WHAT IS AN ANSWER?

HERBERT S. WILF
Department of Mathematics, University of Pennsylvania, Philadelphia, PA 19104

In many branches of pure mathematics it can be surprisingly hard to recognize when a question has, in fact, been answered. A clearcut proof of a theorem or the discovery of a counterexample leaves no doubt in the reader's mind that a solution has been found. But when an "explicit solution" to a problem is given, it may happen that more work is needed to evaluate that "solution," in a particular case, than exhaustively to examine all of the possibilities directly from the original formulation of the problem. In such a situation, other things being equal, we may justifiably question whether the problem has in fact been solved.

Examples of this sort can turn up anywhere, but here we will concentrate on problems in combinatorial mathematics, specifically those of the type "how many—are there?" Such enumeration problems lie at the heart of the subject, and it is important to be able to recognize solutions when they appear. The point, of course, is that sometimes the "answer" is presented as a formula that is so messy and long, and so full of factorials and sign alternations and whatnot, that we may feel that the disease was preferable to the cure.

An answer to such an enumeration question may be given by means of a generating function, a recurrence relation, or by an explicit formula. Each of these is, in essence, just an algorithm for the computation of the counting sequence that is to be determined.

How do we judge the usefulness of such answers? Obviously we might be able to do many things with the answer, such as to make asymptotic estimates, to discover congruence relations, to delight in its elegance, and so forth. We're going to restrict attention here to the appraisal of solutions from the point of view of how easily they allow us to calculate the number of objects in the set that is being studied.

The quality of such formulas should therefore be judged by the usual combination of esthetic and quantitative benchmarks that are used on algorithms. In particular, the quantitative criterion is the computational complexity: the amount of work required to get an answer. We suggest here that the same criterion should be applied to enumeration formulas. We will see that a corollary of this attitude is that our decision as to what constitutes an answer may be time-dependent: as faster algorithms for listing the objects become available, a proposed formula for counting the objects will have to be comparably faster to evaluate.

For concreteness, suppose that for each integer $n > 0$ there is a set S_n that we want to count. Let $f(n) = |S_n|$ (the cardinality of S_n), for each n.

Suppose further that a certain formula has been found, say

$$f(n) = \text{Formula}(n) \qquad (n = 1,2,\ldots) \tag{1}$$

in which Formula(n) may involve various multiple summation signs extending over various sets and various complicated summands, etc.

In order to evaluate the "answer" (1), let's look at the competition. To insure my own immortality in the subject, I am now going to show you a simple formula that "answers" all such questions at once. If you're ready, then, here it is:

$$f(n) = \sum_{S_n} 1. \tag{2}$$

Well, anyway, the summand is elegant, even if the range of summation is a bit untidy.

Now (2) is unacceptable in polite society as an answer, despite its elegant appearance, because it is just a restatement of the question, and it does not give us a tool for calculating $f(n)$ that we

---

In addition to his other accomplishments (see this volume, p. 4) the author is currently one of the associate editors of this MONTHLY.

didn't have before. It does illustrate an important point, though: there is no counting problem for which a formula does not already exist, namely (2).

A first criterion for evaluating an "answer," then, might be that "Formula(n)" should be an improvement over the insightful contribution (2).

What is an improvement? As noted previously, we are considering a formula as an invitation to compute numbers; as an algorithm, if you will, that describes a sequence of steps that will lead to an answer, and it will be useful if less effort is required to use it than to use some other formula or algorithm.

What is effort? Now we're approaching firm ground. The theory of computational complexity has been developed rapidly in recent years and there is no shortage of rigorous standards for measuring computing effort. Usually we adopt some indivisible units of labor, such as multiplications or divisions of numbers in a certain size range, or bit operations, or function evaluations, etc., and then we express the complexity of the computation by counting how many units of each kind of labor need to be performed in order to get the job done.

How do we compare the complexity of evaluating Formula(n) with (2)? Equation (2) calls for the straightforward listing algorithm: produce all of the members of the set S_n and count them. Hence the complexity of (2) is equal to the amount of computational effort that is required to produce all members of S_n.

Next, recall that we need a little more generality in the format of the "answer." Formulas are not always the way solutions are given. Two other useful methods, for example, are the method of generating functions, and the use of recurrence relations. So, in all cases, we will consider the computational complexity of using whatever tool is given for the purpose of computing $f(n)$, whether it be a formula, a recurrence relation, or other.

The functions that will be compared are, therefore:

Count(n) = the complexity of the algorithm for calculating $f(n)$, whether it be given by a formula, an algorithm, et cetera, and

List(n) = the complexity of producing all of the members of the set S_n, one at a time, by the speediest known method, and counting them.

DEFINITION 1: We will say that a solution of a counting problem is *effective* if

$$\lim_{n\to\infty} \frac{\text{Count}(n)}{\text{List}(n)} = 0.$$

What we are saying is that a formula or whatever is an effective solution of a problem if the effort involved in counting the members of S_n by means of that formula is asymptotically small compared to the effort of constructing all of the members of S_n, by the best-known algorithm, and counting them.

EXAMPLE 1: In the theory of numbers it is often said (see [**5**, p. 344]) that there is "no formula for $\pi(n)$ (the number of primes less than or equal to n)." Now, of course, there is such a formula, namely the one given by (2) above. In the light of the definition, though, we can give the precise meaning of the above statement in the form "there is no effective solution to the question of enumerating the primes less than or equal to n."

How good would a formula or other algorithm for $\pi(n)$ have to be in order to qualify as effective? To list all of the primes between 1 and n we can use one of the fast forms of the sieve of Eratosthenes, such as [**4**]. That algorithm operates in time $Kn/\log\log n$. Hence a "formula" for $\pi(n)$ would have to be faster than that in order to satisfy the condition of definition 1.

Again, in complexity theory one speaks of computational problems as being "easy" or "hard" depending on whether the complexity of finding a solution is or is not of polynomial growth in the length of the input bit string. This distinction was first made by Jack Edmonds, and it is responsible for much of the explosive growth in complexity theory today.

To take a leaf from that book, then, a formula "solves" an enumeration problem if the formula yields numbers after "easy" calculations. To make this assumption meaningful we will now restrict attention to those problems in which $f(n)$ grows faster than any polynomial in n. We will

call this the class of $\nu\pi$ (νot πolynomial) problems.

More precisely, then, we propose

DEFINITION 2: We will say that a problem in the class $\nu\pi$ has been p-solved if Count(n) is of polynomial growth as $n \to \infty$.

Respectively, then, we may speak of an enumerative problem as being "unsolved," "effectively solved," or "p-solved."

It is interesting to note that the only input to the computation of $f(n)$ is n itself, and so the length of the input bit string is about log n bits. To ask for polynomial growth in n is therefore to allow exponentially rapid growth according to the customary standards of complexity theory. Nonetheless, this weaker criterion seems quite stern enough, as we will see from the examples.

EXAMPLE 2: For quite a clearcut situation, consider the number of permutations of n letters that have no fixed points. The well-known, and very elegant, solution is that $f(n)$ is equal to the nearest integer to $n!/e$, or equivalently, that

$$f(n) = \sum_{j=0}^{n} (-1)^j \frac{n!}{j!} \qquad (n = 1,2,\ldots). \tag{3}$$

This shows first of all that we are indeed dealing with a problem in the class $\nu\pi$, and second, that this formula really is an answer because we can compute from it in polynomial time, i.e., the problem is p-solved.

EXAMPLE 3: Next, here's an example from graph theory. If we ask for the number of unlabelled graphs on n vertices, we quickly find ourselves using the Frobenius-Burnside theory of group action on a set (e.g., [**1**]). This theory counts the equivalence classes of a set that is acted on by a group of permutations, in terms of the sets of elements that are fixed by the permutations in the group.

Obviously, if a certain permutation has $s(i)$ cycles of length i, then $s(1) + 2s(2) + 3s(3) + \cdots = n$ is a partition of the integer n. The answers that emerge from the Frobenius-Burnside lemma (or from the more general theory of Pólya [**2**]) depend only on the cycles, and so they depend only on the partitions of n. Typically, then, an answer that comes from this theory involves a sum of a more-or-less elementary function extended over all partitions of n.

For instance, the number of nonisomorphic unlabelled graphs on n vertices is exactly ([**2**])

$$f(n) = \frac{1}{n!} \sum_{\pi} \left(\frac{n!}{1^{s(1)}s(1)!2^{s(2)}s(2)!\cdots} \right) 2^{g(\pi)} \tag{4}$$

where

$$g(\pi) = (1/2) \left\{ \sum_{i,j=1}^{n} \gcd(i,j)s(i)s(j) - \sum_{k} s(2k+1) \right\}$$

and π: $n = s(1) + 2s(2) + 3s(3) + \cdots$ runs through the partitions of n.

Evaluation of g, for a given partition of n, is clearly of polynomial complexity. The number of terms in the sum (4) is $p(n)$, the number of partitions of n, and this is well known to grow like $A \exp(B\sqrt{n})/n$. Thus the evaluation of the formula (4) is not a polynomial-time job, since Count(n) grows faster than $A \exp(K\sqrt{n})$ for some constant K.

How big is $f(n)$ itself? The asymptotic behavior of the number of graphs on n vertices is

$$f(n) \sim 2^{\binom{n}{2}}/n! \qquad (n \to \infty).$$

Hence the criterion of Definition 1 is amply satisfied: the answer given by Frobenius-Burnside's lemma is a drastic improvement over (2). This problem is therefore effectively solved. It is not p-solved, however, in the sense of definition 2, because the formula (4) calls for more than a polynomial amount of work.

This raises the interesting question of whether there exists a polynomial-time formula for this

problem. Indeed, can one describe a reasonable and natural family of combinatorial enumeration problems for which there is provably no polynomial-in-n time formula or algorithm to compute $f(n)$? Further, is there any relationship between the intractability of the isomorphism problem for unlabelled graphs and the apparent un-polynomial-ness of the counting problem? These questions are reminiscent of, but not identical to, the concept of $\#P$-completeness discussed in [3].

In addition to the theoretical interest of the questions of computational complexity of formulas, there are practical algorithmic consequences too. Frequently we find that in order to carry out a combinatorial algorithm we need some values of the relevant counting functions. In [6], for example, there is an algorithm that needs to know the number of unlabelled graphs on n vertices before it can begin to do its job. Other algorithms need Bell numbers, and so forth.

EXAMPLE 4: We'll conclude with the example of the partition function itself. Since the sum (4) extended over partitions was hard, what about the sum of 1 over the partitions of n?

That one is easy, because we have any number of simple recurrence relations, generating functions, and so forth, from which the values of $p(n)$ can be rapidly calculated. For instance

$$np(n) = \sum_{k=1}^{n} p(n-k)\sigma(k) \qquad (n \geq 1, p(0) = 1) \tag{5}$$

where $\sigma(k)$ is the sum of the divisors of k. This raises at least the hope that the previous problem may have a polynomial time formula also, and it raises the question of describing the class of functions f of partitions that have the property that the function $g(n)$, obtained by summing f over all partitions of n, can be evaluated in polynomial time.

Added in proof: After reading a preprint of this article, Jeffrey Lagarias and Andrew Odlyzko of Bell Laboratories found an algorithm that computes $\pi(x)$ in time $O(x^{\frac{3}{5}+\epsilon})$: a true "formula" for $\pi(x)$.

References

1. N. G. de Bruijn, Pólya's theory of counting, Chapter 5 in Applied Combinatorial Mathematics, E. F. Beckenbach, editor, Wiley, New York, 1964.

2. G. Pólya, Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen, Acta Math., 68 (1937) 145–254.

3. Michael R. Garey and David S. Johnson, Computers and Intractability, a Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.

4. Paul Pritchard, A sublinear additive sieve for finding prime numbers, Comm. ACM, 24 (1) 1981, pp. 18–23.

5. G. H. Hardy and E. M. Wright, An Introduction to the Theory of Numbers, 3rd ed., Oxford, the Clarendon Press, 1954.

6. John D. Dixon and Herbert S. Wilf, On the random selection of unlabelled graphs, in preparation.

A MATHEMATICAL ANALYSIS OF THE GAME OF JAI ALAI

LOUISE E. MOSER
Department of Mathematics, California State University, Hayward, CA 94542

1. Introduction. Jai Alai is a ball game which evolved during the seventeenth century in the Basque provinces of Spain. It is played in a court with three walls by eight players (singles) or eight teams of two players (doubles). The ball, called the pelota, is thrown against the front playing wall with a curved reed basket, called the cesta. It can bounce off the side wall, the back

Louise Moser received her Ph.D. from the University of Wisconsin in 1970. Her main area of research and publication has been 3-dimensional topology and knot theory. She is currently a professor at California State University, Hayward, where she teaches courses in Mathematics and Computer Science.