Algorithms (fast and slow) and Oracles: a famous problem

Algorithms

Algorithms could deal with almost anything. As Knuth remarks, cookbook recipes are examples of algorithms. I'll discuss math algorithms. Math algorithms could deal with geometry and many other topics but here algorithms will be simple: their inputs will be positive integers and their outputs will be other positive integers. Each number has a size. The size of $345{,}772$ is 6. The size tells how many decimal digits are used in the standard written description of the number. 845 and 120 have the same size, 3.

Work

Our algorithms will involve the usual operations of arithmetic such as addition and multiplication. We'll measure an algorithm's work by estimating the total number of single digit arithmetic operations the algorithm must do. For example, we've seen that SQUARING whose input is a number X with size m can be done by an algorithm taking at most $5m^2$ operations to compute X^2: that's at most m^2 single digit multiplies and at most $4m^2$ single digit adds. Our simple algorithm for the problem SQUARE INQUIRY, which answers "Yes" if the input, X is a square, needs at most $5m^2X$ operations: square all integers from 1 to X (there are X integers, each up to m digits long) and compare the result to X.

Easy and hard

The work done by an algorithm may depend on its input. An algorithm is called POLYNOMIAL TIME if the work is at most a multiple of some power of the input's size. So SQUARING is polynomial time, but SQUARE INQUIRY as described above is not. A now-classical source (*The Design and Analysis of Computer Algorithms* by Aho, Hopcroft, and Uhlman, Addison Wesley, 1974) declares (p. 364):

> How much computation should a problem require before we rate the problem as being truly difficult? There is general agreement that if a problem cannot be solved in less than exponential time, then the problem should be considered completely intractable. The implication of this "rating scheme" is that problems having polynomial-time-bounded algorithms are tractable. . . . we say that a problem is *intractable* if all algorithms to solve that problem are of at least exponential time complexity.

The algorithm for SQUARE INQUIRY described above is exponential because if X has m digits, then X is about 10^m in size (that's what it means to have m digits) so that $5m^2X$ is $5m^2 10^m$, and this really will grow very very **very** rapidly, faster than any polynomial, as m gets large.

Oracles

One dictionary entry for oracle is "a person or thing regarded as an infallible guide to future action" and here I'll try to explain how an oracle could help with an algorithm. Imagine that a very wise and trustworthy being always knows the right answer to any question an algorithm proposes to compute. For example, our oracle for SQUARING responds instantaneously with 1600 when told 40, and our oracle for SQUARE INQUIRY responds immediately "Yes" when asked "Is 801025 a square?"

We could wonder how these oracles work, but we assume they are always correct and always truthful. Someone who is not a true believer, however, might want *evidence* that the oracle's answer is correct. The evidence the SQUARE ORACLE could present is just the multiplication of 40 by itself, a rather limited computation. The SQUARE INQUIRY ORACLE could give very convincing evidence by actually supplying a number, 895*, which we could then use to verify that 801025 is a square.

In both cases the oracle could supply enough evidence so that mere mortals can verify the oracle's correctness with a polynomial amount of computation. Such oracles can be checked in polynomial time.

The question

Suppose that a problem has answers which can always be checked in polynomial time. More poetically, suppose there's an oracle for a problem which can always supply evidence supporting its assertion which can be verified in polynomial time. The central question of complexity theory is:

* Calculator help was accepted here!

IF AN ORACLE CAN BE CHECKED IN POLYNOMIAL TIME, MUST THERE ALWAYS BE SOME ALGORITHM WHICH SOLVES THE PROBLEM DIRECTLY IN POLYNOMIAL TIME?

That is, do we need an oracle's help? This statement is a major part of what is called the P=NP PROBLEM. Many people have worked on this problem over the last 10 or 15 years. They haven't solved it.

Relevance to cryptography

Encrypting and decrypting should not be too difficult, otherwise people won't use our methods. So at worst, the descriptions of these processes should be polynomial. Giving the recipient of a message the key for the decryption process is logically the same as an oracle telling us a solution to an inquiry, and decryption can be viewed as our polynomial time verification of the oracle's response. The assertion that there is no polynomial time method of "breaking" our encryption scheme means that for all practical purposes communication is secure**. We actually hope that the cryptosystem can only be "solved" by an exponential time algorithm.

One famous NP problem which has polynomial time oracles is factoring integers – that is, writing a given integer as a product of two or more other integers greater than 1. Here's a version of the factoring problem which we'll see lies at the foundation of the security of many current cryptosystems:

INPUT A number which is known to be the product of two primes.
OUTPUT The two primes.

So one input could be "15", and the corresponding output would be "3 and 5". On the other hand, if the input were a rather large (300 digit?) product of two primes, then finding the factors might be difficult, even with lots and lots of cleverly used computer resources. Yet if the two primes are given, checking that their product is what's specified is not very hard because multiplication can be done in polynomial time.

Appendix SQUARE INQUIRY **done in polynomial time**

I'll describe another way to do SQUARE INQUIRY so you won't think I've solved a famous problem by displaying a question which is NP but not P***. That is, I've shown you that SQUARE INQUIRY has an oracle whose answers can be supported by evidence in polynomial time. Now I'll show you an ugly but polynomial way to do square inquiry. Please note that I used a calculator here, also. I'll describe the process first with an example: is the number 8765432 (a number of size 7) the square of another number?

Step 1 Square these numbers: **0**999999, **1**999999, ... , **8**999999, **9**999999. These are each 7 digit numbers so their squaring takes at most $5 \cdot 7^2$ operations, and there are 10 of them, so I've done $10 \cdot 5 \cdot 7^2$ operations. I compare the results to my original number, and deduce that 8765432 is less than the square of **0**999999.

Step 2 Square these numbers: 0**0**99999, 0**1**99999, 0**2**99999, ... , 0**9**99999. These are each 7 digit numbers so their squaring takes at most $5 \cdot 7^2$ operations, and there are 10 of them, so I've done $10 \cdot 5 \cdot 7^2$ operations. I compare the results to my original number, and deduce that 8765432 is less than the square of 0**0**99999.

Step 3 does the same, concentrating at the third digit, reading from the right. It uses again at most $10 \cdot 5 \cdot 7^2$ operations. We learn that the square of 00**0**9999 is also larger than 8765432.

Step 4 uses the same amount of work, but the outcome is quite different: we learn that 8765432 must lie between the squares of 000**1**999 and 000**2**999 (wow!).

Step 5 considers the numbers 0002**0**99, 0002**1**99, ... 0002**9**99, and with the same amount of work tells us that 8765432 must lie between the squares of 0002**8**99 and 0002**9**99.

We do two more **Steps** (each with the same amount of work), and learn that 8765432 is between the square of 2960 and 2961, but is equal to neither one of them. So the answer to SQUARE INQUIRY is "No".

This procedure answers SQUARE INQUIRY. It uses the squares of 10 numbers for each decimal place in the original number. So we compute 10 squares with work $5m^2$ for every digit of a number whose size is m. The total work is the product of these three terms, which is $40m^3$, certainly a polynomial. The implementation is certainly unappealing, but our work estimate is valid. Therefore SQUARE INQUIRY has a polynomial time algorithm.

** Yes, everyone can get lucky, but luck isn't really too reliable if the chance of being lucky is rather small. Such luck is a much less likely way of getting the message than simply bribing the message creator!

*** I wish that I knew one!