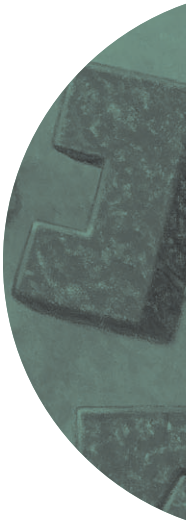
SOCIETY OF ACTUARIES

Article from:

CompAct

May 2004 – Issue 17



P versus NP, or How long will this take?

by Carol Marler

For a new and creative solution to this question, you could win a prize of $1,000.000.

On May 24, 2000, the Clay Mathematics Institute offered prizes of $1 million each for seven unsolved problems in mathematics. Landon Clay (whose major at Harvard was not in mathematics, but in English) wanted to get more media attention for the field of mathematics. His personal fortune was used to found the Clay Mathematics Institute (www.claymath.org).

The idea for these "millennium problems" came from an address given by mathematician David Hilbert in 1900, in which he listed the most significant mathematics problems at that time. One problem on his list was Fermat's Last Theorem, which was solved recently.

Although the solution to Fermat's Last Theorem made extensive use of computers, the theorem itself does not refer in any way to computers. Very few problems in theoretical math, even today, fall under the heading of computer science. Most computer science questions are practical, not theoretical. But looking back in history, some theory was developed even before any modern computers were built.

The theoretical basis of computer science was developed by a mathematician named A.M. Turing. His model of a computer was a black box of sorts with n different internal states, and an input "tape" consisting of zeros and ones. The operation of the black box was defined by a transition matrix in which the current state and the current value on the input tape determined the next state.

The amazing thing about this model is that any real-world computer program could be mimicked with a suitable set of states and a transition matrix. The theoreticians can then analyze whole families of programs. The P versus NP millennium problem considers a grouping of programs based on a formula for how long (at most) it takes to process an input string of length n.

Programs whose time formula is a polynomial in n are denoted as P, for polynomial time. For example, consider a program that sorts N values in ascending order. A simple way to do this is to loop through the list, comparing each pair of numbers (N-1 comparisons), and switching them if the second number is smaller than the first. This loop would then be repeated N-1 times, until all the numbers had been sifted through to their proper position in the sorted list. If each compare and switching took k units of time, the formula is k*(N-1)*(N-1) plus some small amount of time to set things up, etc. The time required is clearly a second degree polynomial. Improvements are possible, such as noting that each iteration of the outer loop reduces by one the number of comparisons required, since the last i+1 entries are now determined.

Incidentally, changing a list of n zeros and ones into N values is also known to be a polynomial process. But some processes simply cannot be done in polynomial time. For example, consider the "traveling salesman" problem. The salesman (or woman, but let's use male pronouns for simplicity) has a list of N destinations that he wants to visit. There is an N*N matrix of the distances from each destination to any other. The salesman wants to minimize

Carol Marler is a former editor of the Speculative Fiction contest. She is currently unaffiliated. She can be reached at 704-948-0545.

time on the road, and this can be programmed by listing all possible routes, calculating the length of each route, and then taking the shortest. If there are only five or so cities, this can even be accomplished by hand. But it is apparent that as N increases, the program will require N! distance calculations. And N! for large N grows exponentially. You don't want to kick off a calculation for 50 destinations if you expect the answer in a reasonable length of time.

The next step in analyzing this kind of problem is to generalize the computer model. One very useful variation, and one that is rather actuarial in nature, is to replace the deterministic computer with a non-deterministic one. Instead of defining the resulting state, the transition matrix produces a probability distribution of states. Nobody has yet built a non-deterministic computer, but if we had one, the traveling salesman problem could be solved on it in polynomial time.

In fact, the traveling salesman problem is just one of a whole family of problems for which the individual calculations are simple enough, but the number of possibilities to be tested grow exponentially. These problems are denoted as NP-complete.

In 1971, Stephen Cook proved that if any NP-complete problem could be solved on a deterministic computer in polynomial time, then all of them could be similarly solved. The millennium problem then asks whether these NP-complete problems are a subset of P, meaning that they could be solved deterministically with the right kind of algorithm, or are they inherently not solvable in that way?

In the real world, the Cook analysis leaves just two possibilities. First, if a problem is known to be NP-complete, and a software developer doesn't want to waste time looking for a solution that has already been sought so many times without success, the problem can be discarded and the software can be built for some other problem instead. A great many problems of importance to business and industry have been dropped for this reason. Some other problems of less practical value also fall into this category—you may want to Google "NP-complete" just to see how many times the minesweeper game pops up. Second, if some genius inventor in a garage can find the answer to any particular NP-complete problem, he will soon be richer than Bill Gates (even without the millennium prize).

However, if you think it is important to have the ability to do secure online transactions, you had better hope that the genius is not successful, since the most commonly used security method is based on a known NP-complete process—factoring a very large number into its prime factors. If NP-complete turns out to be a subset of P, the methodology will no longer be secure.

If, in fact, NP-complete is not a subset of P, the millennium prize will go to the person who can prove that fact. A great many attempts have been made to do this, and none have yet been successful. It could be another century before this problem gets solved. Some even theorize that it is true but cannot be proven.

One intermediate possibility also exists. Even if NP-complete is a subset of P, the power of the required polynomial may prove to be so large that no practical solution is available. A 50-degree polynomial is not much of an improvement on an exponential formula if the traveling salesman has no more than 50 destinations.

Do you and your PC have the insight and CPU cycles to attack the problem? If so, see http://www.claymath.org/millennium/ to get entry information.

Nobody has yet built a non-deterministic computer, but if we had one, the traveling salesman problem could be solved on it in polynomial time.