*IOSR Journal of Computer Engineering (IOSR-JCE)*
*e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 23, Issue 6, Ser. I (Nov. –Dec. 2021), PP 60-68*
*www.iosrjournals.org*

# The Complexity of Computing The Permanent

Ms Anveshika Srivastava
*Enrollment No- MUR2001590*
*Submitted- Mewar University, Chittorgarh July, 2022*

Abstract. it is shown that the permanent function of (0, 1) matrices is a complete problem for the class of counting problems associated with non-deterministic polynomial time computations. Related counting problems are also considered. The reductions used are characterized by their nontrivial use of arithmetic.

Let A be a n x n matrix. The *Permanent* of A is defined as

$$\textbf{Perm } A = \sum_{\sigma} \prod_{i=1}^{n} A_{i,\sigma(i)}$$

where the summation is over the n! permutations of (1, 2, ... , n). It is the same as the *determinant* except that all the terms have positive sign. Despite this similarity, while there are efficient algorithms for computing the determinant all known methods for evaluating the permanent take exponential time. This discrepancy is annoyingly obvious even for small matrices, and has been noted repeatedly in the literature since the last century [15]. Several attempts have been made to determine whether the permanent could be reduced to the determinant via some simple matrix transformation. The results have always been negative, except in certain special cases [12, 13, 16].

The aim of this paper is to explain the apparent intractability of the permanent by showing that it is “complete” as far as counting problems. The results can be summarized informally as follows:

**Theorem *1.*** *The complexity of computing the permanent of n n* (0, 1)-motricer *is NP-hard* [3, 11] *and, in fact, of at least as great difficulty {to within a polynomial factor) as that* o/ *counting the number of* accepting *computations of any nondeterministic Polynomiul time tuiring machine.*

**Theorem 2.** *For any integer K that is not an exact power of two the complexity of comp@ing the permanent of a* (0, I)-matrix mod *K is UP-hard* [24] *(i.e. • R**v -nomial time algorithm for it would imply that any single -i› valued function that can be checked Position can* a/so *be evaluated false).*

**Theorem *3.*** for ony *integer k the permanent of an integer matrix* mod 2* *can be computed in* O(n $^{4}$'ᵒ') *steps if k* 2 *(and* O(n”") *steps if k* -- 1 *since the permanent and determinant are equal* mod 2).

To express Theorem 1 precisely we define the class *IP* of all problems computed by nondeterministic polynomial time Turing machines that have the additional facility of outputting the i‹ number of accepting computations. (N.B. This class is essentially equivalent to the polynomial time “probabilistic” TMs of Gill [4] and the “threshold” TMs of Simon [19]). For IP-complete problems counting the number of “solutions” (e.g. for satisfiability of propositional formulae we would count the number of satisfying assignments) is usually “4 P-complete” for trivial reasons. This is also the case for some polynomial computable problems that can be related to an IP-complete one in some direct fashion. The permanent function is a more surprising member of the 4 P-complete class since the natural correspondence is with counting sets of distinct representatives (or equivalently perfect matchings in bipaitite graphs) and for these the naturally related detection problems are polynomial time computable [5, 10].

On a more general level we establish a framework for classifying counting problems and dehne a hierarchy for this purpose. Examples of problems that one ma;' attempt to classify are the graph-enumeration problems [6]. An example of the immediate consequences of our results is the following: If A is a polynomial computable predicate, then the number of ' belled n-node graphs having property *R* can expressed “explicitly” as the permanent of a matrix that can be computed intime polynomial in n. Unfortunately, the best method 1. nown for evaluating the permanent of an *n* X *n* mai *iz* takes $2^{n*o}$ t‚o‚‚‚‚ steps [17, p. 26].

We do not know of any pair *ot* functions, other than the permanent and determinant, for which the explicit algebraic t:xpressions are so similar, and yet the computational complexities are apparently sc different. With this example we can understand better the difficulties that beset certain approaches to proving lower bounds on the complexity of unrestricted arithmetic (and Boolean) computations. Ar. optimistic hope, th•t is fulfilled in certain restricted cases [18], is that lower bounds can be proved by assigning complexity measures to intermediate results according to syntactic criteria on their expressions. Clearly any such measure that could so distinguish *P* irom *NP* would have to distinguish pairs of expressions that resemble each other as closely as the permanent and determinant.

## 2. The complexity of counting

For most non-deterministic algorithms each accepting computation corresponds in a natural way to a “solution” to the problem. It has been widely observed (e.g. [8, 19, 231) that for almost all pairs of NP-complete problems there exist poly-nomial transformations between them that preserve the number of solutions. These problems are therefore equivalent not only as far as the existence of solutions but also as far as the problem of counting the solutions.

**Definition 2.1. A** *counting Turing machine* is a standard nondeterministic TM with an auxiliary output device that (magically) prints in binary notation on a special tape the number of accepting computations induced by the input. It has (worst-case) *time-complexity* /(n) if the longest accepting computation induced by the set of all inputs of size n takes /(n) steps (when the TM is regarded as a standard nondeterministic machine with no auxiliary device).

**Definition 2.2.** H P is the class of functions that can be computed by counting TMs of polynomial time complexity.

We denote the r.glass of functions computed by deterministic polynomial time TMs by FP, and the class of predicates by P or *DP*. For convenience we shall often identify a class of machines with the class of functions it computes. It will be assumed that objects are represented in some standard economical manner as words over an alphabet ñ (say (0, 1}). |x| will denote the size of z if is a set, and its length if z is a string. A function /: A* x A* (or a relation fi G i ‟ ) iS §O$\ *-nomial bounded* iff there is a polynomial *p* such that for all z |/(z)| < p(|z |) (or such that fl(z, y) W |y| < *P(\x\)).*

The notion of reduction used is one by oracles, in a similar sense to Cook [3] except thai the oracles cannot only be predicates but also arbitrary polynomial bounded functions. An oracle TM is a TM with a query tape, an answer tape, and some working tapes. To consult the oracle the TM prints a word on the query tape and, on going into a special query state an answer is returned in unit time on the answer tape, and a special answer state entered. An oracle TM is said to be in *DP* (or NP. or NP, or # P, etc.) ifl for all polynomial bounded oracles it behaves like a machine in P (or FP, or NP, or # *P,* etc.).

IN o is a class ot oracle-TMs and x.an appropriate function for it fi.e. polynomial bounded in the present conteX:J then we denote the class of function (or predicates) that can be computed by ora‹'le-TMs from n with oracles for *x* by a". A problem v is # *P-hard* itf # *PG FP’.* It is# *P-complete iR # P z FP’* and y e # *P*.

The following notation is i:seful for unifying the necessary concepts with those from [3, 11, 14]. Incl *? - [N, S , D,* F)+(P}. Each element of *r* will denote a complexity class de fined inductively starting from *DP*, *NP*, If' or # *P:* (i) If a e

**L.G. Valiant** and *Z a\N, 4 , D, F\,* then Zo = ,« 2P . (ii) If a e I‟ is a class of predicates, then co-‹r is the class of complements *ot* elements of a.

Clearly any occurrence of *D* or *F* in o e *r* s redundant except if it is the first letter. No other equivalences among the defined classes are known however. In the polynomial hierarchy [11, 14, 21] \;, {];‟ and Aï‟ respectively become N'P, co-N'P and *DN”’*¹P. The # symbols define a potentially inânite tree-hierarchy of classes which collapses if # P p IP. (N.B. NP= # P). However, we know of only such trivial containments as the following which hold for all n.

$$\alpha \cup \text{co-}\alpha \subseteq D\alpha \subseteq N\alpha \cap \text{co-}N\alpha \subseteq N\alpha \cup \text{co-}N\alpha \subseteq D\#\alpha$$

In general a problem z is n *-Carl* via Q ifl n n Q,„ and is n *-appropriate Q* iff in addition x e n. We assume throughout that Q is *P* or *IP* as :ippropriate. In this paper we are concerned mainly with I P-completeness. We shall also observe that there are natural problems complete in # NP, and also that certain well-known problems occur lower in the *N -# -P* hierarchy than is immediately apparent.

For IP-complete problems proving # P-completeness for their natural counter-parts is usually easy. Finding related polynomial time problems for which counting is still # f•-complete is also easy. As an artificial example consider a Boolean conjunctive normal form formula N. Finding an assignment that makes *F* false is trivial, yet counting them is4 I -complete. As a better example consider a mono-tone Boolean formula. Detecting the existence of a satisfying assignment is trivial. However, since any N can be rewritten, without changing the number of solutions as Cr a Off (where G, H are monotone), counting the solutions of G » ff and of G would give the number of solutions of *F.* The rewriting consists of replacing each unnegated variable z; in *F* by new variable y,, and each negated variable I; by r;, and conjoining *F* with

$$\bigwedge_i (y_i \vee z_i) \wedge \neg \bigvee_i (y_i \wedge z_i).$$

A nondeterministic "furing machine is *unambiguous* [24] if for any input there is at most one accepting computation. UP is the class of predicates computed by unambiguous polynomial time "FMs. The significance of the question f •- ? Lff• is that a positive answer would imply that the problems of checking and evaluating are polynomially related for all single-valued functions [24]. There are some problems (e.g. computing the prime decomposition of an integer) for which this is currently conjectured by some not to be true. Theorem 2 can be interprcedeted therefore either as giving evidence of intractability in a circumstance where h'f'-hardness is in doubt, or alternatively as a possible route to a powerful positive result. In particular, a fast algorithm for computing permanents mod 3 would imply that such schemes for cryptography as those proposed in [26] are nonexistent.

It is relevant to observe that among 4 I-complete problems several finer dis-tinctions can be made. A problem is *P-enumerable* if all the solutions can be listed in time p(n) *N* where /V is the number of solutions found and *p(n)* some poly-nomial in the input size. While matchings in biparite graphs and satisfying assign-ments in monotone circuits are p-enumerable, IP-complete problems will not be in general unless f• = £fP. Another distinction involves dehning # *JP* to be# *P* but with arithmetic modulo *K.* Our results show that the permanent mod *K* is *# z P*-complete if *K* is not a power of 2, but is polynomial time computable otherwise.

## 3. Results and proofs

The main reduction used is the following:

Lemma *There ic a function f o IP from propotional formulae in conjunctive normal form to matrices with entries/rom* (—1, 0, 1, 2, 3} *such that*

$$\forall F \quad \mathrm{Perm}(f(F)) = 4^{t(F)} \cdot s(F)$$

*where UF) dentes "twice the number of occurrences of liberals in F, minus the number of classes in F", and s{F) is the number of assignments that satisfy F.*

To prove H P-hardness for the permanent of general integer matrices we need only the following additional fact.

Lemma 3.2. *There is a* /unriion g e *FP that maps* an *arbitrary NP* TM .U *nnd an input x for it to a propositionnl formula in 3-coni•• ctioe normal form such that the number of* saris/ying *assignmentc of g(M, x) is* equal *to the number of accepting computations* o/M on x.

Proof. By a modification of Ccuk"s construction [3] using the idea of [2]. See [19] or [251-

To obtain results for (0, 1)-matrices we also need the following:

**Lemma 3.3.** there is e *transformation h, computable its. time polynomial in m and the order of the matric, that maps matrices with elements from the set* {0, 1, *, m] to* (0, *I)-matrices such that*

¥A Perm A = Perm h(H).

Before proving Lemmas 3.1 and 3.3 we observe that Theorems 1 and 2 therom them.

194 **L.G. Valiant**

**Proposition 3.4.** *For* some *positive constant d the problem of computing the permanent* mod *r,* giren on *nñ n* (0, 1)-riairir *and* a *positivee integer r<dn* log2 », *is # P-hard.*

Proof. If *C* is an integer matrix in which no entry is larger than in magnitude then |Perm *C m y-* n! To compute Perm *C* it is sufficient to compute its value mod *p,* for each *p,* in some set ( rd. . p,} of distinct prime numbers whose product exceeds 2p- n! For some constant *d'* it is always sufficient that each p, d'n(!ogi *pn)* [7, p. 34a]. But by Lemma 3.3, for each p:, C' mod p, can be transformed in polynomial tax.e into a (0, 1)-matrix with the same permanent. The result therefore follows from Lemmas 3.1 and 3.2.

Theorem *L. Computing the permanent of a* (0, 1)-ma/rix *is 'P-complete.*

**Proof.** Proposition 3.4 implies that the problem is # P-hard. That it also belongs to A *P* is immediate.

**Theorem 2.** *For* ray *fixed positive integer K that is not an exact power of two, computing the permanent* mod *K of* (0, *I}-matrices is UP-hard.*

Proof. From Lemmas 3.1 and 3.2 given any machine 3f and an input z the permanent of th.e matrix *f(g{M, r)) will* equal either 4''"ᶜᶜ or zero according to whether *M* accepts x. Hence Perm /(g(M, z)) will be divisible by *K* if and only if Af does not accept x.

Proof of Lemma 3.1. Any n x *n* matrix *A* can be regarded as the adjacency matrix of an n-node weighted directed graph G where *A;,* gives the weight of the edge from node *i* to node /. Each additive term in Perm A corresponds to the product of the weights of the edges in some set of node-disjoint directed cycles that cover all the nodes of G (called “cycle covers” for short). To prove our result we exhibit a function / Such that in the graph associated with *f(F)* the cycle covers that cor-respond to satisfying assignments of N will each contribute 4",, to the permanent, while the contributions of all the “spurious” cycle covers will cancel each other out.
Let N=Ui *Cl* -I- - *n* C, where C. -true v y,, v y,t) with y;,e 1. Al. 2› 2. • • › xp, *,,,}. (N.B. This assumption of 3-form is not essential.) We
construct the graph G =/(N) by superposing th‹• following structures: a *truck* Fi for earh variable z,, an *interchange Ri* for each clause *Cl,* and, for each literal *y;.;* Such that v;.; is xt or *,, a *junction J;,z* at which fi; and *Tz* meet. Interchanges also have *internal junctions* of the same structure as junctions. The construction of the tracks and interchanges is taken from a proof in [23] which itself is adapted from one in [9]. We describe them in Fig. ! by an example fragment. 'fi5 £tnd fi 3 fi re shown for

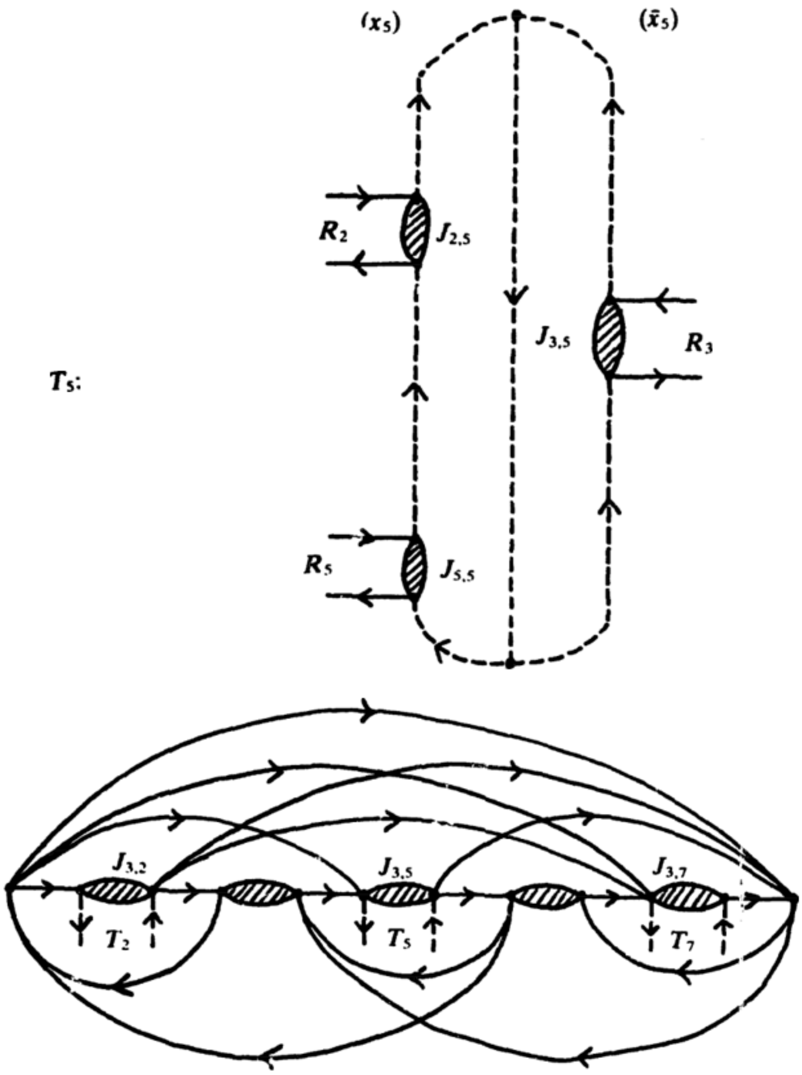


Fig. 1. A track and an interchange.

the case where C› -- (<2 *5 v <7). and where xs occurs in *Cz* and C,, and * in +3. We assume that all the edges outside junctions or internal junctions are weighted one.

The crucial part ot the construction is the structure of the junctions. The junctions and internal junctions are all identical four-node weighted digraphs corresponding to the following 4 x 4 matrix X.

$$\begin{pmatrix} 0 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 1 & 3 & 0 \end{pmatrix}$$



Each one has external connections only via nodes 1 and 4 and not via 2 or 3. Denoting by A(y; fi) the matrix A with rows y and columns 6 removed, the
following properties of N can be verified: (i) Perm A =0
(ii) Perm X(1; 1)= 0 (iii) Perm X(4; 4)= 0 (iv) Perm X(1, 4; 1, 4)= 0
(v) Perm X(1; 4)= Perm A(4; 1) = nonzero constant (=4)

[N.B. The given X is about the simplest possible among all matrices with properties (i}-(v) if (1, 4) is taken to denote an arbitrary pair of indices, and any nonzero constant is allowed in (v). This can be seen from the following easily proved facts: (a) any such matrix has to be at least 4 x 4, and if it is 4 x 4 then (b) it is not symmetric and (c) at least two entries are greater than one in magnitude. Also (d) rio matrix can have the same set of properties for the determinant, and hence if ii has them for the permanent the constant in (v) must be even.]

Let a *route* in the graph /(N) be the set of all cycle covers that have the same set of edges outside the junctions. A route is *good* if every junction and internal junction is entered exactly once the left exactly once and at the opposite end. Routes may fail to be good either because (a) some junction or internal junction is not entered and left, or (b) because it is entered and left at the same end, or (c) because it is entered twice and left twice. By virtue of con‹5itions (i) for (a), (ii) and (iii) for (b) and (iv) for (c) any route that. is not good contributes zero to the permanent. Condition (v) ensures that any good route contributes exactly 4"" .

It is clear that in any track *Tz* of any good route either all junctions on the left are "picked up" by the track and all the ones on the right by interchanges, or vice versa (corresponding to xi and ?i respectively). The interchanges are so constructed thai any route can pick up any subset of them, except for the whole set itself. Further-more it can do so in exactly one way for each subset. 3"hus if for some At at leasi one of the junctions is picked up by the tracks, then all the remaining ones will be picked up by ft; in the unique good route available.

Using the obvious correspondence be*seen good routes and assignments of iruth values, we conclude that there is a one—one correspondence between good routes in the graph, each of which contributes 4""" to the permanent, and satisfying assign-ment of F. The result follows.

Proof of Lemma 3.3. To obtain 6(A) we replace each edge of weight $k > 1$ in *A* by a subgraph. The subgraph is illustrated in Fig. 2 for the case *k* -- 5. It replaces an edge of weight 5 from node x to node y. All the other nodes shown are new

d CtLt)t°OFIS tO .

Now if (.x, y) is not covered by a cycle in *A* then there is just one way to cover the corresponding new nodes in *li(A).* On the other hand, i' (z, y) is covered by a cycle in A, then so must be also the chain of these edges fr rim x to y in fi(A j. Then there

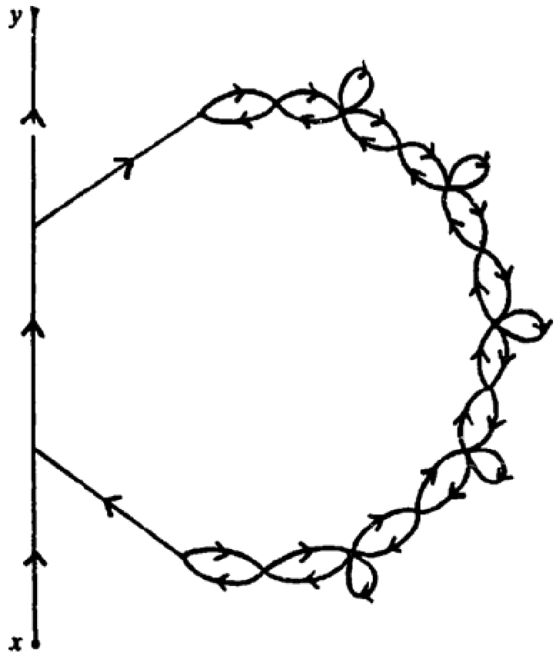


Fig. 2.

are hve ways of covering the remainder, each corresponding to the inclusion of a different self-loop.

To obtain our main positive result we generalise the Gaussian elimination technique.

**Theorem** ***3s*** *Let k be any positive integer and let Oz(n) be the number of bitwise* operations *required* to *evaluate the permanent* mod 2 , *for n z n integer matrices.* Then O›(n)= O(n"*')/or £ w 2 and Or(n)= Ci(n' ')

Proof. The proof is by induction on matrices of the two types shown in Fig. 3. *Fx* n) and G (n) are the complexities of evaluating the permanent of the hrst and second types respectively. We shall assume arithmetic mod 2,, throughout.

To evtiuate the permanent of a matrix of the second type we scan the columns from left to right for i = 1, ... , n — g. For each value of *i,* for every set of i of the first *i* + g rows, we compute the permanent of the submatrix formed ay these rows and the first *i* columns Since, for each *i,* there are fewer than n' such submatric‹•s, we have tower than n' values to compute at each stage. Each value at stag.e is the sum of some of the values at stage *i* — 1. Furthermore, each value at stage i — 1 influences at most g + 1 values at stage i (corresponding to the possible choices of rows omitted at stage *i* — 1 but used at stage *i).* Hence the total complexity is O(n'+'). The task is completed in the same fashion for the last g columns, givinp, Gz(n) = (<”')



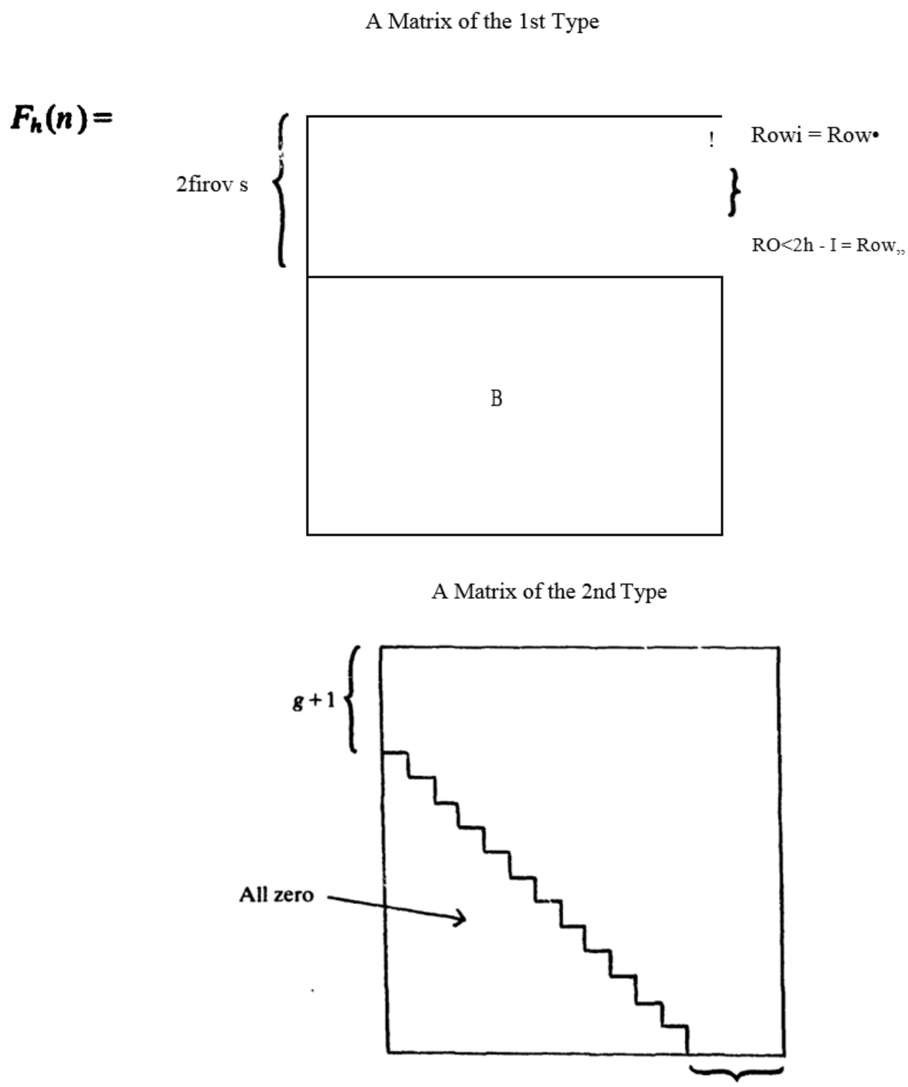


Fig. 3.

To evaluate the permanent of a matrix of the first kind we perform “elimination" on it to give a matrix of the second kind, plus n" matrices of the hrs" kind with *h* increased by one.

The c1lmination is based on the fact that if by adding row i to row *i* in an integer matrix A we obtain A", then

Perm A" = Perm *A* + Perm A“

*be complexity of* co*f*ft*pfiu in& ihe pe anent*

where A“ is the same as A but with row *¡* replaced by row *i. A“* must then have two equal rows and hence Perm A“ -— 0(mod 2). (Note that *Fg(n)* = 0 for a similar
reason: the permanent of any matrix with k dis]oint pairs of equal rows is a multiple of 2 .)
We can therefore reduce the *{n* - 2h) *z n* submatrix *B* (i.e. the bottom n —2fi
rows) so that *EU* becomes zero for *i >¡* and hence the overall matrix becomes of the second type. The reduction consists of computing O(n $^{2}$) linear combinations of pairs of rows in the manner of standard Gaussian elimination except that now we get an additive term of the first kind (with â increased by one) at each step. As pivot we always choose from among the elements of the relevant submatrix one that is odd, or if no odd element exists, then one tha: has the fewest factors of two. This ensures that we can work in arithmetic mod 2'. Hence we have

$$F_h(n) = O(n^3) + O(n^2)F_{h+1}(n) + G_{2h}(n),$$

and

'•t(n)= 0.

Solving this gives o(<) - O(f2“„$^{3}$) for all *k* w 2. (The case *k* -- 1 gives O(n') Gaussian elimination, which can be improved to O(n $^{2}$“") [22]).

4. Remarks

For many predicates in P the corresponding counting problem is in *FP*. This is evident, for example, for problems with a dynamic programming flavour. More interesting examples are spanning trees in graphs, and Eulerian paths in diagraphs, either of which can be counted by evaluating appropriate determinants [6].
For many other easily computed predicates the counting problem is # *P*-complete. This is true for satisfiability even in the very restricted case of monotone CNF formulae with two disjuncts per conjunct. (A collection of such complete problems appears in [25].) Note that good proofs of # P-completeness appear to be characterized by the necessity for nontrivial arithmetic.
Isomorphism of graphs has a seemingly intermediate position: R. Mathon has recently observed that counting isomorphic embeddings belongs to FNP, and hence that the full power of counting (i.e. #P) is not apparently required.
The above result is relevant ii› classifying enumeration problems [6]: Let z e *NP* and consider ihe problem C,: “given a graph G count the number of subgraphs with property z that are distinct under relabellings”. Many natural enumeration problems are of this flavour for the special case that G is the complete graph.

Proof. Run a H P machine *M* of which each nondeterrriinistic branch performs the following: guess a subgraph G“, test it for x, and if this is true then run a nondeter-ministic computation with g accepting branches for

g = *h* {mod *p)*

where fi is the number of automorphisms of G' and p is a computed prime number
larger than the expected answer. Finding p and â can be done within f?VP [27].

As a final remark we note that the following problem NSAT can be easily shown to be complete in I *NP:* A propositional formula *F* can be regarded as being nondeterministic if its variables are partitioned into set X and K. A *solution* to the NSAT problem is an assignment of values to the elements of X for which there exists an assignment of values to Y that together satisfaction. The counting problem for FJSAT is that of counting the number of such solutions.

## References


[1]. A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms (Addison-Wesley, Reading, MA, 197 ).

[2]. M. Bauer. D. Brand, M. Fischer, A. Meyer and M. Paterson, A note on disjunctive form tautologies. StGACT News 52 (1973) 17-20.

[3]. S.A. Cook, The complexity of theorem proving procedures, Pror. 3rd ACW $ymp. on Theory of Computing (1971) 151-158.
[4]. J. Gill, Computational complexity of probabilistic Turing machines, Doc. 6iâ AC3f Sump. on Theory of Computing (1974) 91-95.
[5]. hL Hall Jr., An algorithm for distinct representatives, Am. Math. Monthly 63 (1956) 716—717. [6] F. Harary and E.M. Palmer, Graphical Enumeratio n (Academic Press, Nen' York,1973).
[6]. G.H.HardyandE.M.Wright,AnIntroductiontotheTheoryofNumbers(OxfordUniversityPress, 4th Ed., 19h0).
[7]. J. Hartmanis and L. Berman, On Isomorphisms and density of NP and other complete sets, Proc. 8th AC ñf Sump. on Theory of Computing (1976) 30—40.
[8]. P.P. Herrmann, On reducibility among combinatorial problems. MAC TR-113, MIT(1973).
[9]. J.E. Hopcroft and R.SL Karp, An n"' algorithm for maximum matchings in bipartite graphs, SIAM J. Compiir. 2 (1973) 225-231.
[10]. R.M. Karp, Reducibility among eombinatorial problems, in: R.E. Miller and J.W. Thatcher, Eds., Complexity of Computer Computations (Plenum Piess, New York, 1972). ImlC.IN.C.Li:tle,Acharacterizationofconvertible(0,1)-matrices,J.Comhin.Theory18(Bj(1975) 187—208.
[11]. M. Marcus and H. Mine, On the relation between the determinant and the permanent, Illinois J. Math. 5 (1961) 376-381.
[12]. A.R. Meyer and L.J. Stoc'‹meyer, The equivalence problem for regular expressions with squaring requires exponential space, Proc. l3th 8ymp. on Switching and Automate Titeory (1972)125-129.
[13]. T. Muir, On a class of permanent symi.:etric functions, Plec. boy. Soc. fidioéurg6 11 (1882) 409—418.
[14]. G. Polya, Aufgabe 424, Arch. Math. Phys. (3) 2.0 (1913) 27.
[15]. H.J. Ryser, Combinatoriol Mathematics, Cams Math. Monograph no. 14 (1963).
[16]. C.P. Schnorr, A lower bound on the number of ,additions in monotone computations, Theos. Comput. Sci. 2 (1976) 305-315.
[17]. J. Simon, On some central problems in computational complexity, Ph.D. Thesis, TR75-224, Cornell Univ. (1975).
[18]. J.Simon,Onthedifferencebetweenoneandmany,Proc.4ihColloq.onAn.•omata,Languagesand fYogrammii:g, Turku.1977.
[19]. [2t) L.J. Stockmeyer, The polynomial-time hierarchy, Theorem. Comput. dci. 3 (1977) 1-22. [22] V. Strassen, Gaussian elimination is not optimal, Numer. ñfnift. 13 (1969) 354-356.
[20]. L.G. Valiant. A polynomial reduction of satisfiability to Hamiltonian circuits that preserves the number of solutions, Manuscript, Univ. of Leeds (1974).
[21]. L.G. Valiant, The relative complexity of checking and evaluating, fit/ormoiion Processing Lett. 5(1) (1976} 20-23.
[22]. L.G.Valiant.Thecomplexityofenumerationandreliabilityproblems,$IAMS.Comput.(toappear).
[23]. W.DiffieandM.E.Hellman,Newdirectionsincryptography, IffEE.TransInformation 27ieory (New York (Hov. 1976).
[24]. t27) V.R. Prati, Every prime has a succint certihcate, DIAM J. Comput. 4(t975) 214—220.
[25]. R.L.Rivest,A.ShamirandL.Adleman,Ondigitialsignaturesandpublic-keycryptosystems,TM 82, Lab. for Comput. Sci., MIT (April 1977).