London Mathematical Society ISSN 1461–1570

COUNTING UNLABELLED SUBTREES OF A TREE IS #P-COMPLETE

LESLIE ANN GOLDBERG AND MARK JERRUM

Abstract

The problem of counting unlabelled subtrees of a tree (i.e., subtrees that are distinct up to isomorphism) is #P-complete, and hence equivalent in computational difficulty to evaluating the permanent of a 0,1-matrix.

1. *Introduction*

Valiant's complexity class #P (see [**11**]) stands in relation to counting problems as NP does to decision problems. A function $f : \Sigma^* \to \mathbb{N}$ is in #P if there is a nondeterministic polynomial-time Turing machine M such that the number of accepting computations of M on input x is $f(x)$, for all $x \in \Sigma^*$. A counting problem, i.e., a function $f : \Sigma^* \to \mathbb{N}$, is said to be #P-*hard* if every function in #P is polynomial-time Turing reducible to f; it is *complete for* #P if, in addition, $f \in$ #P. A #P-complete problem is equivalent in computational difficulty to such problems as counting the number of satisfying assignments to a Boolean formula, or evaluating the permanent of a 0,1-matrix, which are widely believed to be intractable. For background information on #P and its completeness class, refer to one of the standard texts, e.g., [**3, 8**].

The main result of the paper—advertised in the abstract, and stated more formally below—is interesting on two counts. First, it provides a rare example of a natural question about trees which is unlikely to be polynomial-time solvable. (Two other examples are determining a vertex ordering of minimum bandwidth [**1, 4**], or determining the "harmonious chromatic number" [**2**].) Second, it is, as far as we are aware, the first intractability result concerning the counting of unlabelled structures.

Some definitions. By *rooted tree* (T,r) we simply mean a tree T with a distinguished vertex r, the *root*. An *embedding* of a tree T' in a tree T is a injective map ι from the vertex set of T' to the vertex set of T such that $(\iota(u),\iota(v))$ is an edge of T whenever (u,v) is a edge of T'. Sometimes T' and T will be rooted, in which case we may insist that ι maps the root r' of T' to the root r of T. We now define a sequence of problems leading to one of interest; we do not claim that both the intermediate problems are particularly natural.

Name. #BIPARTITEMATCHINGS.
Instance. A bipartite graph G with n vertices in each of its two vertex sets.
Output. The number of matchings of all sizes in G.

Name. #COMMONROOTEDSUBTREES.
Instance. Two rooted trees, (T_1,r_1) and (T_2,r_2).

This work was supported in part by the ESPRIT Working Group 21726 "RAND2" and by EPSRC grant GR/L60982.

.

1991 Mathematics Subject Classification 68Q25,68R10,05C05,05C30,05C60

Output. The number of distinct (up to isomorphism) rooted trees (T,r) such that (T,r) embeds in (T_1,r_1) and (T_2,r_2) with r mapped to r_1 and r_2, respectively.

Name. #ROOTEDSUBTREES.
Instance. A rooted tree, (T,r).
Output. The number of distinct (up to isomorphism) rooted trees (T',r') such that (T',r') embeds in (T,r) with r' mapped to r.

Name. #SUBTREES.
Instance. A tree T.
Output. The number of distinct (up to isomorphism) subtrees of T.

We will use each of the problem names in an obvious way to denote a function from instances to outputs: thus #ROOTEDSUBTREES(T,r) denotes the number of distinct rooted subtrees of the rooted tree (T,r). Our main result is the following.

Theorem 1. #SUBTREES *is* #P-*complete.*

Proof. The #P-hardness of #BIPARTITEMATCHINGS follows from Valiant's paper [**11**]. In particular, Valiant shows that the problem IMPERFECTMATCHINGS is #P-complete. IMPERFECTMATCHINGS is the same as #BIPARTITEMATCHINGS except that the size of the two vertex sets may differ. IMPERFECTMATCHINGS may be reduced to #BIPARTITEMATCHINGS by adding vertices to the smaller vertex set. Thus, #P-hardness of #SUBTREES follows from Lemmas 2–4 and from the transitivity of polynomial-time Turing reducibility. We will now show that #SUBTREES is in #P. Suppose that T is a tree with vertex set $V_n = \{v_0,\ldots,v_{n-1}\}$. We will order the vertices in V_n so that $v_i < v_j$ iff $i < j$. For every (labelled) subtree T' of T, let $V(T')$ denote the vertex set of T'. We will say that subtree T'' is *larger* than subtree T' if and only if there is a vertex $v_i \in V_n$ such that $v_i \in V(T'')$, $v_i \notin V(T')$ and

$$V(T') \cap \{v_{i+1},\ldots,v_n\} = V(T'') \cap \{v_{i+1},\ldots,v_n\}.$$

Let T'' be a subtree of T. Either T'' is the smallest subtree of T in its isomorphism class or there is a vertex $v_\ell \in V(T'')$ such that the sub-forest F_ℓ of T induced by vertex set

$$\{v_i \in V_n \mid v_i < v_\ell\} \cup \{v_i \in V(T'') \mid v_i > v_\ell\}$$

contains a tree isomorphic to T''. Thus, one can determine whether T'' is the smallest subtree of T in its isomorphism class by solving *subgraph isomorphism* with inputs F_ℓ and T'' for all $v_\ell \in V(T'')$. Since F_ℓ is a forest and T'' is a tree, this can be done in polynomial time [**3**] using the method of Edmonds and Matula. It is now simple to describe the #P computation: With input T, each branch picks a subtree T'' of T and rejects unless T'' is the smallest subtree of T in its isomorphism class.

□

2. *The reductions*

Denote by $\leqslant_T$ the relation "is polynomial-time Turing reducible to."

Lemma 2.

$$\#\textsc{BipartiteMatchings} \leqslant_T \#\textsc{CommonRootedSubtrees}.$$

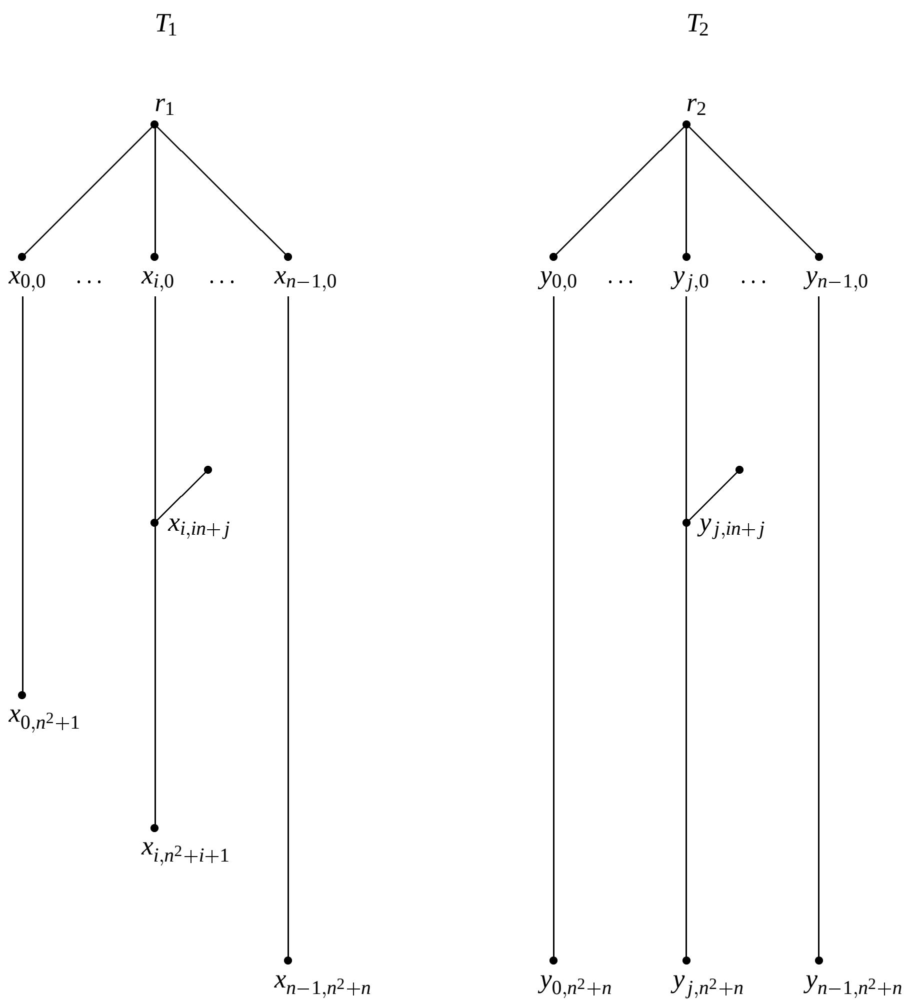

Figure 1: The skeleton of trees T_1 and T_2, illustrating the presence of edge (u_i, v_j) in G.

Proof. Let G be an instance of #BIPARTITEMATCHINGS with vertex sets $\{u_0, \ldots, u_{n-1}\}$ and $\{v_0, \ldots, v_{n-1}\}$. From G, we construct two rooted trees, (T_1, r_1) and (T_2, r_2), each based on a fixed skeleton. The skeleton of T_1 has vertex set

$$\{x_{i,j} : 0 \leqslant i \leqslant n-1 \text{ and } 0 \leqslant j \leqslant n^2 + i + 1\} \cup \{r_1\},$$

and edge set

$$\{(x_{i,j}, x_{i,j+1}) : 0 \leqslant i \leqslant n-1 \text{ and } 0 \leqslant j \leqslant n^2 + i\} \cup \{(r_1, x_{i,0}) : 0 \leqslant i \leqslant n-1\}.$$

Informally, the skeleton of T_1 consists of n paths of different lengths emanating from the root r_1, as illustrated in Figure 1. These n paths correspond to the n vertices $\{u_i\}$ of G.

The skeleton of T_2 is similar to the skeleton of T_1, except the paths now have equal length. It has vertex set

$$\{y_{i,j} : 0 \leqslant i \leqslant n-1 \text{ and } 0 \leqslant j \leqslant n^2+n\} \cup \{r_2\},$$

and edge set

$$\{(y_{i,j}, y_{i,j+1}) : 0 \leqslant i \leqslant n-1 \text{ and } 0 \leqslant j \leqslant n^2+n-1\} \cup \{(r_2, y_{i,0}) : 0 \leqslant i \leqslant n-1\}.$$

The n paths emanating from r_2 correspond to the to the n vertices $\{v_i\}$ of G.

The trees T_1 and T_2 themselves are built by adding to the respective skeletons certain edges encoding the graph G. Specifically, for each edge (u_i, v_j) of G, we add an edge from a new vertex to vertex $x_{i,in+j}$ of T_1 and add an edge from a new vertex to vertex $y_{j,in+j}$ of T_2.

Let $\mathcal{T}^*$ denote the set of all finite (unlabelled) rooted trees (T,r) that have leaves at all distances in the range $[n^2+2, n^2+n+1]$ from the root r. For any rooted tree (T,r), let $\mathcal{T}(T,r)$ denote the set of all (unlabelled) rooted subtrees of (T,r). Thus, the quantity #RootedSubtrees(T,r) is just the size of $\mathcal{T}(T,r)$. We first observe that there is a bijection between the set of matchings (of all sizes) in G and the set $\mathcal{T}(T_1,r_1) \cap \mathcal{T}(T_2,r_2) \cap \mathcal{T}^*$, and then conclude the proof by showing how to compute the size of $\mathcal{T}(T_1,r_1) \cap \mathcal{T}(T_2,r_2) \cap \mathcal{T}^*$ using an oracle for #CommonRootedSubtrees.

Consider some tree $(T,r) \in \mathcal{T}(T_1,r_1) \cap \mathcal{T}(T_2,r_2) \cap \mathcal{T}^*$. From the definition of $\mathcal{T}^*$ we see that T must contain the entire skeleton of T_1. Let us now see which other edges of T_1 can be present in T. That is, we will now consider the "pendant edges" which hang off of the skeleton of T_1. Suppose that for some i and j in $\{0, \ldots, n-1\}$ there is a pendant edge e at distance $in + j + 1$ from the root of T. Then the edge (u_i, v_j) must be present in $E(G)$. Also, for any $j' \in \{0, \ldots, n-1\}$ which is not equal to j, T cannot contain a pendant edge e' at distance $in + j' + 1$ from the root. (To see this, note that by the construction of T_1, edge e' would be a descendant of $x_{i,0}$ in T_1. The presence of e in T ensures that $x_{i,0}$ and $y_{j,0}$ are associated with the same vertex of T but e' is not a descendant of $y_{j,0}$ in T_2.) Similarly, for any $i' \in \{0, \ldots, n-1\}$ which is not equal to i, T cannot contain a pendant edge e' at distance $i'n + j + 1$ from the root. Thus, T contains at most n pendant edges and these correspond to a matching in $E(G)$. So, every rooted tree $(T,r) \in \mathcal{T}(T_1,r_1) \cap \mathcal{T}(T_2,r_2) \cap \mathcal{T}^*$ may be interpreted as a matching in G, and vice versa. This is the sought for bijection between the set of matchings in G and the set $\mathcal{T}(T_1,r_1) \cap \mathcal{T}(T_2,r_2) \cap \mathcal{T}^*$. To conclude, we just need to show how compute the size of the latter set using an oracle for #CommonRootedSubtrees.

Let L be the set of all *leaves* in (T_1,r_1) whose distances from the root r_1 are in the range $[n^2+2, n^2+n+1]$. Let U be the set of all *vertices* in (T_2,r_2) whose distances from r_2 are in the range $[n^2+2, n^2+n+1]$. For each $j \in \{0, \ldots, n\}$, let T_1^j be the tree formed from (T_1,r_1) by adorning every vertex in L with a tuft of $n+j$ edges and let T_2^j be the tree formed from (T_2,r_2) by adorning every vertex in U with a tuft of $n+j$ edges. By the phrase "adorning a vertex v with a tuft of t edges" we mean the following: create t new vertices and add an edge from each of these new vertices to v." For $k \in \{0, \ldots, n\}$, let a_k be the number of rooted trees in $\mathcal{T}(T_1^0, r_1) \cap \mathcal{T}(T_2^0, r_2)$ that have k vertices of degree $n+1$. Clearly,

$$a_n = |\,\mathcal{T}(T_1,r_1) \cap \mathcal{T}(T_2,r_2) \cap \mathcal{T}^*\,|.$$

So we want to show how to compute a_n using an oracle for #CommonRootedSubtrees.

We claim (and shall justify presently) that

$$|\mathcal{T}(T_1^j, r_1) \cap \mathcal{T}(T_2^j, r_2)| = \sum_{k=0}^{n} a_k (j+1)^k. \tag{1}$$

Thus, we can use an oracle for #CommonRootedSubtrees to evaluate the left-hand side of 1 at $j = 0, \ldots n$; then we can compute a_n by Lagrange interpolation.[1]

It remains to prove equation (1). We define a projection function

$$\pi : \mathcal{T}(T_1^j, r_1) \cap \mathcal{T}(T_2^j, r_2) \to \mathcal{T}(T_1^0, r_1) \cap \mathcal{T}(T_2^0, r_2)$$

as follows. For any rooted tree (T, r) in the domain, $(T', r) = \pi(T, r)$ is the maximum r-rooted subtree of (T, r) that has no vertex of degree greater than $n+1$. To see that T' is uniquely defined, consider an embedding of (T, r) into (T_1^j, r_1). The only vertices of degree greater than $n+1$ are those which are mapped to tufts. Thus, (T', r) is obtained from (T, r) by pruning tufts with more than n pendant edges down to exactly n pendant edges. Note also that the resulting tree (T', r) can be embedded in both (T_1^0, r_1) and (T_2^0, r_2), so π is indeed well defined.

How large is $\pi^{-1}(T', r)$? To every tuft with exactly n pendant edges we may add any number of pendant edges, from 0 to j. All the tufts are distinguishable, because they are all at distinct distances from the root r. Thus all these possible augmentations lead to distinct trees, and $\pi^{-1}(T', r) = (j+1)^k$, where k is the number of vertices in (T', r) of degree $n+1$. Thus, each of the a_k rooted trees in $\mathcal{T}(T_1^0, r_1) \cap \mathcal{T}(T_2^0, r_2)$ with k vertices of degree $n+1$ are mapped by π^{-1} to $(j+1)^k$ trees in $\mathcal{T}(T_1^j, r_1) \cap \mathcal{T}(T_2^j, r_2)$. The lemma follows. □

Lemma 3.

$$\#\textsc{CommonRootedSubtrees} \leqslant_{\mathrm{T}} \#\textsc{RootedSubtrees}.$$

Proof. Suppose that (T_1, r_1) and (T_2, r_2) constitute an instance of #CommonRootedSubtrees. Let (T, r) be the rooted tree formed by taking T_1 and T_2 and adding a new root, r, and edges (r, r_1) and (r, r_2). For notational convenience, introduce the following quantities:

$$\begin{aligned}
N_1 &= \#\textsc{RootedSubtrees}(T_1, r_1),\\
N_2 &= \#\textsc{RootedSubtrees}(T_2, r_2),\\
N &= \#\textsc{RootedSubtrees}(T, r), \text{ and}\\
C &= \#\textsc{CommonRootedSubtrees}((T_1, r_1), (T_2, r_2)).
\end{aligned}$$

We start by observing that

$$N = 1 + N_1 + N_2 - C + N_1 N_2 - \binom{C}{2}.$$

To see this, note that (T, r) has

- one distinct subtree in which the degree of r is 0, and
- $N_1 + N_2 - C$ distinct subtrees in which the degree of r is 1, and
- $N_1 N_2 - \binom{C}{2}$ distinct subtrees in which the degree of r is 2.

[1]See Valiant [**11**] for details of this process, particularly the claim that interpolation is a polynomial-time operation.

Thus, $C(C+1) = 2Z$, where Z denotes

$$1 + N_1 + N_2 + N_1 N_2 - N.$$

To compute C, first calculate Z using an oracle for #ROOTEDSUBTREES. Then, observe that

$$C^2 < 2Z < (C+1)^2,$$

so C is the *integer square root* of $2Z$, which can be computed in $\Theta(\log Z)$ time. Note that $\log Z$ is polynomially-bounded in the size of the input, since, for example, $N_1 \leqslant 2^{n_1}$, where n_1 is the number of vertices in T_1. ☐

Lemma 4.

$$\#\text{ROOTEDSUBTREES} \leqslant_{\text{T}} \#\text{SUBTREES}.$$

Proof. For any i, an "i-tuft" is a tree consisting of one (centre) vertex with degree i and i (outer) vertices, each of which has degree 1.

Suppose that (T, r) is an instance of #ROOTEDSUBTREES. Let Δ denote the maximum degree of a vertex in T. Let T' be the tree formed from T by taking a new $(\Delta+3)$-tuft, and identifying one of the outer vertices with r. Let T'' be the tree formed from T by taking a new $(\Delta+2)$-tuft, and identifying one of the outer vertices with r. Let N' denote #SUBTREES(T') and let N'' denote #SUBTREES(T''). Then #ROOTEDSUBTREES(T, r) is equal to $N' - N''$, so it can be computed using an oracle for #SUBTREES. ☐

3. *Some consequences*

Following Valiant [**11**], we say that a function $f : \Sigma^* \to \mathbb{N}$ is in FP if it can be computed by a deterministic polynomial-time Turing machine. We say that it is in FP^g for a problem g if it can be computed by a deterministic polynomial-time Turing machine which is equipped with an oracle for g. Finally, we say that it is in FP^A for a complexity class A if there is some $g \in A$ such that $f \in \mathrm{FP}^g$.

Let #CONNECTEDSUBGRAPHS be the problem of counting unlabelled connected subgraphs of a graph. Formally, let it be defined as follows.

Name. #CONNECTEDSUBGRAPHS
Instance. A graph G.
Output. The number of distinct (up to isomorphism) connected subgraphs of G.

Corollary 5. #CONNECTEDSUBGRAPHS *is complete for* $\mathrm{FP}^{\#\mathrm{P}}$.

Proof. #CONNECTEDSUBGRAPHS is $\mathrm{FP}^{\#\mathrm{P}}$-hard by Theorem 1. We will show that #CONNECTEDSUBGRAPHS is in the class $\mathrm{FP}^{\text{span-P}}$, which will be defined shortly. The result will then follow by Toda's theorem [**9**].

We start by defining the relevant complexity classes. A function $f : \Sigma^* \to \mathbb{N}$ is in the class span-P [**7**] if there is a polynomial-time nondeterministic Turing machine M (with an output device) such that the number of *different* accepting outputs of M on input x is $f(x)$, for all $x \in \Sigma^*$.

A function $f : \Sigma^* \to \mathbb{N}$ is in #NP if there is a polynomial-time nondeterministic Turing machine M and an oracle $A \in \mathrm{NP}$ such that the number of accepting computations of M^A on input x is $f(x)$, for all $x \in \Sigma^*$.

The classes #P, span-P, and #NP are related [**7**] by

$$\#\mathrm{P} \subseteq \text{span-P} \subseteq \#\mathrm{NP}.$$

Thus,

$$\mathrm{FP}^{\#\mathrm{P}} \subseteq \mathrm{FP}^{\text{span-P}} \subseteq \mathrm{FP}^{\#\mathrm{NP}}.$$

But $\mathrm{FP}^{\#\mathrm{NP}} \subseteq \mathrm{FP}^{\#\mathrm{PH}}$, where #PH is the class of functions that count the number of accepting computations of polynomial-time nondeterministic Turing machines with oracles from PH. Furthermore, Toda and Watanabe [**10**] show $\#\mathrm{PH} \subseteq \mathrm{FP}^{\#\mathrm{P}}$. Thus,

$$\mathrm{FP}^{\#\mathrm{P}} = \mathrm{FP}^{\text{span-P}}.$$

(See also Section 1.8 of Welsh's book [**12**].)

We now complete the proof by showing that #CONNECTEDSUBGRAPHS is in $\mathrm{FP}^{\text{span-P}}$. Let $N(G,k)$ denote $k!$ times the number of distinct (up to isomorphism) connected size-k subgraphs of G. Since

$$\#\textsc{ConnectedSubgraphs}(G) = \sum_{k=1}^{n} \frac{1}{k!} N(G,k),$$

where n is the number of vertices of G, it suffices to show that computing $N(G,k)$ is in span-P. Each branch of the computation tree for $N(G,k)$ chooses

- a size-k connected subgraph H of G,
- a bijection σ from the vertices of H to the set $\{v_1,\ldots,v_k\}$, and
- a permutation π of $v_1,\ldots,v_k$.

Let H' be the graph formed from H by relabelling each vertex v of H with the label $\sigma(v)$. If π is an automorphism of H' then (H',π) is output. Otherwise, the branch rejects. The result now follows from Burnside's Lemma, which implies that for any given isomorphism class of k-vertex graphs, the number of graphs in the isomorphism class times the number of automorphisms of any member of the class is equal to $k!$. (For example, see [**5**].) □

Let #GRAPHSUBTREES be the problem of counting unlabelled subtrees of a graph. Formally, let it be defined as follows.

Name. #GRAPHSUBTREES
Instance. A graph G.
Output. The number of distinct (up to isomorphism) subtrees of G.

Corollary 6. #GRAPHSUBTREES *is complete for* $\mathrm{FP}^{\#\mathrm{P}}$.

Proof. This is the same as the proof of Corollary 5, except that the span-P computation rejects any subgraph H which is not a tree. A more direct proof could be obtained by using a polynomial-time canonical labelling algorithm for trees such as the one by Hopcroft and Tarjan [**6**]. □

References

1. G. Blache, M. Karpinski and J. Wirtgen, On approximation intractability of the bandwidth problem, *Electronic Colloquium on Computational Complexity*, Report TR98-014, 1998, http://www.eccc.uni-trier.de/eccc-local/Lists/TR-1998.html.

2. K. J. Edwards and C. J. H. McDiarmid, The complexity of harmonious colouring for trees, *Discrete Applied Mathematics* **57** (1995), 133–144.

3. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

4. M. R. Garey, R. L. Graham, D. S. Johnson and D. E. Knuth, Complexity results for bandwidth minimization, *SIAM Journal on Applied Mathematics* **34** (1978), 477–495.

5. F. Harary and E. M. Palmer, *Graphical Enumeration*, Academic Press, 1973.

6. J. E. Hopcroft and R. E. Tarjan, Efficient planarity testing, *Journal of the ACM* **21** (1974) 549–568.

7. J. Köbler, U. Schöning and J. Toran, On counting and approximation, *Acta Informatica* **26** (1989) 363–379.

8. C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.

9. S. Toda, PP is as hard as the polynomial-time hierarchy, *SIAM Journal on Computing* **20 (1991)**, 865–877.

10. S. Toda and O. Watanabe, Polynomial-time 1-Turing reductions from #PH to #P, *Theoretical Computer Science* **100** (1992) 205–221.

11. L. G. Valiant, The complexity of enumeration and reliability problems, *SIAM Journal on Computing* **8** (1979), 410–421.

12. D. J. A. Welsh, *Complexity: Knots, Colourings and Counting*, Cambridge University Press, 1993.

Leslie Ann Goldberg `leslie@dcs.warwick.ac.uk`
`http://www.dcs.warwick.ac.uk/~leslie/`

Department of Computer Science, University of Warwick,
Coventry, CV4 7AL, UK

Mark Jerrum `mrj@dcs.ed.ac.uk` `http://www.dcs.ed.ac.uk/~mrj/`

Department of Computer Science, University of Edinburgh,
The King's Buildings, Edinburgh EH9 3JZ, UK