $S \subseteq [n]$. The set S is called the domain of σ, denoted $\mathrm{dom}(\sigma)$. The completion of σ, denoted $\hat{\sigma}$, is the permutation in S_n obtained by letting all elements outside $\mathrm{dom}(\sigma)$ be fixed points. This permutation $\hat{\sigma}$ corresponds to a cycle cover $\mathcal{C}$ in G_A, and σ corresponds to a subset of the cycles in $\mathcal{C}$. We call such a subset a partial cycle cover $\mathcal{PC}$, and we call $\mathcal{C}$ the completion of $\mathcal{PC}$. A partial cycle cover is defined to have the same parity and sign as its completion. It is easy to see that the completion need not be explicitly accounted for in the parity; a partial cycle cover $\mathcal{PC}$ is even (resp. odd) iff the number of cycles in it, plus the number of edges in it, is even (resp. odd).

Getting back to the characteristic polynomial, observe that to collect the contributions to c_l, we must look at all partial cycle covers with l edges. The $n-l$ vertices left uncovered by such a partial cycle cover $\mathcal{PC}$ are the self-loops, from whose weight the λ term has been picked up. Of the l vertices covered, self-loops, if any, contribute the $-a_{kk}$ term from their weight, not the λ term. And other edges, say $\langle i, j\rangle$ for $i \neq j$, contribute weights $-a_{ij}$. Thus the weights for $\mathcal{PC}$ evidently come from the graph G_{-A}. If we interpret weights over the graph G_A, a factor of $(-1)^l$ must be accounted for independently.

Formally,

DEFINITION 2.1.

A cycle is an ordered sequence of m edges $C = \langle e_1, e_2, \ldots, e_m\rangle$ where $e_i = \langle u_i, u_{i+1}\rangle$ for $i \in [m-1]$ and $e_m = \langle u_m, u_1\rangle$ and $u_1 \leq u_i$ for $i \in [m]$ and all the u_i's are distinct. u_1 is called the head of the cycle, denoted $h(C)$. The length of the cycle is $|C| = m$, and the weight of the cycle is $w(C) = \prod_{i=1}^{m} w(e_i)$. The vertex set of the cycle is $V(C) = \{u_1, \ldots, u_m\}$.

An l-cycle cover $\mathcal{C}$ is an ordered sequence of cycles $\mathcal{C} = \langle C_1, \ldots, C_k\rangle$ such that $V(C_i) \cap V(C_j) = \phi$ for $i \neq j$, $h(C_1) < \ldots < h(C_k)$ and $|C_1| + \ldots + |C_k| = l$.

The weight of the l-cycle cover is $wt(\mathcal{C}) = \prod_{j=1}^{k} w(C_j)$, and the sign is $sgn(\mathcal{C}) = (-1)^{l+k}$.

As a matter of convention, we call n-cycle covers simply cycle covers.

PROPOSITION 2.2. *The coefficients of $\chi_A(\lambda)$ are given by*

$$c_l = (-1)^l \sum_{\mathcal{C} \text{ is an } l\text{-cycle cover in } G_A} sgn(\mathcal{C}) wt(\mathcal{C})$$

3. Summing over permutations efficiently. As noted in Proposition 2.2, evaluating the determinant (or for that matter, any coefficient of the characteristic polynomial) amounts to evaluating the signed weighted sum over cycle covers (partial cycle covers of appropriate length). We consider four efficient algorithms for computing this sum. Each expands this sum to include more terms which mutually cancel out. The differences between the algorithms is essentially in the extent to which the sum is expanded.

3.1. From Cycle Covers to Clow Sequences. Genaralize the notion of a cycle and a cycle cover as follows:

A *clow* (for closed-walk) is a cycle in G_A (not necessarily simple) with the property that the minimum vertex in the cycle – called the *head* – is visited only once. An l-clow sequence is a sequence of clows where the heads of the clows are in strictly increasing order and the total number of edges (counting each edge as many times as it is used) is l.

Formally,

DEFINITION 3.1. *A clow is an ordered sequence of edges $C = \langle e_1, e_2, \ldots, e_m\rangle$ such that $e_i = \langle u_i, u_{i+1}\rangle$ for $i \in [m-1]$ and $e_m = \langle u_m, u_1\rangle$ and $u_1 \neq u_j$ for $j \in \{2, \ldots, m\}$ and $u_1 = min\{u_1, \ldots, u_m\}$. The vertex u_1 is called the head of the clow and denoted $h(C)$. The length of the clow is $|C| = m$, and the weight of the clow is $w(C) = \prod_{i=1}^{m} w(e_i)$.*

An l-clow sequence $\mathcal{C}$ is an ordered sequence of clows $\mathcal{C} = \langle C_1, \ldots, C_k\rangle$ such that $h(C_1) < \ldots < h(C_k)$ and $|C_1| + \ldots + |C_k| = l$.

The weight of the l-clow sequence $\mathcal{C}$ is $wt(\mathcal{C}) = \prod_{j=1}^{k} w(C_j)$, and the sign of $\mathcal{C}$ is $sgn(\mathcal{C}) = (-1)^{l+k}$.

Note that the set of l-clow sequences properly includes the set of l-cycle covers on a graph. And the sign and weight of a cycle cover are consistent with its sign and weight when viewed as a clow sequence.

THEOREM 3.2 (Theorem 1 in [15]).

$$c_l = (-1)^l \sum_{\mathcal{C} \text{ is an } l\text{-clow sequence}} sgn(\mathcal{C}) wt(\mathcal{C})$$

Proof. We construct an involution φ on the set of l-clow sequences. The involution has the property that φ^2 is the identity, φ maps an l-cycle cover to itself, and otherwise $\mathcal{C}$ and $\varphi(\mathcal{C})$ have the same weight but opposing sign. This shows that the contribution of l-clow sequences that are *not* l-cycle covers is zero. Consequently, only l-cycle covers contribute to the summation, yielding exactly c_l.

Let $\mathcal{C} = \langle C_1, \ldots, C_k \rangle$ be an l-clow sequence. Choose the smallest i such that C_{i+1} to C_k is a p-cycle cover for some p. If $i = 0$, the involution maps $\mathcal{C}$ to itself. Otherwise, having chosen i, traverse C_i starting from $h(C_i)$ until one of two things happen.

1. We hit a vertex that touches one of C_{i+1} to C_k.
2. We hit a vertex that completes a cycle within C_i.

Let us call the vertex v. Given the way we chose i, such a v must exist. Vertex v cannot satisfy both of the above conditions.

Case 1: . Suppose v touches C_j. Map $\mathcal{C}$ to a clow sequence

$$\mathcal{C}' = \langle C_1, \ldots, C_{i-1}, C_i', C_{i+1}, \ldots, C_{j-1}, C_{j+1}, \ldots C_k \rangle$$

The modified clow, C_i' is obtained from C_i by inserting the cycle C_j into it at the first occurence of v.

Case 2: . Suppose v completes a simple cycle C in C_i. Cycle C must be disjoint from all the later cycles. We now modify the sequence $\mathcal{C}$ by deleting C from C_i and introducing C as a new clow in an appropriate position, depending on the minimum labeled vertex in C, which we make the head of C.

Figure 3.1 illustrates the mapping.

In both of the above cases, the new sequence constructed maps back to the original sequence in the opposite case. Furthermore, the number of clows in the two sequences differ by one, and hence the signs are opposing, whereas the weight is unchanged. This is the desired involution. ∎

Furthermore, the above mapping does not change the head of the first clow in the sequence. So if the goal is to compute the determinant which sums up the n-cycle covers, then the head of the first cycle must be the vertex 1. So it suffices to consider clow sequences where the first clow has head 1.

Algorithm using clow sequences. Both sequential and parallel algorithms based on the clow sequences characterization are described in [15]. We briefly describe the implementation idea below, for the case c_n.

The goal is to sum up the contribution of all clow sequences. The clow sequences can be partitioned into n groups based on the number of clows. Let C_k be the sum of the weights of all clow sequences with exactly k clows, and let $D_k = (-1)^{n+k} C_k$. Then $c_n = \sum_{k=1}^{n} D_k$.

To compute C_k, we use a divide-and-conquer approach on the number of clows: any clow sequence contributing to C_k can be suitably split into two partial clow sequences, with the left sequence having $\lceil k/2 \rceil$ clows. The heads of all clows in the left part must be less than the head of the first clow in the rightmost part. And the

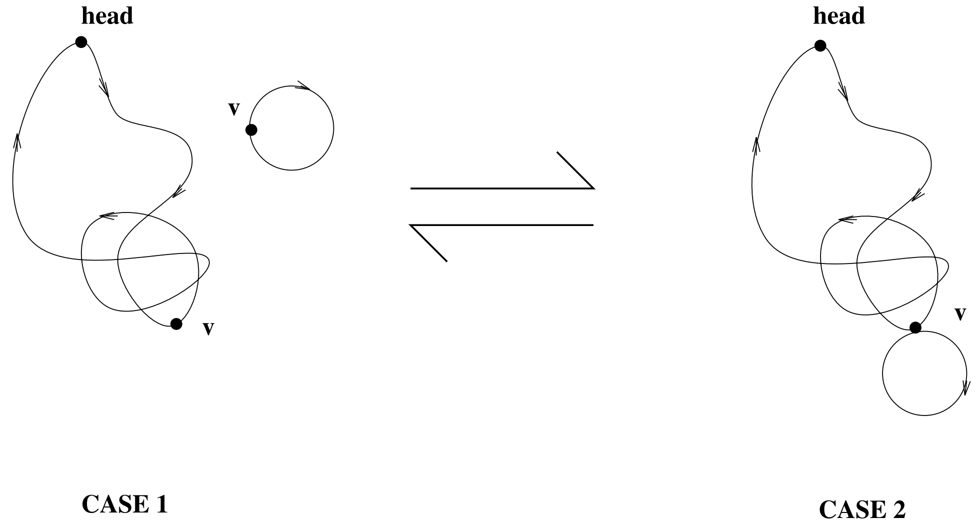

FIG. 3.1. *Pairing clow sequences of opposing signs*

lengths of the left and the right partial clow sequences must add up to n. Let variable $g[p, l, u, v]$ sum up the weights of all partial clow sequences with p clows, l edges, head of first clow u, and heads of all clows at most v. (We need not consider variables where $l < p$ or $u > v$.) Then $C_k = g[k, n, 1, n]$, and such variables can be evaluated by the formula

$$g[p, l, u, v] = \begin{cases} \displaystyle\sum_{\substack{q \leq r \leq q + (l-p) \\ u < w \leq v}} g[q, r, u, w-1] \cdot g[p-q, l-r, w, v] & \text{if } p > 1 \\ g[l, u] & \text{if } p = 1 \end{cases}$$

where $q = \lceil p/2 \rceil$. The variable $g[l, u]$ sums up the weights of all clows of length l with head u, and is also evaluated in a divide-and-conquer fashion. A clow with head u is either a self-loop if $l = 1$, or it must first visit some vertex $v > u$, find a path of length $l - 2$ to some vertex $w > u$ through vertices all greater than u, and then return to u. So

$$g[l, u] = \begin{cases} a_{uu} & \text{if } l = 1 \\ \sum_{v, w > u} a_{uv} \cdot a_{wu} & \text{if } l = 2 \\ \sum_{v, w > u} a_{uv} \cdot c[l-2, u, v, w] \cdot a_{wu} & \text{otherwise} \end{cases}$$

The variable $c[l, u, v, w]$ sums the weights of all length l paths from v to w going through vertices greater than u. These variables can be evaluated as follows:

$$\begin{aligned} c[1, u, v, w] &= a_{vw} \\ c[l, u, v, w] &= \textstyle\sum_{x > u} c[p, u, v, x] \cdot c[l-p, u, x, w] \qquad \text{if } l > 1, \text{ where } p = \lceil l/2 \rceil \end{aligned}$$

3.2. Clow Sequences with the Prefix Property: Getting to Samuelson's Method. The generalization from cycle covers to clow sequences has a certain extravagance. The reason for going to clow sequences is that evaluating their weighted sum is easy, and this sum equals the sum over cycle covers. However, there are several clow sequences which we can drop from consideration without sacrificing ease of computation. One such set arises from the following consideration:

In a cycle cover, all vertices are covered exactly once. Suppose we enumerate the vertices in the order in which they are visited in the cycle cover (following the order imposed by the cycle heads). If vertex h becomes the head of a cycle, then all vertices in this and subsequent cycles are larger than h. So all the lower numbered vertices must have been already visited. So at least $h-1$ vertices, and hence $h-1$ edges, must have been covered.

We can require our clow sequences also to satisfy this property. We formalize the prefix property: a clow sequence $\mathcal{C} = \langle C_1, \ldots, C_k \rangle$ has the prefix property if for $1 \leq r \leq k$, the total lengths of the clows $C_1, \ldots, C_{r-1}$ is at least $h(C_r) - 1$. A similar prefix property can be formalized for partial cycle covers. Formally,

DEFINITION 3.3. *An l-clow sequence $\mathcal{C} = \langle C_1, \ldots, C_k \rangle$ is said to have the prefix property if it satisfies the following condition:*

$$\forall r \in [k], \quad \sum_{t=1}^{r-1} |C_t| \geq h(C_r) - 1 - (n - l)$$

The interesting fact is that the involution constructed in the previous sub-section for clow sequences works even over this restricted set!

THEOREM 3.4 (Theorem 2 in [25]).

$$c_l = (-1)^l \sum_{\substack{\mathcal{C} \text{ is an } l\text{-clow sequence} \\ \text{with the prefix property}}} sgn(\mathcal{C}) wt(\mathcal{C})$$

A new proof of the above theorem.
In [25], Valiant observes that prefix clow sequences are the terms computed by Samuelson's method for evaluating $\chi_\lambda(A)$. Hence the correctness of the theorem follows from the correctness of Samuelson's method. And the correctness of Samuelson's method is traditionally shown using linear algebra.

Here is a simple alternative combinatorial proof of this theorem. Observe that the involution defined in the proof of Theorem 3.2 maps clow sequences with prefix property to clow sequences with prefix property. Why? Let $\mathcal{C}$ be an l-clow sequence with the prefix property, satisfying case 1 in the proof. Since the length of clow C_i only increases in the process, the prefix property continues to hold. Now let $\mathcal{C}$ be an l-clow sequence with the prefix property, satisfying case 2. The involution constructs a new l-clow sequence $\mathcal{C}'$ by detaching cycle C from clow C_i and inserting it later in the sequence, say between C_{j-1} and C_j. This does not change $h(C_i)$. Let $\mathcal{C}' = \mathcal{D} = \langle D_1, \ldots, D_{k+1} \rangle$; here $D_t = C_t$ for $t \in [i-1]$ or for $t = i+1$ to $j-1$, $D_i = C_i \setminus C$, $D_j = C$ and $D_{t+1} = C_t$ for $t = j$ to k. We must show that $\mathcal{D}$ has the prefix property. For $r \in [i]$, and for $r = j+1$ to $k+1$, the condition $\sum_{t=1}^{r-1} |D_t| \geq h(D_r) - 1 - (n-l)$ holds because $\mathcal{C}$ has the prefix property. Now let $i+1 \leq r \leq j$. Since C_i was chosen from $\mathcal{C}$ for modification, and since $i+1 \leq r$, we know that $D_r, \ldots, D_{k+1}$ form a partial cycle cover i.e. they form simple disjoint cycles. And the heads of these cycles are arranged in increasing order. So the vertices covered in $D_r, \ldots, D_{k+1}$ must all be at least as large as $h(D_r)$ and all distinct. But there are only $n - h(D_r) + 1$ such vertices. Hence

$$\begin{array}{rcl} \sum_{t=r}^{k+1} |D_t| & \leq & n - h(D_r) + 1 \\ l - \sum_{t=1}^{r-1} |D_t| & \leq & n - h(D_r) + 1 \\ \sum_{t=1}^{r-1} |D_t| & \geq & h(D_r) - 1 - (n - l) \end{array}$$

and $\mathcal{D}$ satisfies the prefix property.

Thus in summing over all l-clow sequences with the prefix property, the only l-clow sequences that do not cancel out are the l-cycle covers, giving the claimed result. $\square$

Algorithm using prefix clow sequences. To compute c_l using this characterization, we must sum up the contribution of all l-clow sequences with the prefix property. One way is to modify the dynamic programming approach used in the previous sub-section for clow sequences. This can be easily done. Let us instead do things differently; the reason will become clear later.

Adopt the convention that there can be clows of length 0. Then each l-clow sequence $\mathcal{C}$ has exactly one clow C_i with head i, for $i = 1$ to n. So we write $\mathcal{C} = \langle C_1, \ldots, C_n \rangle$.

Define the signed weight of a clow C as $sw(C) = -w(C)$ if C has non-zero length, and $sw(C) = 1$ otherwise. And define the signed weight of an l-clow sequence as $sw(\mathcal{C}) = \prod_{i=1}^{n} sw(C)$. Then $sgn(\mathcal{C})w(\mathcal{C}) = (-1)^l sw(\mathcal{C})$. So from the preceding theorem,

$$c_l = \sum_{\substack{\mathcal{C} \text{ is an } l\text{-clow sequence} \\ \text{with the prefix property}}} sw(\mathcal{C})$$

We say that a sequence of non-negative integers $l_1, \ldots, l_n$ satisfies the property prefix(l) if

1. $\sum_{t=1}^{n} l_t = l$, and
2. For $r \in [n]$, $\sum_{t=1}^{r-1} l_t \geq r - 1 - (n - l)$. Alternatively $\sum_{t=r}^{n} l_t \leq n - r + 1$.

Such sequences are "allowed" as lengths of clows in the clow sequences we construct; no other sequences are allowed.

We group the clow sequences with prefix property based on the lengths of the individual clows. In a clow sequence with prefix property $\mathcal{C}$, if the length of clow C_i (the possibly empty clow with head i) is l_i, then any clow with head i and length l_i can replace C_i in $\mathcal{C}$ and still give a clow sequence satisfying the prefix property. Thus, if $z(i,p)$ denotes the total signed weight of all clows that have vertex i as head and length p, then

$$c_l = \sum_{l_1, \ldots, l_n : \text{prefix}(l)} \prod_{i=1}^{n} z(i, l_i)$$

To compute c_l efficiently, we place the values $z(i,p)$ appropriately in a series of matrices $B_1, \ldots, B_n$. The matrix B_k has entries $z(k,p)$. Since we only consider sequences satisfying prefix(l), it suffices to consider $z(k,p)$ for $p \leq n - k + 1$. Matrix B_k is of dimension $(n-k+2) \times (n-k+1)$ and has $z(k,p)$ on the pth lower diagonal as shown below.

$$B_k = \begin{bmatrix} z(k,0) & 0 & 0 & \cdots & 0 & 0 \\ z(k,1) & z(k,0) & 0 & \cdots & 0 & 0 \\ z(k,2) & z(k,1) & z(k,0) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & 0 \\ z(k,n-k) & z(k,n-k-1) & z(k,n-k-2) & \cdots & z(k,1) & z(k,0) \\ z(k,n-k+1) & z(k,n-k) & z(k,n-k-1) & \cdots & z(k,2) & z(k,1) \end{bmatrix}$$

Now from the equation for c_l, it is clear that

$$c_l = \sum_{\substack{l+1=j_0 \geq j_1 \geq j_2 \geq \ldots \geq j_n = 1 : \\ j_0 - j_1, j_1 - j_2, \ldots, j_{n-1} - j_n : \text{prefix}(l)}} \left(\prod_{i=1}^{n} B_i[j_{i-1}, j_i] = \right) \left(\prod_{i=1}^{n} B_i \right) [l+1, 1]$$

or, more succintly,

$$[\ c_0\ c_1\ c_2\ c_3\ \cdots\ \cdots\ c_n\ ]^T = \prod_{k=1}^{n} B_k$$

It remains now to compute $z(i,p)$, the entries in the B matrices. We know that $z(i,0) = 1$ and $z(i,1) = -a_{ii}$. For $p \geq 2$, a clow of length p with head i must first visit a vertex $u > i$, then perform a walk of length $p-2$ via vertices greater than i to some vertex $v > i$, and then return to i. To construct the path, we exploit the fact that the (j,k)th entry in a matrix A^p gives the sum of the weights of all paths in G_A of length exactly p from j to k. So we must consider the induced subgraph with vertices $i+1, \ldots, n$. This has an adjacency matrix A_{i+1} obtained by removing the first i rows and the first i columns of A. So $A_1 = A$. Consider the submatrices of A_i as shown below.

$$A_i = \begin{pmatrix} a_{ii} & \begin{pmatrix} \quad R_i \quad \end{pmatrix} \\ \begin{pmatrix} \\ S_i \\ \\ \end{pmatrix} & \begin{pmatrix} \\ \quad A_{i+1} \quad \\ \\ \end{pmatrix} \end{pmatrix}$$

Then the clows contributing to $z(i,p)$ must use an edge in R_i, perform a walk corresponding to A_{i+1}^{p-2}, and then return to i via an edge in S_i. In other words,

$$z(i,p) = -R_i\ A_{i+1}^{p-2}\ S_i$$

So the matrices B_k look like this:

$$B_k = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -a_{kk} & 1 & 0 & \cdots & 0 & 0 \\ -R_kS_k & -a_{kk} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & 0 \\ -R_kA_{k+1}^{n-k-2}S_k & -R_kA_{k+1}^{n-k-3}S_k & -R_kA_{k+1}^{n-k-4}S_k & \cdots & -a_{kk} & 1 \\ -R_kA_{k+1}^{n-k-1}S_k & -R_kA_{k+1}^{n-k-2}S_k & -R_kA_{k+1}^{n-k-3}S_1 & \cdots & -R_kS_k & -a_{kk} \end{bmatrix}$$

This method of computing $\chi_A(\lambda)$ is precisely Samuelson's method [19, 8, 1, 25]. Samuelson arrived at this formulation using Laplace's Theorem on the matrix $\lambda I - A$, whereas we have arrived at it via clow sequences with the prefix property. This interpretation of the Samuelson-Berkowitz algorithm is due to Valiant[25]; the combinatorial proof of correctness (proof of Theorem 3.4) is new. (It is mentioned, without details, in [15].)

3.3. From Clows to Tour Sequences Tables: Getting to Chistov's Algorithm. We now move in the other direction – generalize further beyond clow sequences. Firstly, we relax the condition that the head of a clow may be visited only once. This gives us more generalized closed walks which we call tours. To fix a canonical representation, we do require the edges of the tour to be listed beginning from an occurrence of the head. Since there could be multiple such occurrences, we get different tours with the same multiset of edges. For instance, the tour corresponding to the vertex sequence `253246` is different from the tour corresponding to the vertex sequence `246253`. Secondly, we deal with not just sequences but ordered lists, or tables, of sequences. Within a sequence, the tours are ordered by their heads (and all heads are distinct). However, there is no restriction on how the sequences must be ordered in the table. In fact, for the same multi-set of sequences, different orderings of the sequences will give different tables that we treat as distinct. Thirdly, the parity of a tour sequence table depends on the number of sequences in it, not the number of tours in it. A clow sequence is thus a tour sequence table where (i) each sequence contains a single tour which is a clow, and (ii) within the table the sequences are ordered by their tour heads. Formally,

DEFINITION 3.5. *A tour is an ordered sequence of edges $C = \langle e_1, e_2, \ldots, e_p \rangle$ such that $e_i = \langle u_i, u_{i+1} \rangle$ for $i \in [p-1]$ and $e_p = \langle u_p, u_1 \rangle$ and $u_i = min\{u_1, \ldots, u_m\}$. The vertex u_1 is called the head of the tour and denoted $h(C)$. The length of the tour is $|T| = p$, and the weight of the tour is $wt(T) = \prod_{i=1}^{m} w(e_i)$.*

A j-tour sequence $\mathcal{T}$ is an ordered sequence of tours $\mathcal{T} = \langle T_1, \ldots, T_k \rangle$ such that $h(T_1) < \ldots < h(T_k)$ and $|T_1| + \ldots + |T_k| = j$. The weight of the tour sequence is $wt(\mathcal{T}) = \prod_{j=1}^{k} wt(T_j)$, and the length is $|\mathcal{T}| = j$.

An l-tour sequence table TST is an ordered sequence of tour sequences $\mathcal{F} = \langle \mathcal{T}_1, \ldots, \mathcal{T}_r \rangle$ such that $|\mathcal{T}_1| + \cdots + |\mathcal{T}_r| = l$. The weight of the TST is $wt(\mathcal{F}) = \prod_{j=1}^{r} wt(\mathcal{T}_j)$, and the sign is $(-1)^{l+r}$.

The following theorem shows that even TSTs can be used to compute the characteristic polynomial.

THEOREM 3.6.

$$c_l = (-1)^l \sum_{\mathcal{F} \text{ is an } l\text{-}TST} sgn(\mathcal{F}) wt(\mathcal{F})$$

Proof. We present an involution on the set of l-TSTs with all l-clow sequences being fixed points, and all other l-TSTs being mapped to TSTs of the same weight but opposing sign. Since l-clow sequences which are not cycle covers also yield a net contribution of zero (Theorem 3.2), the sum over all l-TSTs is precisely c_l.

Given an l-TST $\mathcal{F} = \langle \mathcal{T}_1, \ldots, \mathcal{T}_r \rangle$, let H be the set of all vertices which occur as heads of some tour in the table. For $S \subseteq H$, we say that S has the *clow sequence property* if the following holds: There is an $i \leq r$ such that

1. The tour sequences $\mathcal{T}_{i+1}, \ldots, \mathcal{T}_r$ are all single-tour sequences (say tour sequence $\mathcal{T}_j$ is the tour T_j),
2. No tour in any of the tour sequences $\mathcal{T}_1, \ldots, \mathcal{T}_i$ has a head vertex in S,
3. Each vertex in S is the head of a tour T_j for some $i+1 \leq j \leq r$. i.e. $\{h(T_j) \mid j = i+1, \ldots, r\} = S$
4. The tour sequence table $\langle \mathcal{T}_{i+1}, \ldots, \mathcal{T}_r \rangle$ actually forms a clow sequence. i.e. the tours T_j for $i+1 \leq j \leq r$ are clows, and $h(\mathcal{T}_{i+1}) < \ldots < h(\mathcal{T}_r)$.

In other words, all tours in $\mathcal{F}$ whose heads are in S are actually clows which occur in a contiguous block of single-tour sequences, arranged in strictly increasing order of heads, and this block is not followed by any other tour sequences in $\mathcal{F}$.

Note that the empty set vacuously has the clow sequence property.

Example: In the TST $\langle\ \langle 1,2,5 \rangle, \langle 3 \rangle, \langle 4 \rangle, \langle 6 \rangle\ \rangle$, where only tour heads have been represented and where all tours are clows, $\{3,4,6\}$ has this property but $\{3,4\}$, $\{3,6\}$, $\{5,6\}$ do not.

Now, in H, find the smallest vertex v such that $H_{>v} = \{h \in H \mid h > v\}$ has the clow sequence property but $H_{\geq v} = \{h \in H \mid h \geq v\}$ does not.

If no such v exists, then H satisfies the clow sequence property and hence $\mathcal{F}$ is an l-clow sequence. In this case, map it to itself.

If such a v exists, then locate the first tour sequence $\mathcal{T}_i = \langle T_1, \ldots, T_k \rangle$ where v appears (as a head). Then v is the head of the last tour T_k, because all tours with larger heads occur in a contiguous block of single-tour sequences at the end. The tour T_k can be uniquely decomposed as TC, where T is a tour and C a clow, both with head v.

Case 1. $T \neq \phi$.
Map this l-TST to an l-TST where $\mathcal{T}_i$ is replaced, at the same position, by the following two tour sequences: $\langle C \rangle, \langle T_1, \ldots, T_{k-1}, T \rangle$. This preserves weight but inverts the sign. In the modified l-TST, the newly introduced sequence containing only C will be chosen for modification as in Case 3.

Case 2. $T = \phi$, and $k > 1$.
Map this l-TST to an l-TST where $\mathcal{T}_i$ is replaced, at the same position, by the following two tour sequences: $\langle C \rangle, \langle T_1, \ldots, T_{k-1} \rangle$. This too preserves weight but inverts the sign. In the modified l-TST, the newly introduced sequence containing only C will be chosen for modification as in Case 3.

Case 3. $T = \phi$ and $k = 1$.
Then a tour sequence $\mathcal{T}_{i+1}$ must exist, since otherwise $H_{\geq v}$ would satisfy the clow sequence property. Now, if $\mathcal{T}_{i+1}$ has a tour with head greater than v, then, since $H_{>v}$ satisfies the clow sequence property, the TST $\mathcal{T}_{i+1}, \ldots, \mathcal{T}_r$ must be a clow sequence. But recall that T has the first occurrence of v as a head and is itself a clow, so then

$\mathcal{T}_i, \dots, \mathcal{T}_r$ must also be a clow sequence, and $H_{\geq v}$ also satisfies the clow sequence property, contradicting our choice of v. So $\mathcal{T}_{i+1}$ must have all tours with heads at most v. Let $\mathcal{T}_{i+1} = \langle P_1, \dots, P_s \rangle$. Now there are two sub-cases depending on the head of the last tour P_s.

Case 3(a). $h(P_s) = v$. Form the tour $P'_s = P_s C$. Map this l-TST to a new l-TST where the tour sequences $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$ are replaced, at the same position, by a single tour sequence $\langle P_1, \dots, P_{s-1}, P'_s \rangle$. The weight is preserved and the sign inverted, and in the modified l-TST, the tour P'_s in this new tour sequence will be chosen for modification as in Case 1.

Case 3(b). $h(P_s) \neq v$. Map this l-TST to a new l-TST where the tour sequences $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$ are replaced, at the same position, by a single tour sequence $\langle P_1, \dots, P_s, C \rangle$. The weight is preserved and the sign inverted, and in the modified l-TST, the tour C in this new tour sequence will be chosen for modification as in Case 2.

Thus l-TSTs which are not l-clow sequences yield a net contribution of zero. □

(The involution may be simpler to follow if we modify the notation as follows: decompose each tour uniquely into one or more clows with the same head, and represent these clows in the same order in which they occur in the tour. Now a TST is a table of sequences of clows where, within a sequence, clows are ordered in *non-decreasing* order of head. It is easy to see that we are still talking of the same set of objects, but only representing them differently. Now, the involution picks the vertex v as above, picks the first tour sequence where v occurs as a head, picks the last clow in this sequence, and either moves this clow to a new sequence if it is not alone in its sequence, as in cases 1 and 2, or appends it to the following sequence, as in case 3.

Example: For a TST $\mathcal{F}$ represented using clows, let the clow heads be as shown below

$$\langle\ \langle 1,2,2,5,5\rangle, \langle 3\rangle, \langle 4\rangle, \langle 6\rangle\ \rangle \qquad .$$

Vertex 5 is chosen as v, the first tour sequence is chosen, and as dictated by Case 1, this TST is mapped to a new TST $\mathcal{F}'$ with the tours rearranged as shown below

$$\langle\ \langle 5\rangle\langle 1,2,2,5\rangle, \langle 3\rangle, \langle 4\rangle, \langle 6\rangle\ \rangle$$

In $\mathcal{F}'$ again vertex 5 is chosen, and the first tour sequence is merged with the second as dictated by Case 3(a), to get back $\mathcal{F}$.

If the first tour sequence of $\mathcal{F}$ were $\langle 1,2,2,5\rangle$ instead, then by Case 2, $\mathcal{F}$ would be mapped to

$$\langle\ \langle 5\rangle\langle 1,2,2\rangle, \langle 3\rangle, \langle 4\rangle, \langle 6\rangle\ \rangle$$

from which $\mathcal{F}$ would be recovered by Case 3(b).)

Algorithm using tour sequence tables. We show how grouping the l-TSTs in a carefully chosen fashion gives a formulation which is easy to compute.

Define $e_l = (-1)^l c_l$, then

$$e_l = \sum_{\mathcal{F} \text{ is an } l\text{-TST}} sgn(\mathcal{F}) wt(\mathcal{F})$$

To compute c_l and hence e_l using this characterization, we need to compute the contributions of all l-TSTs. This is more easily achieved if we partition these contributions into l groups depending on how many edges are used up in the first tour sequence of the table. Group j contains l-TSTs of the form $\mathcal{F} = \langle \mathcal{T}_1, \dots, \mathcal{T}_r \rangle$ where $|\mathcal{T}_1| = j$. Then $\mathcal{F}' = \langle \mathcal{T}_2, \dots, \mathcal{T}_r \rangle$ forms an $(l-j)$-TST, and $sgn(\mathcal{F}) = -sgn(\mathcal{F}')$ and $wt(\mathcal{F}) = wt(\mathcal{T}_1) wt(\mathcal{F}')$. So the net contribution to e_l from this group, say $e_l(j)$, can be factorized as

$$\begin{aligned} e_l(j) &= \sum_{\substack{\mathcal{T}:\ j\text{-tour sequence} \\ \mathcal{F}':\ (l-j)\text{-TST}}} -sgn(\mathcal{F}')wt(\mathcal{F}')wt(\mathcal{T}) \\ &= -\left(\sum_{\mathcal{T}:\ j\text{-tour sequence}} wt(\mathcal{T})\right)\left(\sum_{\mathcal{F}':\ (l-j)\text{-TST}} sgn(\mathcal{F}')wt(\mathcal{F}')\right) \\ &= -d_j e_{l-j} \end{aligned}$$

where d_j is the sum of the weights of all j-tour sequences.

Now we need to compute d_j.

It is easy to see that $A^l[1,1]$ gives the sum of the weights of all tours of length l with head 1. To find a similar sum over tours with head k, we must consider the induced subgraph with vertices $k, k+1, \ldots, n$. This has an adjacency matrix A_k obtained by removing the first $k-1$ rows and the first $k-1$ columns of A. (We have already exploited these properties in Section 3.2.) Let $y(l,k)$ denote the sum of the weights of all l-tours with head k. Then $y(l,k) = A_k^l[1,1]$.

The weight of a j-tour sequence $\mathcal{T}$ can be split into n factors: the kth factor is 1 if $\mathcal{T}$ has no tour with head k, and is the weight of this (unique) tour otherwise. So

$$\begin{aligned} d_j &= \sum_{0\le l_i \le j:\ l_1+\ldots+l_n=j} \prod_{i=1}^n y(l_i, i) \\ &= \sum_{0\le l_i \le j:\ l_1+\ldots+l_n=j} \prod_{i=1}^n A_i^{l_i}[1,1] \end{aligned}$$

Let us define a power series $D(x) = \sum_{j=0}^{\infty} d_j x^j$. Then, using the above expression for d_j, we can write

$$D(x) = \left(\sum_{l=0}^{\infty} x^l A_1^l[1,1]\right)\left(\sum_{l=0}^{\infty} x^l A_2^l[1,1]\right)\ldots\left(\sum_{l=0}^{\infty} x^l A_n^l[1,1]\right)$$

Since we are interested in d_j only for $j \le n$, we can ignore monomials of degree greater than n. This allows us to evaluate the first $n+1$ coefficients of $D(x)$ using matrix powering and polynomial arithmetic. And now e_l can be computed inductively using the following expression:

$$e_l = \sum_{j=1}^{l} e_l(j) = \sum_{j=1}^{l} -d_j e_{l-j}$$

But this closely matches Chistov's algorithm [5]! The only difference is that Chistov started off with various algebraic entities, manipulated them using polynomial arithmetic, and derived the above formulation, whereas we started off with TSTs which are combinatorial entities, grouped them suitably, and arrived at the same formulation. And at the end, Chistov uses polynomial arithmetic to combine the computation of $D(x)$ and e_l. For completeness, we sketch below how Chistov arrived at this formulation.

Chistov's algorithm adopts the following technique (See for example [13]): let C_i be the sub-matrix obtained by deleting the first $n-i$ rows and first $n-i$ columns of A. (In our earlier notation, C_i is the matrix A_{n-i+1}. We use C_i here to keep subscripts shorter.) Let $\Delta_i(x)$ be the determinant of $E_i = I_i - xC_i$, where I_i is the $i \times i$ identity matrix. Then $\chi_A(\lambda) = \lambda^n \Delta_n(1/\lambda)$. First, express $1/\Delta_n(x)$ as a formal power series as follows: let $\Delta_0(x) \equiv 1$, then

$$\frac{1}{\Delta_n(x)} = \frac{\Delta_{n-1}(x)}{\Delta_n(x)} \cdot \frac{\Delta_{n-2}(x)}{\Delta_{n-1}(x)} \cdots \frac{\Delta_0(x)}{\Delta_1(x)}$$

But $\Delta_{i-1}(x)$ and $\Delta_i(x)$ are easily related using matrix inverses;

$$\frac{\Delta_{i-1}(x)}{\Delta_i(x)} = \frac{\det(E_{i-1})}{\det(E_i)} = (E_i^{-1})[1,1]$$

Furthermore, it is easy to verify that $E_i^{-1} = (I_i - xC_i)^{-1} = \sum_{j=0}^{\infty} x^j C_i^j$. So

$$\frac{1}{\Delta_n(x)} = \left(\sum_{j=0}^{\infty} x^j (C_n^j)[1,1]\right)\left(\sum_{j=0}^{\infty} x^j (C_{n-1}^j)[1,1]\right)\dots\left(\sum_{j=0}^{\infty} x^j (C_1^j)[1,1]\right)$$

Let f_j be the coefficient of x^j in $1/\Delta_n(x)$.

Now, since $\Delta_n(x) \times 1/\Delta_n(x) \equiv 1$, all coefficients other than that of the constant term must be 0. This gives us equations relating the coefficients of $\Delta_n(x)$, and hence of $\chi_A(\lambda)$, to those of $1/\Delta_n(x)$. Let $1/\Delta_n(x) = 1 - xH(x)$, where $H(x) = -(f_1 + f_2 x + f_3 x^2 + \dots)$. Then

$$\Delta_n(x) = \sum_{i \geq 0} x^i \left[H(x)\right]^i = c_0 x^n + c_1 x^{n-1} + \dots + c_{n-1} x + c_n$$

So c_{n-m}, the coefficient of x^m in $\Delta_n(x)$, is given by

$$\begin{aligned} c_{n-m} &= \textstyle\sum_{i \geq 0} \text{ coefficient of } x^m \text{ in } x^i \left[H(x)\right]^i \\ &= \textstyle\sum_{i=0}^{m} \text{ coefficient of } x^{m-i} \text{ in } \left[H(x)\right]^i \end{aligned}$$

Since only the coefficients up to x^n of any power of $H(x)$ are used, the entire computation (of $1/\Delta_n(x)$ and $\Delta_n(x)$) may be done mod x^{n+1}, giving an NC algorithm.

Note that the expression for $1/\Delta_n(x)$ obtained above is precisely the power series $D(x)$ we defined to compute the contributions of j-tour sequences.

3.4. Relating Tours and Cycle Covers: Getting to Csanky's Algorithm. We now consider the most unstructured generalization of a cycle: we relax the condition that a tour must begin from an occurrence of the minimum vertex. All we are interested in is a closed path, and we call such paths loops. Formally,

DEFINITION 3.7. *A loop at vertex v is a walk from v to v. i.e. a loop L is an ordered sequence of edges $L = \langle e_1, e_2, \dots, e_p \rangle$ such that $e_i = \langle u_i, u_{i+1} \rangle$ for $i \in [p-1]$ and $u_{p+1} = u_1$. The loop has length p and weight $\prod_{i=1}^{p} w(e_i)$.*

Having relaxed the structure of a loop, we now severely limit the way in which loops can be combined in sequences. A loop may be combined only with a partial cycle cover. Similar in spirit to Theorems 3.2, 3.4, 3.6, we now show cancellations amongst such combinations.

THEOREM 3.8. *For $k \in \{1, \dots, n\}$,*

$$kc_k + \sum_{j=1}^{k} c_{k-j} \left(\sum_{L \text{ is a loop of length } j \text{ in } G_A} w(L) \right) = 0$$

It is easy to see that $A^j[i,i]$ sums the weights of all paths of length j from i to i in G_A. Such paths are loops; so $\sum_{i=1}^{n} A^j[i,i]$ sums the weights of all loops of length j in G_A. But $\sum_{i=1}^{n} A^j[i,i] = s_j$, the trace of the matrix A^j. Thus the above theorem is merely Leverier's Lemma, usually stated as follows:

Leverier's Lemma [8, 13]: *The coefficients of the characteristic polynomial of a matrix A satisfy the following equalities:*

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ s_1 & 2 & 0 & \cdots & 0 & 0 \\ s_2 & s_1 & 3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & 0 \\ s_{n-2} & s_{n-3} & s_{n-4} & \cdots & n-1 & 0 \\ s_{n-1} & s_{n-2} & s_{n-3} & \cdots & s_1 & n \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ \vdots \\ c_n \end{bmatrix} = - \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ \vdots \\ s_n \end{bmatrix}$$

where s_j is the trace of the matrix A_j.

A combinatorial proof of the above lemma.
Consider the kth claimed equality:

$$kc_k + \sum_{j=1}^{k} s_j c_{k-j} = 0$$

where $c_0 = 1$. The terms contributing to $\sum_{j=1}^{k} s_j c_{k-j}$ consist of loops of length j and partial cycle covers of length $k - j$. The loops carry only a weight but no sign, whereas the partial cycle covers are weighted and signed. We show how to achieve cancellations within this set.

Let S be a loop of length j, and let $\mathcal{C}$ be a $(k-j)$-cycle cover.
Case 1: S forms a simple cycle, disjoint from all the cycles in $\mathcal{C}$. In this case, S can be merged into C to form a k-cycle cover $\mathcal{C}'$, with weight $wt(S)wt(\mathcal{C})$ and sign $-sgn(\mathcal{C})$. This will cancel against one copy of $\mathcal{C}'$ coming from the kc_k part. What about the $k-1$ remaining copies? Note that if $\mathcal{C}' = \langle C_1, \ldots, C_l \rangle$, then each C_i can be pulled out to give a partition into a loop S and a cycle cover C, cancelling against the corresponding term from $s_{l_i} c_{k-l_i}$. Furthermore, each C_i can be written as a loop in l_i different ways, depending on the starting point. So $\mathcal{C}'$ gives rise to $\sum_{i=1}^{l} |l_i| = k$ pairs of the form (loop, partial cycle cover); hence the term kc_k is accounted for.
Case 2: S and $\mathcal{C}$ cannot be merged into a k-cycle cover. Start traversing the loop S until one of the following things happen:

1. S hits a vertex v in $\mathcal{C}$.
2. S revisits a vertex v.

Only one of the two can happen first. Suppose S touches cycle C of $\mathcal{C}$. Let $|C| = l$. Consider the new pair $S', \mathcal{C}'$ where cycle C is removed from $\mathcal{C}$ and inserted in S at the first possible position v. This pair contributes the same weight with opposite sign to the term $s_{j+l} c_{k-j-l}$, and these two terms cancel out. Now suppose S completes a simple cycle C of length l within itself without touching $\mathcal{C}$. Consider the new pair $S', \mathcal{C}'$ where cycle C is removed from S and inserted in $\mathcal{C}$ at the appropriate position. This pair contributes the same weight with opposite sign to the term $s_{j-l} c_{k-j+l}$ where $|C| = l$, and these two terms cancel out. $\square$

Algorithm using loops. Any algorithm for computing $\chi_A(\lambda)$ that uses Leverier's Lemma implicitly exploits cancellations among loops and partial cycle covers in computing a sum which evaluates to precisely l-cycle covers. A straightforward sequential algorithm is to first compute for each j, the sum s_j, of the weights of all loops of length j in G_A using either matrix powering or dynamic programming, and then to compute $c_1, c_2, \ldots, c_n$ in order using the recurrence. Csanky's implementation [6] directly uses matrix inversion to compute the c_j's in parallel from the values of s_i's.

Note that l-loops are closed paths of length l with *no* restriction on the ordering of the edges. As sequences of edges, they thus subsume tours, clows and cycles. In this sense, Csanky's algorithm is more extravagant than the others described above. On the other hand, it is the most frugal in allowing combinations; a loop may only be combined with a partial cycle cover and not with other loops.

4. Discussion. Starting with the definition of the coefficients of the characteristic polynomial as the signed weighted sum of all partial cycle covers, we have considered several ways of expanding the summation while keeping the net contribution the same. In a sense, the expansion corresponding to clow sequences with the prefix property, as generated by Samuelson's method, is the most conservative. All the other expansions we have considered include these sequences and more. (A clow is a tour is a loop, but not vice versa.) There are smaller expansions that still cancel out nicely (for instance, consider clow sequences where $C_i \cap C_j \neq \phi$ for at most one pair i, j). However, these smaller expansions do not seem to yield efficient computational methods. Can this observation be formally proved, i.e., can one show that any efficient method for computing c_l must include at least the l-clow sequences with the prefix property?

One of the oldest methods for computing the determinant is Gaussian elimination. Strassen ([21], or see the textbook presentation in [14]) shows how to obtain a division-free code corresponding to Gaussian elimination. Can this method also be interpreted combinatorially?

If we assume that addition, subtraction, multiplication and division are unit-cost operations, then Gaussian elimination remains one of the most efficent methods for computing the determinant. Can this efficiency be explained combinatorially? Strassen's interpretation uses formal power series expansions, and shows how Gaussian elimination uses the entire power series rather than a truncated polynomial. So high degree monomials are generated, corresponding to sequences of clows or tours or loops of arbitrary length, not just restricted to n. Is this where the computational advantage lies — do higher degrees help?

Acknowledgment. We thank Ashok Subramanian for showing how our combinatorial interpretations and construction in the clow sequences approach extend to Chistov's algorithm as well. We thank an anonymous referee for bringing the references [2, 3, 18] to our notice and for providing several comments that have improved the readability of this paper.

REFERENCES

[1] S. J. Berkowitz, *On computing the determinant in small parallel time using a small number of processors*, Information Processing Letters, 18 (1984), pp. 147–150.

[2] R. A. Brualdi, *The many facets of combinatorial matrix theory*, in Matrix Theory and Applications, C. R. Johnson, ed., 1990. Proc. Symposia Pure and Applied Math. vol. 40, Amer. Math. Soc., Providence.

[3] R. A. Brualdi and H. Ryser, *Combinatorial Matrix Theory*, Cambridge University Press, 1991. Vol. 39 in Encyclopedia of Mathematics and its Applications.

[4] S. Chaiken, *A combinatorial proof of the all minors matrix theorem*, SIAM J. Algebraic Discrete Methods, 3 (1982), pp. 319–329.

[5] A. L. Chistov, *Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic*, in Proc Int. Conf. Foundations of Computation Theory, LNCS 199, Springer, 1985, pp. 63–69.

[6] L. Csanky, *Fast parallel inversion algorithm*, SIAM J of Computing, 5 (1976), pp. 818–823.

[7] C. Damm, *DET=L*$^{(\#\mathrm{L})}$, Tech. Report Informatik–Preprint 8, Fachbereich Informatik der Humboldt–Universität zu Berlin, 1991.
[8] D. Fadeev and V. Fadeeva, *Computational Methods in Linear Algebra*, Freeman, San Francisco, 1963.
[9] D. Foata, *Etude algébrique de certains problèmes d'analyse combinatoire et du calcul des probabilités*, Publ. Inst. Statist. Univ. Paris, 14 (1965), pp. 81–241.
[10] ———, *A combinatorial proof of Jacobi's identity*, Ann. Discrete Math., 6 (1980), pp. 125–135.
[11] A. Garsia and O. Eğecioğlu, *Combinatorial foundations of computer science.* unpublished collection.
[12] I. Gessel, *Tournaments and Vandermonde's determinant*, J Graph Theory, 3 (1979), pp. 305–307.
[13] D. Kozen, *The Design and Analysis of Algorithms*, Springer-Verlag, New York, 1992.
[14] T. F. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann Publishers Inc., San Mateo, 1992.
[15] M. Mahajan and V. Vinay, *Determinant: combinatorics, algorithms, complexity*, Chicago Journal of Theoretical Computer Science `http://cs-www.uchicago.edu/publications/cjtcs`, 1997 (1997), p. 5. A preliminary version appeared as "A combinatorial algorithm for the determinant" in Proceedings of the *Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* SODA97.
[16] M. Minoux, *Bideterminants, arborescences and extension of the matrix-tree theorem to semirings*, Discrete Mathematics, 171 (1997), pp. 191–200.
[17] J. B. Orlin, *Line-digraphs, arborescences, and theorems of Tutte and Knuth*, J. Combin. Theory Ser. B, 25 (1978), pp. 187–198.
[18] D. E. Rutherford, *The Cayley-Hamilton theorem for semirings*, Proc. Royal Society of Edinburgh Sec. A, 66 (1964), pp. 211–215.
[19] P. A. Samuelson, *A method of determining explicitly the coefficients of the characteristic polynomial*, Ann. Math. Stat., 13 (1942), pp. 424–429.
[20] D. Stanton and D. White, *Constructive Combinatorics*, Springer-Verlag, 1986.
[21] V. Strassen, *Vermeidung von divisionen.*, Journal of Reine U. Angew Math, 264 (1973), pp. 182–202.
[22] H. Straubing, *A combinatorial proof of the Cayley-Hamilton theorem*, Discrete Maths., 43 (1983), pp. 273–279.
[23] H. N. V. Tempereley, *Graph Theory and Applications*, Ellis Horwood, Chichester, 1981.
[24] S. Toda, *Counting problems computationally equivalent to the determinant.* manuscript, 1991.
[25] L. G. Valiant, *Why is boolean complexity theory difficult?*, in Boolean Function Complexity, M. S. Paterson, ed., Cambridge University Press, 1992. London Mathematical Society Lecture Notes Series 169.
[26] V. Vinay, *Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits*, in Proc. 6th Structure in Complexity Theory Conference, 1991, pp. 270–284.
[27] D. Zeilberger, *A combinatorial approach to matrix algebra*, Discrete Mathematics, 56 (1985), pp. 61–72.