+plus magazine ...living mathematics

March 2003
Features

How maths can make you rich and famous

by Chris Budd

BOOKMARK

There are many good reasons for doing mathematics. There is the sheer joy of mathematical discovery, the beautiful interplay between mathematical structures and patterns, the fascinating way that mathematics helps to reveal the mysteries of the physical world and the excitement of solving age−old puzzles. Becoming rich and famous is not usually a motivation for doing maths, and is an endeavour more usually linked to film stars and sports personalities. However, in this article I will tell you how you can become rich and famous by doing maths − but be warned, this is not an short cut to riches, and becoming a film star may be easier.

How maths *can't* make you rich and famous

Sometimes maths is no use at all...!
Image DHD Photo Gallery

There are various ways that mathematics has been claimed to be able to make you rich and famous and I thought that it would be good to dispose of some of these as quickly as possible. One of the questions that I am frequently asked when people find out that I'm a professional mathematician (apart from "when will you leave?") is "how can I choose my numbers for the National Lottery to maximise my chances of winning?". Unfortunately mathematics is of no great use here as all combinations of numbers are equally likely (or unlikely) to win (a fact not well appreciated by the general public).

The same conclusion, by and large, applies to any form of gambling. The invention of the branch of mathematics now called probability was motivated, in part, by attempts to understand gambling. However, in the long term, all gambling will lead to a loss of hard−earned cash (as the people who own casinos understand probability as well and can adjust the various games so that the odds are in their favour).

A final way not to become rich and famous with mathematics is to win a Nobel prize.There is no Nobel prize in mathematics. The reason for this is that Alfred Nobel, when setting up the prizes (for physics, chemistry, economics, peace, etc.) did not include mathematics because he, quite misguidedly, thought that mathematics had no practical use.

How mathematics *could* make you rich and famous

Faced with the depressing conclusions of the previous section we have to ask ourselves whether mathematics can ever make us rich and famous. Fortunately (for the reader) the answer to this question is **yes**. One way is to work for a bank. There is a new branch of mathematics called *financial mathematics* which looks at questions like the right price to set for an option (a right to buy shares on the stock market). Mathematicians able to work in this field are in high demand, but have to understand the complexities of tricky subjects such as stochastic differential equations.

Maths is the key...
Image www.freeimages.co.uk

Another way to become rich and famous is to devise, or break a code. Nearly all modern codes are based upon ideas from pure mathematics (number theory to be precise). Codes are incredibly important to nearly all aspects of modern life, including transactions between you and your bank (if you have a bank account) and any time that you send "secret" information (such as a credit card number) over the Internet. Most of them rely for their security on the current belief that it is very hard to find the prime factors of a large number. (Later on we shall find out exactly how hard this is.) If you can find a quick way to factorise a large number quickly, then fame and fortune (possibly obtained illegally) are yours for the taking. (It may however be easier to rob a bank direct, although there is a possibility that a quantum computer will do it for us.)

The final way that maths could make you rich and famous is the main subject of this article. You can attempt to solve one of the prize problems put together by the *Clay Institute*. A prize of $1,000,000 is out there for any one of you who can solve one of these problems.

$1,000,000 to be won!!!!!!!

Be warned, however, the problems are **hard** and it may be easier to rob a bank after all.

The Clay Institute

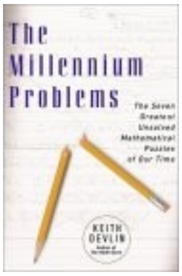
As you will probably have noticed, we are now not only at the start of a new century, but also at the start of a new millennium. It was felt appropriate by many mathematicians the 21st century should start off with a series of unsolved problems in mathematics. There were several groups that proposed different lists of problems ranging from the very applied to the very pure. It is not difficult to pose a problem in mathematics which is hard to solve. Indeed most nonlinear differential equations have no known exact solution, nor can we see how any such equation ever could have an exact solution. Similar technical problems arise in all fields of mathematics.

However, these are not in a sense good problems. A *good problem* is one which is not just hard in its own right, but the solution of which, or even the attempts at a solution of which, will generate a lot of new mathematics in the process and which will lead to the solution of problems in areas of mathematics far distant from those in which the problem was originally posed. This is a much harder test of a good problem.

In the end, out of the vast range of possible problems, a list of seven *Millennium Prize Problems* problems emerged. These were posed by an organisation called the Clay Institute which is based in Cambridge, Massachusetts. As I said in the introduction $1,000,000 is out there to be won. To win this prize you must either solve a problem or produce a counterexample and (here's the catch) your solution must stand the rigours of inspection by all of the world's mathematicians for at least two years!

The full list of problems is as follows:

1. P versus NP;
2. The Hodge Conjecture;
3. The Poincarë Conjecture.
4. The Riemann Hypothesis;
5. Yang−Mills existence and Mass gap;
6. The well−posedness of the Navier−Stokes equations;
7. The Birch and Swinnerton−Dyer Conjecture.


The Millennium Problems
The Seven Greatest Unsolved Mathematical Puzzles of Our Time
KEITH DEVLIN

There − do you feel challenged? If you want more information about any individual problem, full details are given on the Clay institute website, where you will also find the rules of the competition. There is also an excellent book by Keith Devlin, "The Millennium Problems", which goes into detail not only on each problem but also on the various teams of mathematicians around the world who are trying to solve them.

Roughly speaking, problems 4 and 7 are in number theory, 2 and 3 are in topology, 1 is in optimisation and 5 and 6 are problems about differential equations. It is dangerous to say that any one mathematical problem is more applied than any other (and even more dangerous to say which is more important), however I think it is fair to say that the solution of problems 1 and 7 will have immediate practical importance, whilst the application of the others lies further in the future.

We will now look in more detail at Problem 1.

Problem 1: P versus NP

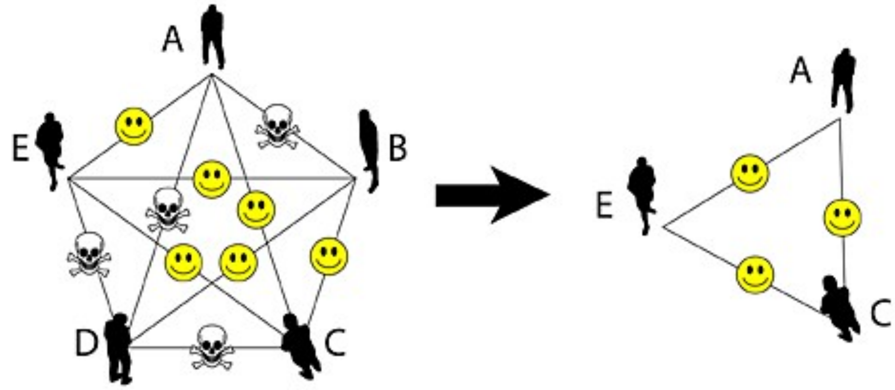
Please forgive the rather dry title to this section as the problems that it lead to are fascinating, relevant, important and (in keeping with the title of this article) potentially the source of great riches. The P versus NP problem is all about *how easy it is to solve problems*, so it is a problem about problems. Put another way, this problem relates to the question of whether a computer can ever replace a mathematician, ie. will I (and perhaps you!) have a job in a few years time.

The party problem

To motivate this question we will look a little ahead, into the future of those readers who are still at school, and endeavour to give a bit of advice to those going up to university. When you go to university certain rather unimportant things will occupy your time, such as the need to study, feed yourself and look for a job. These are all distractions in the pursuit of the most important student activity, namely that of going to parties.

Imagine now that you are in your first week at university, and to celebrate having survived for a whole week you decide to throw a (wild) party. In this first week you have made a grand total of *five* friends, named (conveniently for a mathematician) Angela, Brian, Colin, Daphne and Edward. Sadly you live in rather cramped university accommodation, and your room can only hold a (fun−filled) party of three. The first week at university has already been quite eventful. A and B have gone out, have split up and B is now going out with D. As a consequence A and will not come to the party if B or D are there. C used to be going out with D and will not go to the party if she is there. D beat E in the first week test, and E will also not go if D is there.

So, the question that you are faced with is *which three friends can you invite to the party so that all are compatible with each other?* In this case it doesn't take very long to see that you can safely invite Angela, Colin and Edward to have an ACE party.



Party time − but who is coming?

Now let's move a few years on to the end of your final year at university. You plan now to have a (mega) party to celebrate your departure (complete with refreshments and a band). You have been very socially active during your university career and now have 200 friends (who said that maths wasn't fun?). You are also a bit more organised and decide to hire the Student Union building to hold your party in. This metropolis will hold 100 people. Sadly, however, the love lives of your friends have been varied and complex and you have a long list which tells you which of your friends cannot possibly come to the party together (at least if the union building is to survive the experience).

The party planner
Image freeimages.co.uk

The question is *who do you invite?* Fortunately your university mathematics course has included a module on computing and you decide to try to solve this problem on a computer. At this stage I invite (educated) guesses from the gentle reader as to how long it will take your computer to come up with the party list. Did I hear a minute from the reader on the right? Who was that who said 10 seconds? Would anyone be prepared to wait for an hour?

Solving the party problem

Let's now look at a simple strategy with two parts for solving this problem.

> *Part A*: Get the computer to select a *random* set of 100 people from your list of 200 friends.
>
> *Part B*: Check this selection against your list to see if there are any incompatible friends. If the selection is OK then you have a party. If not, go back to Part A.

We now have to ask how long it will take our computer to implement this strategy. Suppose that we have a possible party chosen in Part A. To check this party we need to go through the list in Part B. This operation takes a time proportional to the number of possible pairings of friends at the party. You go through each possible pair and see whether it is on the list.Your possible party has 100 people and there are a possible $100 \times 99/2 = 4950$ items to check. Thus Part B will take 4950 times the time it takes the computer to do one operation. On a fast computer each operation might be around 1ns (a nanosecond is 10^{-9} seconds), so Part B takes 4590ns=4.59μs (a microsecond is 10^{-6} seconds); ie. it is almost instantaneous.

Now letâ€™s look at the time it takes to do Part A. To implement the algorithm, the computer has to check out every possible party. Given that there are 200 possible people and 100 need to be chosen, the number of possible parties is given by ${}^{200}C_{100}$, where

$$ {}^{n}C_r = \frac{n!}{r!\,(n-r)!} \quad \text{and} \quad n! = n(n-1)(n-2)\ldots 2 \times 1. $$

Using this formula with $n = 200$ and $r = 100$ we find that the total number of possible parties is

$$ \frac{200!}{100! \times 100!} = 9.0548 \times 10^{58}. $$

This is the total number of parties, and each takes 4.5μs to check out. So, in total the computer will take

$$ 4 \times 10^{53} \quad \text{or} \quad 1.32 \times 10^{46}\text{years} $$

to find a possible party. To put you in the picture, the estimated life time of the universe is about 15 billion (15$\times 10^9$) years.

So what are we to do? Well, one solution is to *deep−freeze* the computer programmer until the computer comes up with the answer. Of course by this time your friends may have got a bit bored with waiting. The alternative strategy is to find a better way of programming the computer (or, in maths speak, to find a better algorithm). However, it is very far from obvious what sort of algorithm this should be.

Let's have a closer look at the difficulty of this problem. Suppose that we have n friends and we want to invite half of them to the party, so that $r = n/2$ or $n = 2r$. The number of possible pairs of enemies among your friends cannot be more than the total possible number of pairs which is ${}^nC_2 = n(n-1)/2.$ This is the maximum total length of the list of pairs of enemies. Also, in your list of r attendees at the party there are $r(r-1)/2$ different pairs. To check your party you must check each one of these pairs against the master list, which takes $r(r-1)/2$ operations. Thus the time it takes to *check* a party is (at worst) proportional to n^2.

We say that the time of this checking process is *polynomial in n*, or *takes polynomial time*. Polynomial time means that the checking time is always bounded by some power of *n*.

Anything which can be done in polynomial time is (essentially) doable on a computer. If a problem can be *solved* in polynomial time then we say it is of type **P**, and is in a sense *easy*. An example of an easy problem is that of sorting a list of n numbers into increasing order. Although the total number of different ways of arranging n numbers is $n!$, you can put them into order in a time proportional to $n\ln(n)$ (which is even smaller than n^2). Rapid algorithms for sorting lists lie at the heart of data−bases.

Is the party problem an easy problem?

Letâ€™s look at how many different parties there are which half of your friends can attend. A good approximation to ${}^{2r}C_r$ can be found by using *Stirlingâ€™s formula* for $n!$ and is given by

$${}^{2r}C_r \text{ approximately equals } \frac{4^r}{\sqrt{\pi r}}.$$

This formula allows us to quickly estimate the size of task A. For even moderate values of r this number is *huge*. It is far, far bigger than r^2. For comparison, suppose that we want to host a moderately sized party. Here is a comparison of r^2 with ${}^{2r}C_r$.

r	**r^2**	**${}^{2r}C_r$**
1	1	2
2	4	6
3	9	20
4	16	70
5	25	252
6	36	924

The number of parties grows faster than *any polynomial function of r*. Indeed it is an *exponential function of r*. The time that it takes to check out this number of parties that we need to check is *not* polynomial in *r*, but grows exponentially with *r*. We call such problems **N**. **N** problems are *hard* to solve.

The party problem is a bit special. Although it takes **N**−time to generate all of the parties, it only takes **P**−time to check each one. This type of problem is called **NP** (hence the name of the problem in this section).

The procedure that we have identified for solving the party problem takes far more than polynomial time. Thus the problem appears to be *hard*. Of course, we may not have come up with an especially efficient algorithm for solving the party problem. For example, one way to sort a list of length *n* is to try every one of the *n!* possible lists and to see which one is in order. This method is very slow indeed, much slower than the *n*ln(*n*) operations that the quick−sort algorithm takes.

How long will it take to decide?

The million dollar question (quite literally) is whether we are forced to always use a slow algorithm for the party problem, or whether we can find a perfect party using a computer algorithm which takes *polynomial time*. If this is the case then we could say (for the party problem) that

P = NP.

The problem is that noone has ever found such an algorithm nor has it has even been proven for certain that such an algorithm cannot exist.

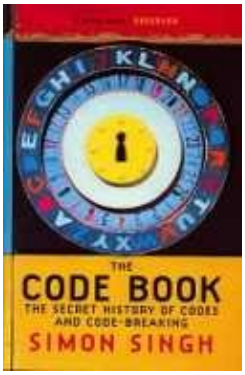
The immensity of our task is this. To show that a problem is of type **P** you must find a polynomial time algorithm, however, to show that it is not−**P** you must check all possible algorithms and show that none of them work in polynomial time.

Why do we care about P versus NP?

If this were just a question about parties, it wouldn't have made it into anyone's list of top problems for the new millennium. However, there are many, many problems which are very like the party problem, in that they take **P** operations to check, but there are **N** possibilities to go through. We call all such problems **NP** problems. The remarkable thing is that it has been proven that there is a large set of **NP** problems which have the property that if you can solve *one* in polynomial time then you can solve *all* such problems in polynomial time. This is called the set of **NP complete** problems. Crack one and you crack them all. It is the all−encompassing nature of this result which makes the **P** versus **NP** question such an important problem.

Here are some of the variety of problems which are **NP**:

- The *knapsack problem*: fitting a set of oddly shaped items into a knapsack (closely related to the question of placing components on a circuit board).
- The *time−tabling problem* : constructing a working timetable for a school so that teachers and students are never in two places at once (a similar problem arises in placing pilots in an airline).
- Solving the minesweeper game that you have on your computer.
- Colouring a map with four colours, so that no two adjacent regions have the same colour.
- Factorising large numbers.



As we hinted earlier, the last of these problems lies at the heart of modern cryptography for which the security of a code relies on the difficulty of finding the factors of a number. You can find out why in *The code book* by Simon Singh, and there is more about cryptography in Safety in numbers from Issue 21 of *Plus*.

Briefly, modern codes (based on the RSA cipher) start with a pair of large prime numbers p_1 and p_2 and multiply them together to give a product $m=p_1p_2$. The number m is released to the public, but p_1 and p_2 are kept secret. To crack the code you have to find p_1 and p_2, given the value of m. Now m is usually a very large number, maybe m has n decimal digits where n may be around 100. To find the factors of m one simple technique would be to check each of the numbers less than the square root of m to see whether it divides into m. Given any number, checking that it divides into m is easy and takes a number of operations proportional to at most n^2 (you can check this for yourself). So seeing if a number is a factor can be determined in a time which is a polynomial function of N.

It is also easy to see that the square root of m must have a value somewhere between $10^{(n-1)/2}$ and $10^{n/2}$. However, this means that we must check about $10^{n/2}$ different numbers to find the factors of m. As in the party problem, this number does not grow polynomially with n. For example, if n=100 we must check about 10^{50} different possible factors. Like our party problem, this is an enormous number and no computer can check all of these numbers within the lifetime of the computer programmer. It is this fact which makes modern codes so secure.

It should be said at this stage − especially if this article is read several years into the future when all of these issues may resolved − that there is a chance that *quantum computers* might be able to perform this feat, however a working quantum computer has yet to be built! Thus, at this stage the **P** versus **NP** problem is wide open, although most mathematicians suspect that (when using a conventional computer) **P** is not equal to **NP**.

If this is so, our party organiser will simply have to wait!

In conclusion

Rich − and who cares about famous?
Image freeimages.co.uk

I hope that this brief overview has given you some insight into the sort of problems that mathematicians are interested in and why some of them are important. In the next issue of *Plus* I will look in more detail at another of the Millennium Prize Problems − the well−posedness of the Navier−Stokes equations. It is certainly true that solving one of these problems would give you a form of mathematical immortality and maybe even fame and fortune. My advice to any of you who think that you might have a crack at one of the problems is *go for it*!

However, it is also worth saying that maths can make you rich and famous in many other ways as it unlocks the doors to a huge number of interesting and varied careers. But, I still wouldn't advise you to rob a bank.

About the author

Chris Budd is Professor of Applied Mathematics at the University of Bath, and Chair of Mathematics for the Royal Institution. He is particularly interested in applying mathematics to the real world and promoting the public understanding of mathematics.

He has recently co−written the popular mathematics book *Mathematics Galore!*, published by Oxford University Press, with C. Sangwin.

Plus is part of the family of activities in the Millennium Mathematics Project, which also includes the NRICH and MOTIVATE sites.