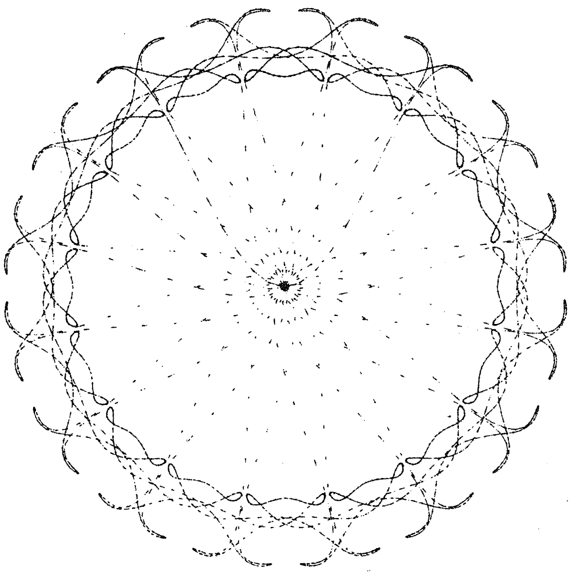
ISSN 0313 - 6825

FUNCTION

Volume 5 Part 2 April 1981



A SCHOOL MATHEMATICS MAGAZINE

Published by Monash University

Function is a mathematics magazine addressed principally to students in the upper forms of schools. Today mathematics is used in most of the sciences, physical, biological and social, in business management, in engineering. There are few human endeavours, from weather prediction to siting of traffic lights, that do not involve mathematics. *Function* contains articles describing some of these uses of mathematics. It also has articles, for entertainment and instruction, about mathematics and its history. Each issue contains problems and solutions are invited.

It is hoped that the student readers of *Function* will contribute material for publication. Articles, ideas, cartoons, comments, criticisms, advice are earnestly sought. Please send to the editors your views about what can be done to make *Function* more interesting for you.

∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞

EDITORS: M.A.B. Deakin (chairman), N. Cameron, G.B. Preston, R. Sacks-Davis, G.A. Watterson (all at Monash University); N.S. Barnett (Footscray Institute of Technology); K.McR. Evans (Scotch College); D.A. Holton (University of Melbourne); P.E. Kloeden (Murdoch University); J.M. Mack (University of Sydney); E.A. Sonenberg (R.A.A.F. Academy); N.H. Williams (University of Queensland).

BUSINESS MANAGER: Joan Williams (Tel. No. (03) 541 0811, Ext.2548)

ART WORK: Jean Sheldon

Articles, correspondence, problems (with or without solutions) and other material for publication are invited. Address them to:

The Editors,
Function,
Department of Mathematics,
Monash University,
Clayton, Victoria, 3168.

Alternatively correspondence may be addressed individually to any of the editors at the addresses shown above.

The magazine will be published five times a year in February, April, June, August, October. Price for five issues (including postage): $4.50; single issues $1.00. Payments should be sent to the business manager at the above address: cheques and money orders should be made payable to Monash University. Enquiries about advertising should be directed to the business manager.

∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞

Registered for posting as a periodical - "Category B"

This is a somewhat unusual issue of *Function* in that much of it is devoted to a single topic. Rodney Topor's article on Computer Chess is longer and more specialized than most of those we publish, and in places, it is rather difficult. However, we have received more requests for articles in this area than in any other (much to our own surprise). This article is so thorough and authoritative that we were loathe to cut it. We are most grateful for the help given to us in editing it by Mr Robert Jamieson, International Master and current Australian Open Champion. Mr Jamieson edits his own journal *Chess Players' Quarterly* (P.O. Box 119, Ringwood, Victoria, 3134, Subscription $5 p.a.) and had in fact published a summary of Dr Topor's article. The topic of computer chess is related to the important field of "artificial intelligence", or AI as it is called. This link is briefly mentioned on p.22.

CONTENTS

THE FRONT COVER

N.S. Barnett,
Footscray Institute of Technology

The design of the front cover was drawn using a Tektronix 4051 microprocessor incorporating special graphics facilities. The microprocessor operates on the 'BASIC' language, for those who have already done a little programming, with the special graphics commands - MOVE, DRAW, ROTATE, RMOVE and RDRAW. Given two points with coordinates (x_1,y_1) and (x_2,y_2) we can, of course, join them by a single straight line. In the diagram on p.3, a set of points has been chosen and consecutive ones joined by a straight line. The curved effect is obtained by choosing points sufficiently close to one another. The form of the resulting drawing is determined by the manner in which the set of points has been prescribed.

For the cover design the set of points is given by (x,y) where

$$\begin{aligned} x = {} & [\text{Cos}(T + \cdot 05) + \text{Cos}(2(T + \cdot 05))] \,.\, \text{Cos}(T + \cdot 05) \\ & \qquad .\, \text{Cos}(2(T + \cdot 05)) \\ & \qquad .\, \text{Cos}(3(T + \cdot 05)) \\ & -[\text{Cos}\, T + \text{Cos}\, 2T].\text{Cos}\, T.\text{Cos}\, 2T.\text{Cos}\, 3T \end{aligned}$$

and

$$y = [\text{Sin}(T + \cdot 05) + \text{Sin}\, 2(T + \cdot 05)] - [\text{Sin}\, T + \text{Sin}\, 2T]$$

where T takes the 128 values,

$$\frac{\pi}{64}, \frac{2\pi}{64}, \ldots, 2\pi.$$

Joining these 128 points consecutively produces the diagram on p.3.

Using the rotation command the whole procedure is repeated, rotated anti-clockwise through $\frac{\pi}{8}$ radians and rotation continued through units of $\frac{\pi}{8}$ radians until the original figure is obtained. The final result is the diagram on the front cover.

Both program I and program II (on p.3) accomplish the same result; however, program I has a total execution time of 30 minutes and program II an execution time of $3\frac{3}{4}$ minutes. You might like to discuss amongst yourselves and with your teacher why this should be. The WINDOW and VIEWPOINT commands fix the size and location of the drawing on the screen; they are not essential to the understanding of the program.

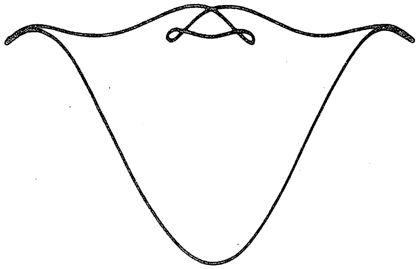
The whole aim of this exercise was to produce a 'pretty picture' and there are numerous books on the market displaying many computer drawn designs. Computer graphics, however, have many practical applications and it is hoped to cover some of these in a future article.

PROGRAM I

```
100  WINDOW -0·7,4·7,-2·7,2·7
101  VIEWPORT 15,115,0,100
110  FOR I=0 TO 15*PI/8 STEP PI/8
120  ROTATE I
130  MOVE 2,0
140  FOR T=0 TO 127*PI/64 STEP PI/64
150  S=T+0·05
160  X1 = (COS(T)+COS(2*T))*COS(T)*COS(2*T)*COS(3*T)
170  X2 = (COS(S)+COS(2*S))*COS(S)*COS(2*S)*COS(3*S)
180  Y1 = SIN(T)+SIN(2*T)
190  Y2 = SIN(S)+SIN(2*S)
200  RDRAW X2-X1,Y2-Y1
210  NEXT T
220  NEXT I
```

PROGRAM II

```
 90  DIM A(128),B(128),C(128),D(128)
100  WINDOW -0·7,4·7,-2·7,2·7
110  VIEWPORT 15,115,0,100
150  N=0
160  FOR T=0 TO 127*PI/64 STEP PI/64
170  N=N+1
180  S=T+0·05
190  A(N)=(COS(T)+COS(2*T))*COS(T)*COS(2*T)*COS(3*T)
200  B(N)=(COS(S)+COS(2*S))*COS(S)*COS(2*S)*COS(3*S)
210  C(N)=SIN(T)+SIN(2*T)
220  D(N)=SIN(S)+SIN(2*S)
230  NEXT T
231  FOR I=0 TO 15*PI/8 STEP PI/8
232  ROTATE I
233  MOVE 2,0
240  FOR N=1 TO 128
250  RDRAW A(N)-B(N),C(N)-D(N)
260  NEXT N
270  NEXT I
```



To the right, an artist's reconstruction of the basic diagram which gives rise to the cover design.

THE VARYING EFFECT OF MORTALITY [†]

G. Ward, A.M.P. Society

Rates of Mortality, Probabilities of Survival and the Life Table

The chance of a person age x dying between age x and $x + 1$ is denoted by q_x, while the probability of a person surviving from age x to age $x + 1$ is denoted by p_x, that is $p_x = 1 - q_x$.

The probability of a person surviving from age x to $x + t$ is is denoted by $p_x(t) = p_x \cdot p_{x+1} \cdot p_{x+2} \cdot \ldots \cdot p_{x+t-1}$. In order to use these probabilities more conveniently, actuaries use a hypothetical model called a Life Table. It commences with an arbitrary large number of lives at a convenient age (usually 0, but sometimes other ages are used). The model shows the number ℓ_x at each age x surviving from the original number, assuming the lives at each age experience mortality according to some scale of mortality. Then:

$$q_x = (\ell_x - \ell_{x+1})/\ell_x$$

$$p_x = \ell_{x+1}/\ell_x$$

$$p_x(t) = \ell_{x+t}/\ell_x$$

At the age (ω) by which all lives are assumed to have ended,

$$\ell_\omega = 0.$$

As mortality is a probability relating to the average of a particular group of lives, mortality varies from group to group. For example, males and females in the Australian population exhibit different mortality, and this varies with time. The Australian Government Actuary uses $\ell_0 = 100\ 000$ for the Australian Life Tables. Using mortality measured over the years indicated, the Australian Life Tables give the following values of ℓ_x.

[†] This article, by a Senior Actuary for AMP, is reprinted with permission from *Parabola*, Vol.14, Part 2 (1978).

	Years 1901-10		Years 1970-72	
Age	Males	Females	Males	Females
0	100 000	100 000	100 000	100 000
20	84 493	86 459	96 473	97 596
40	75 887	78 001	93 150	95 848
60	56 782	63 247	77 574	86 719
80	14 330	21 356	23 399	44 242

This table indicates how dramatically mortality has reduced since the turn of the century, and also the greater longevity of females.

Expectation of Life

The average duration of life for a member of a group of lives is known as the "expectation of life". This can be calculated starting at any age, although "expectation of life" as commonly used means from birth. The actuarial formula is:

$$\text{Expectation of life} = (\tfrac{1}{2} + \ell_1 + \ell_2 + \ldots)/\ell_0$$

$$\text{Expectation of life for those alive at age } x = (\tfrac{1}{2} + \ell_{x+1} + \ell_{x+2} + \ldots)/\ell_x.$$

From the examples from the Australian Life Tables, expectations are as follows.

	1901-10		1970-72	
Age	Males	Females	Males	Females
0	55.20	58.84	67.81	74.49
40	28.56	31.47	31.61	37.16
80	4.96	5.73	5.52	6.88

Curve of Deaths

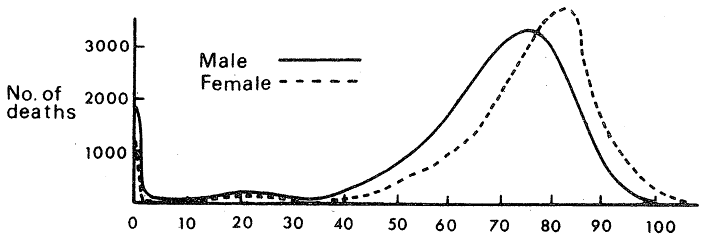
The number of deaths at age x is denoted in the life table as d_x, so that

$$d_x = \ell_x - \ell_{x+1} = \ell_x q_x.$$

As all lives must end by age ω,

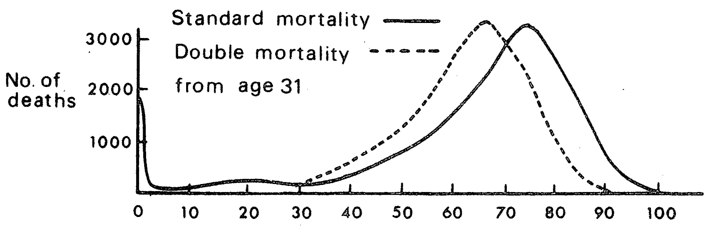
$$d_0 + d_1 + d_2 + \ldots + d_\omega = \ell_0 = 100\ 000.$$

The following graph shows the number of deaths at each age for males and females according to the 1970-72 Australian Life Tables.



The total area under each of these curves is the same, as each shows the deaths from 100 000 births. The lower mortality rates for females cause the years at which the peak number of deaths occur to be later than the peak for males. This difference accounts for the fact that expectation of life at birth for males (67.8 years) is less than that for females (74.5 years).

Annual mortality rates at young ages are fairly small. Thus both male and female rates reach 1% between 50 and 60. If the male rates of mortality were doubled at ages above 30 the expectation of life would decrease from 67.8 years to 61.1 years. The curve of deaths would change in the following manner.



This difference is believed to correspond roughly to the effect of being a heavy smoker (30-40 cigarettes a day), rather than being a non-smoker.

A problem for the reader: Can you explain how the formula for "expectation of life" is derived?

∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞

For more on the work of an actuary, see the paragraph on p.23 of this issue of *Function*.

6 ENTER 8 TIMES =[†]

Stanley J. Farlow, University of Maine

There was a knock on the door.

"Come in", Mr Takach said. The door opened and a youngish looking man of about 25, followed by a young boy of 17, entered the room. "What can I do for you?" Mr Takach said from behind his desk.

"If you have a few minutes we would like to talk to you about introducing some new ideas into Advanced Math 213", the older of the two was saying. He was Mr Mercier, Head of the Mathematics Department at Bangor High, and had brought with him one of his more creative students. The student had spent the summer working on a special project for Mr Mercier and had uncovered some ideas that could be useful, in fact downright revolutionary in the teaching of higher mathematics.

"We're always ready for new ideas", Principal Takach was saying. "Why don't you two have a chair?" Mr Takach was secretly sceptical about the whole affair but didn't want to offend his teacher. The teacher began, "It's about pocket calculators".

"What about 'em?" Mr Takach said.

"Well, ...", he hesitated.

"Well what?" Mr Takach asked quizzically.

"Well, .. well, my student Mr Henley has discovered a way to get around 'em."

"What do you mean, 'around 'em'?" the Principal said impatiently.

[†]Reprinted with permission from the *International Journal of Mathematical Education in Science and Technology* Vol.11, No.2 (1980).

"You see sir, I assigned Mr Henley to work on a summer project looking into the history of the pocket calculator and in the process he discovered how to carry out all the arithmetic operations with a paper and pencil".

"It's impossible", the Principal replied. The teacher responded, "Let me show you, sir - Mr Henley, what is six times eight?" The student thought for a while and then replied, "Forty-eight sir". The Principal pulled open his top desk drawer and took out his pocket calculator. Entering the proper numbers he then pushed the multiplication key. "He's right!" the Principal said, then spoke again, "What about nine times four?"

"Thirty-six", the student responded instantaneously. Again resorting to his calculator the Principal checked the calculation. "Amazing!"

"He can even multiply larger numbers."

"I don't believe it", the Principal said, "Multiply 112 by 68". The student took out a sheet of paper and wrote down the numbers. Both the teacher and the principal watched curiously. A few minutes later the student was finished.

"7616", he said. The Principal quickly checked the calculations.

"I don't believe it!" the Principal exclaimed, "How does he do it?" The teacher spoke, "As I said before, I assigned Mr Henley to work on a project this summer to look into the history of the calculator. During his studies he found some old manuals which explained how the calculators actually did their calculations. He has been practising these rules all summer. Not only can he multiply, but he can also carry out addition, subtraction, and even division."

"Division!" the Principal gasped.

"The rules are actually fairly simple - I have been practising them myself for the past month and already can add and subtract."

"Amazing!" the Principal said.

"This brings us to the reason why we wanted to see you", the teacher continued. The Principal was now listening with great anticipation. "We would like to introduce hand computation into Math 213. We would teach the students how to add and subtract in the fall term and then continue with multiplication and division in the spring. By the time graduation arrives the students would have mastered the four basic operations of arithmetic."

"It would be a fantastic accomplishment if it could be done!", exclaimed the Principal. "But what's the point?"

"Don't you see", continued the teacher, "By learning to carry out these operations with pencil and paper, the student would be freed from the routine tasks of operating and maintaining the calculator. His mind could be used for more *creative* tasks."

"I see", said the Principal; he was starting to understand the real significance of hand computation. "Maybe the students could even learn how to compute areas of squares, rectangles, triangles, and other geometrical shapes."

"Exactly", agreed the teacher. In the past these areas had been computed by keying the appropriate data into the calculator. "As a matter of fact in a few years we might even get to the point where the calculators won't be needed."

"Incredible!" said the Principal, "absolutely incredible". With this the Principal looked out the window at some distant object, "*There is no limit to the human mind*", he was thinking.

∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞

TWO CORRECTIONS

Like *Time*, *Function* erred. Twice in fact in our last issue.

1. The cover story gives (correctly) the theorem that almost all numbers between 0 and 1 are normal. However, it also states (incorrectly) that the exceptions - the non-normal numbers - are countable. To show that this is not so, let x be a normal number between 0 and 1 and let x' be the number produced by replacing each 0 in the binary representation of x by 00. Clearly x' is non-normal. But this means that the normal numbers may be mapped 1-1 onto a subset of the non-normal numbers, which must therefore be uncountable.

2. On p.29, the word "mass" was used in place of the more strictly accurate "thermal capacity". Thermal capacity is the product of the mass and the specific heat.

   For real substances, the specific heat always depends to some extent on the temperature, so that the physical example given (ice and water) is not strictly speaking, accurate. However, this does not affect the logic of the Landsberg argument. Landsberg has gone on to prove many more (and much more difficult) inequalities by his novel method.

For articles on the mathematics of countable and uncountable infinities, see *Function*, Vol.2, Parts 1 and 2.

CHESS - THE RULES AND THE TERMS

Chess is one of the most widely played, most demanding, most studied and most satisfying of board games. It is played on an 8 × 8 board of squares alternately coloured black and white. There are two players whose pieces are black or white also. The player handling the white pieces is referred to for convenience as "White". The same convention names "Black". The board is so arranged that White has a white square to his right (as, of course, does Black).

The game is one in which players move alternately, White always moving first. It is a game of "perfect information" in that no random element, such as the throw of a die or the draw of a card, is involved.

Each player has, at the start of the game, one king (*K*,♔), one queen (*Q*,♕), two rooks (*R*,♖), two bishops (*B*,♗), two knights (*N*,♘) and 8 pawns (*P*,♙). These pieces each move according to well defined rules, which are best understood in terms of a coordinate system now becoming standard (only the English- and Spanish-speaking countries now hold out against it).

As White (by tradition shown at the bottom of diagrammed positions) views the board, he sees 8 rows (or *ranks*) of squares. These are numbered 1,2,...,8. Counting from his left, he sees 8 columns (or *files*) which are labelled *a*,*b*,...,*h*. Thus the square nearest White on his extreme left is *a*1, while the square on Black's extreme left is *h*8, etc.

The following brief, and necessarily inadequate, summary of the rules aims to assist readers in following the ensuing article. It is, however, no substitute for a good basic manual. We recommend *Chess Made Easy* (publ. Chess Made Easy partnership, 139 Fisher Street, Malvern, S.A.) and probably available through your local newsagent. Alternatively, your municipal library, and maybe even your school library, can assist you here.

The pieces move in various ways. Kings may move to squares adjacent to those they occupy; Queens move in straight lines - horizontally, vertically or diagonally; Rooks move vertically or horizontally; Bishops diagonally and Knights to the opposite corner of a 3 × 2 rectangle.

These "chessmen" are termed the pieces. All bar the knight may proceed in the allowed manner up to a point where the path is blocked. If the path is blocked by an enemy piece (or pawn), they may move onto that square, "capturing" the enemy occupier by removing it from the board.

Pawns move forward along files one square at a time - although on the first move they may exercise an option of moving two squares. They capture, however, diagonally one square forwards to left or right.

There are also three other types of move allowed which space does not permit us to describe - the *en passant* capture, the "castling" manoeuvre, and "pawn promotion".

Initially White's pieces are deployed as follows: *K* on *e*1, *Q* on *d*1, *R*'s on *a*1 and *h*1, *B*'s on *c*1 and *f*1, *N*'s on *b*1 and *g*1. His pawns occupy the 8 squares on rank 2. Black's pieces are similarly placed on ranks 8 and 7 respectively.

The object of the game is for one player or the other to "checkmate" the opponent's king. That is to say - force it into a position in which it cannot avoid capture. The king may never actually be captured in the course of a game. If he is threatened with capture, the player involved must respond by either: (a) capturing the threatening piece or pawn, (b) interposing a piece or pawn on the line of attack (again, this does not apply to a threatening knight), or (c) by moving the king. Checkmate ensues if none of these options is available.

Moves are best recorded in the so-called "algebraic notation" based on the coordinate system described. In its simplest form, this gives the initial and final positions of the piece or pawn moved. Thus *e*2-*e*4, *d*2-*d*4 or *g*1-*f*3 are all legal and strong opening moves for White. Other legal moves, quite commonly employed, are *g*2-*g*3 or *c*2-*c*4. Black may reply *e*7-*e*5, *d*7-*d*5, *e*7-*e*6 and so on - but of course, his choice depends somewhat on White's first move.

This system is the one used on most commercially available chess-playing machines; in practice among human players, it is normally abbreviated in ways we won't go into here. We do, however, use (without explanation) another system along with that described above in the following article. This is the so-called "descriptive notation", which we use because many of our chess-minded readers will know it.

Positions are evaluated differently (and not in strict mathematical terms) by players of varying ability. Ultimately, the only true guide is the ability of a given player to win - or at least not to lose. This, however, often involves matters of such complexity that approximate rules are employed. One such sees the queen as worth 10 points, the rook as worth 5, the bishop and knight as each worth 3 and the pawn as one. The king - being irreplaceable - is not assigned a value.

A *sacrifice* occurs when a player, seeing a long-term gain, violates this approximate rule. Figure 1 of the ensuing article is a good example. (The first move shown and also later moves are given an exclamation mark, indicating "good move". The notation *ch*. following a move indicates that the opposing king is threatened.) Other tactical ploys are the *fork* - a double attack by one piece on two opponents, and the *pin* - the forcible immobilization of an opponent's piece, due to the fact that its movement would jeopardize the life of (in particular) the king.

The game is, all in all, challenging and difficult. It is hard even to describe it in so limited a space as here available. For many hours of enjoyable brain-exercise, learn it from *Chess Made Easy* or some comparable book, and then improve your skills through play and further study.

COMPUTER CHESS

Rodney Topor, Monash University

Introduction

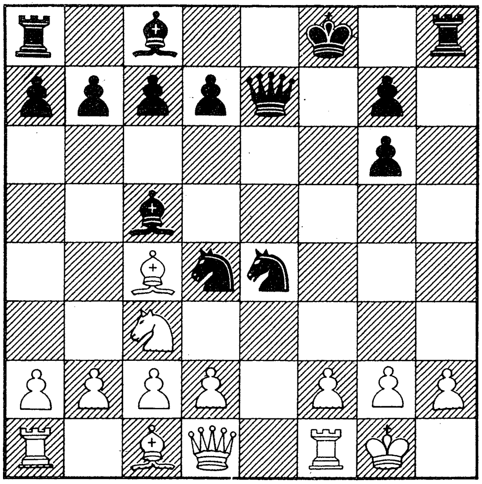
For hundreds of years man has been fascinated by the idea of machines that could play chess. In 1770 Baron von Kempelen constructed a machine which played excellent chess, winning most of its games against all comers including, supposedly, Frederick the Great and Napoleon. The machine consisted of a large box with chess pieces on top and a figure of a Turk which moved the pieces. The box contained a proliferation of gear wheels which were shown in magician fashion, one compartment at a time, to the audience before a performance. The method by which this machine played such good chess remained a mystery for many years. The solution was that the box actually contained a skilled (and small) chessplayer artfully hidden in the midst of all this machinery.

In contrast, a genuine piece of science was the electromechanical device constructed by Torres y Quevedo, a Spanish engineer, which was capable of mating with king and rook vs. king. Its construction in 1890 was a marvel of its day and copies of it were exhibited throughout the first half of the 20th century.

In 1950, when digital computers were first becoming available, the mathematicians Claude Shannon and Alan Turing independently described how such computers could be programmed to play chess. Their proposals were soon implemented, and their basic ideas are still embodied in the best of the modern chess programs.

Since 1970, computer chess programs have been regularly competing in tournaments against each other and against humans. The best of these programs have beaten many expert human players in tournaments, and they have beaten several masters at speed chess (5 minutes per game). Thus, they are now considered worthy of respect by the chess community. One of the best of these programs, Chess 4.7 (written by Slate and Atkins at Northwestern University), running on a super-fast Cyber 176 computer, had a USCF rating of 2050 two years ago, and played speed chess at about the 2300 level. A fine example of play occurred in the game Blitz vs. Belle in the 1978 North American Computer Chess Championship. In the position of Figure 1, black played the sacrifice 10 ... *R X P*! (i.e. 10 ... *h*8-*h*2!) after which white was lost whatever it did. The game continued 11 *K X R*, *Q - R*5 *ch*; 12 *K - N*1, *N - KN*6! (i.e. 11 *g*1-*h*2, *e*7-*h*4; 12 *h*2-*g*1, *e*4-*g*3!). In this position white can capture a piece, threaten the queen, and give check, only to be mated on the next move.

13 *Q* - *R*5,*P X Q*;14 *P X N ch N* - *B*6 mate (i.e. 13 *d*1 - *h*5, *g*6 - *h*5; 14 *f*2 - *g*3 *ch*, *d*4 - *f*3 mate).



Position after 10 O-O

Figure 1: Blitz vs. Belle(1978)

The rest of this article describes the techniques used by such programs, their achievements, and their limitations. To understand these techniques, we must first review the standard algorithm for playing any (finite, deterministic, 2-person) game (of perfect information) such as noughts and crosses, draughts, chess, or go (a Japanese game, of great difficulty).

Tree searching techniques

Starting with the initial position one generates all possible moves for the first player (white). From the resulting positions one generates all possible moves for the second player (black). This process is repeated for each player alternately until no more moves are possible from any position. The resulting structure is called a "game tree". Terminal positions (or "leaves") in this tree may be assigned values according to the result of the game, e.g. + 100 if white has won, 0 for a draw, - 100 if black has won. From each penultimate position white (black) would naturally choose the move which leads to the position of highest (lowest) value. More generally, the value of any position with white (black) to move is the maximum (minimum) of the values of the positions which can be reached in one move. Thus values (and corresponding best moves) propagate up the tree, being alternately maximized and minimized, until the value of the starting position (and the corresponding best first move) is found.

In practice, game trees for real games are of course too large to be searched completely. It is estimated that the game tree for chess, for instance, contains about 10^{120} terminal positions requiring centuries of computation time on the fastest imaginable computer.

Shannon's proposal was to generate only the start of this tree, to a fixed number of levels. (Each level, or half-move, is called a "ply" to avoid the ambiguity of the word "move".) A (heuristic) evaluation function would then be applied to each of the resulting terminal positions, and the resulting values backed up to select a move as before. The evaluation function returns an *estimate* of the true value of a position, giving a high score if white appears to be winning, and a low score if black appears to be winning. An example of a partial game tree, heuristic values of terminal nodes, and backed up values is shown in Figure 2.

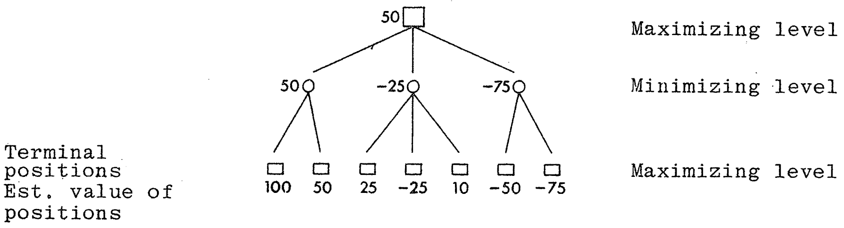


Figure 2: A partial game tree and backed-up values

This process of minimaxing may be described algorithmically in the following simple way.

```
MaxPos($p$) ⇐
  if $p$ is terminal then return $f(p)$
     else return MaxList(succs($p$))

MaxList($ps$) ⇐
  local $p$,val; val := $-\infty$;
  for each $p$ in $ps$ do
      val := max(val, MinPos($p$));
  return val
```

```
MinPos($p$) ⇐
  if $p$ is terminal then return $f(p)$
     else return MinList(succs($p$))

MinList($ps$) ⇐
  local $p$,val; val := $\infty$;
  for each $p$ in $ps$ do
      val := min(val, MaxPos($p$));
  return val
```

Figure 3: The minimax algorithm

In this algorithm, *p* is a position, succs(*p*) the list of positions reachable in one move from *p*, *ps* is a list of positions, and *f* is the heuristic evaluation function. This algorithm uses only enough storage for the current branch, and does not require the whole tree to be built before being processed. In practice, a simple modification must be made to return the move leading to the best position.

Note that if the evaluation function was perfect (i.e. always returned the correct value of a position), it would only be necessary to look one ply ahead.

In practice, the further ahead one looks, the more accurate is the resulting backed up value.

Shannon proposed three different types of program:

A. A program based on a complete search to a given depth using a simple terminal evaluation function.

B. A program that searched selectively by considering only moves which appeared promising a priori (forward pruning).

C. A program that was goal oriented, though it was not specified how this could be done.

All three types of programs have subsequently been implemented with varying degrees of success.

The efficiency of the tree searching method described above may be greatly increased by observing that it is unnecessary to consider some parts of the tree at all. For example, if the first move considered in a given position leads to a forced win, there is no need to even consider the remaining moves from that position. More typically, in Figure 4 there is no need to consider further moves from position 3 since black would never move to this position.

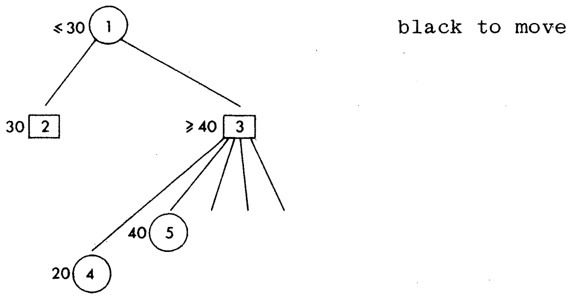


Figure 4: α-β pruning

This method of (backward) pruning is called α-β pruning because of the names given to two key parameters in its usual description. α is the value of the best move found so far and β is the best possible value. For example, when considering the move from position 3 to position 5 in Figure 4, α = 20 and β = 30. The algorithm may be expressed as a simple generalization of the above minimax algorithm.

```
MaxPos($p$, $\alpha$, $\beta$) ⇐
  if $p$ is terminal then
     return max($\alpha$, min($f(p)$, $\beta$))
  else
     return MaxList (succs($p$), $\alpha$, $\beta$)

MaxList($ps$, $\alpha$, $\beta$)
  local $p$, val; val := $\alpha$;
  for each $p$ in $ps$ do {
      val := max(val, MinPos($p$, val, $\beta$))
      if val > $\beta$ then return $\beta$};
  return val
```

Figure 5: The α-β algorithm

MinPos and MinList are defined analogously, as in Figure 3, but with the roles of α and β reversed. Initially, with white to move, we would call MaxPos(p, $-\infty$, ∞).

The α-β algorithm is used in all modern chess programs as it has the following important properties:

1. It always returns the same value as a complete search.

2. In the worst case (when the best move from each position is considered last), it considers exactly the same positions as a complete search. The overhead, even in this case, is minimal.

3. In the best case (when the best move is always considered first), the number of positions considered is reduced dramatically. If there are B possible moves from each position, and the search depth is D, the number of terminal positions reached falls from B^D to about $2B^{D/2}$. This allows the tree to be searched to almost twice the depth, greatly improving the quality of play.

4. The average number of positions considered under various probability distributions of the terminal positions' values, is a subject of ongoing research.

A description of the minimax and α-β algorithms may be found in Chapter 3 of Frey's book (see references following this article).

Modern chess programs extend this basic tree searching algorithm in the following ways:

1. Quiescence: Both Shannon and Turing realized that only quiescent positions should be evaluated. Quiescent positions are ones in which no capture or check is possible. Accordingly, if a terminal position of a game tree is not quiescent, capture (and sometimes checking) sequences are followed to completion before the evaluation function is applied. This may extend the effective search depth by several plies.

2. Full width searching: The authors of earlier. Shannon type *B* programs were often embarrassed in tournaments when their progams failed to even consider the best move because it was "pruned" by the move selection routine. Accordingly, all the best programs are now Shannon type *A* programs, i.e. they consider every possible move from each position. They still need to order the moves in decreasing initial promise to gain maximum benefit from α-β cutoffs, but this task is made simpler by iterative searching (described next).

3. Iterative searching: Instead of doing a single α-β search to maximum depth D, modern programs now perform a sequence of searches of increasing depth $d = 1,2,\ldots,D$. The best moves found in the depth d search are then considered first in the depth $d + 1$ search. This idea has two benefits: first, it increases the efficiency of the α-β search, and secondly it eliminates the possibility of running out of time after considering the consequences of only a few (complex) moves. Moreover, because the number of moves from each position is quite large, the overheads of the iterative search are low and the final depth D search still takes almost all of the time available.

4. Transposition tables: Programs with access to large main memories store each position they encounter in a hash table, together with the resulting value and move found by searching. Every time they reach a new position, they first look up the table to see whether they have encountered the position before. If so, they can use the previously computed value and move without having to search from that position again. This technique is most effective in deep searches, especially in endgames. Chess 4.7 can store up to 32 000 positions in a table of about one million 60-bit words.

5. Analogies: This idea, used by the Russian program Kaissa, is that if a new position has "similar" characteristics to an earlier position, use the value and move of the earlier position. It is a generalization of the idea of transposition tables, but requires (difficult) chess reasoning to determine whether two positions are "similar".

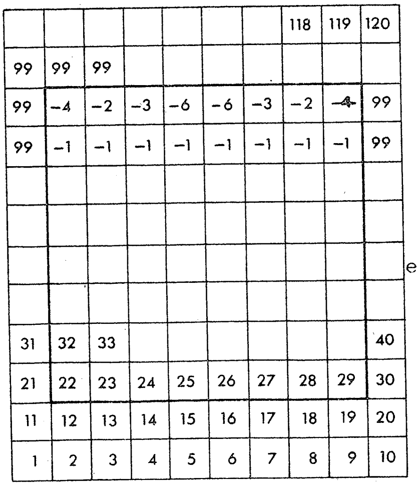
6. Killer heuristic: A move found to be a refutation on one branch may as well be tried first on other branches.

Using these techniques, together with efficient board representation and evaluation functions (described below), programs such as Chess 4.7 typically do exhaustive searches of 6-7 plies in middle games and 10-12 plies in endgames, evaluating 200 000 - 800 000 terminal positions in a 3-minute move. In speed chess, 5-second moves, they do exhaustive searches of 4-5 plies. Their good performance is entirely due to the size of these trees.

Board Representation and Move Generation

1. Shannon's original proposal: Use an 8 × 8 matrix containing 0 for empty squares, positive integers for white pieces and negative values for black pieces. Each type of piece could be assigned a particular value, e.g. 1 = pawn, 2 = knight, 3 = bishop, etc. This representation is inefficient because accessing 2-dimensional arrays is slow on most computers, and because move generation is awkward. One of the following two representations is now used instead.

2. 10 × 12 board: The board is represented by an integer vector of length 120, indexed as shown in Figure 6.



eg. *N* moves: ±8, ±12, ±19, ±21

R moves: ±*k*1, ±*k*, 10

etc.

Figure 6: The 10 × 12 board

Squares outside the board are given an impossible value such as 99. This makes it easy to recognize the boundary of the board. Squares on the board contain values as in Shannon's representation. Moves may then be generated by adding one of a set of constant values to a piece's current position, e.g. +/-8, +/-12,+/-19,+/-21 for a knight, and successive integer multiples of +/-9 or +/-11 for a bishop. Moves for "sliding" pieces are generated by adding successive integer multiples of the appropriate constant until a nonempty square is reached.

3. Bit board: A bit board is a 64-bit "word" which represents a set of squares on the board. Each bit in the word corresponds to a square on the board. The square is in the set if and only if the corresponding bit in the word is 1. To describe a chess position requires 12 such bit boards: one bit board represents all white pawns, a second represents all black pawns, a third represents all white knights, and so on. Since there are six types of piece, this requires 12 bit boards in all.

The advantage of this approach is that for each piece on the board one can equally easily store the set of squares it can reach in one move, the set of pieces it attacks, the set of pieces attacking it and so on. To update these boards one must store permanently the set of squares to which any piece could move from any position.

Bit boards may be very efficiently updated using a computer's Boolean instructions such as logical-or (+), logical-and (*), and complement (not). For example, to determine whether white can successfully fork black's queen and king with a knight in one move, it is only necessary to determine whether the expression

```
  knight moves from black king square

* knight moves from black queen square

* knight moves from white knight square

* not (white pieces)

* not (squares attacked by black)
```

is zero or not. The use of bit boards in this way allows even more complex chess relationships to be represented efficiently.

Evaluation Functions

The strength of a program which uses tree searching in the above way depends on the size of the tree searched and the accuracy of the evaluation function. It is surprising how simple the evaluation of the function can be for the program to still find good moves. Chess 4.7 is considered to have one of the best evaluation functions. Yet it takes into account only the following factors.

1. Material balance: This is usually the dominating factor. Pieces are given their usual values, which vary slightly depending on the state of the game. The winning side is encouraged to exchange pieces but not pawns. Certain endgames are known to be drawn.

2. Position: This takes into account the number of pieces being attacked, pawn structure, development, mobility, occupation of the centre, nearness to the opponent's king, and king safety. Credit is given for rooks which are doubled, on the seventh rank, or on open files. In the endgame the king is encouraged to be in the centre and near to any pawns.

The most important property of Chess 4.7's evaluation function is that it is very fast, allowing deeper searches to be performed. This speed is achieved largely by differential updating, i.e. only reevaluating those factors which have changed in the last move.

Openings and Endgames

All programs now contain opening "books", large sets of positions with the recommended move, as copied from chess textbooks. These books speed up and improve the programs' opening play. Their size may be up to 100 000 positions. Even without these books, the normal evaluation function leads the program to push centre pawns, develop minor pieces, castle, and generally increase mobility. This is adequate against weak opponents.

It used to be possible to defeat any chess program by playing the simplest possible line, exchanging all the pieces as quickly as possible, and then winning the endgame. This was possible because in the endgame it is usually necessary to form a plan of greater depth than the program's search horizon. Although most programs are still unable to form plans, the use of super-fast computers, special hardware (in some cases), and transposition tables mean this strategy no longer works against the best programs. In Figure 7, Chess 4.7 found the correct continuation only after an iteratively deepening search to a depth of 26 plies. Such a deep search, requiring 632 seconds and evaluating 3.3 million terminal positions, was only possible because of the use of transposition tables.

Increasingly, programs are improving their endgame performance by using special algorithms and heuristics. For example, to mate with king and rook vs. king (*KRK*), it suffices to decentralize the opponent's king, keep your rook near the opponent's king and protect your rook. Special databases have been constructed which result in virtually perfect play in the endgames *KPK*, *KPKP*, *KPPK*, *KRKN*, *KRKB*, *KQKR* and *KRPKR*.

Limitations

Despite the fact that programs such as Chess 4.7 have expert ratings and have beaten masters at speed chess, they rely on brute-force computing power for their performance, and as such have many inherent weaknesses.

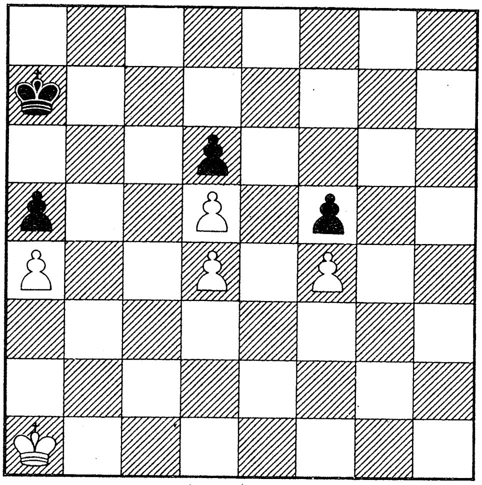


Figure 7: Endgame solved by Chess 4.7

1. Limited chess knowledge: Concepts such as pins, forks, discovered attacks, whether a pawn can be promoted or not and back-rank threats are only discovered by an inefficient search process.

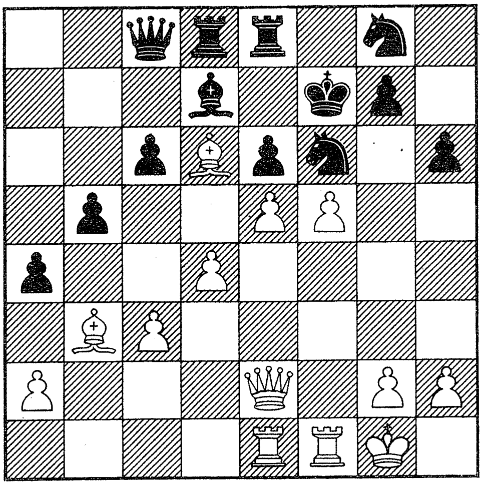
2. Inability to form plans: Because programs are unaware of chess concepts such as those listed above, they are unable to set themselves goals or subgoals to achieve, or to form plans. Thus, in many endgames, such as that in Figure 7, the correct move is only discovered by an exhaustive search (if at all), and then it is often chosen for the wrong reason.

3. Horizon effect: This effect is present in any program based solely on a fixed-depth lookahead. It is a consequence of the fact that anything which is not detectable at evaluation time (beyond the search horizon) is assumed not to exist. For example, a program about to lose a rook on the queen side of the board, may postpone the inevitable by first sacrificing a bishop on the king side, thereby obtaining an even worse position. Examples of this effect occur frequently in computer chess games. It should be noted, however, that a sufficiently deep search, together with quiescence analysis, ameliorates the effect.

Alternative Approaches

Clearly, human chess masters do not play using the methods described above. Psychological studies have shown that they consider fewer than 100 positions to select a move; that they conduct very narrow searches if they search at all, and that they may have up to 50 000 stored patterns which they can recognize at a glance. In short, they rely on perception rather than search.

Chess programmers know this, but have found it difficult to build these ideas into working programs. Notable attempts have been made by Newell, Shaw and Simon (AI pioneers), Botvinnik (a former world chess champion), and Berliner (a former world correspondence chess champion). A recent program by David Wilkins at Stanford University attempts to find the best move in tactically sharp middle game positions. One achievement of Wilkins' program was to find the best move, *Q-R*5 *ch* (*e*2-*h*5*ch*), in the position of Figure 8. This sacrifice, which leads to mate in ten moves, was made by a world champion in a simultaneous exhibition. Wilkins' program generated a game tree of depth 19 containing only 109 nodes in 20 minutes to solve this problem.



1.	*Q-R5ch*	*N*×*Q*	*e2-h5ch*	*f6-h5*
2.	*P*×*Pch*	*K-N3*	*f5-e6ch*	*f7-g6*
3.	*B-B2ch*	*K-N4*	*b3-c2ch*	*g6-g5*
4.	*R-B5ch*	*K-N3*	*f1-f5ch*	*g5-g6*
5.	*R-B6ch*	*K-N4*	*f5-f6ch*	*g6-g5*
6.	*R-N6ch*	*K-R5*	*f6-g6ch*	*g5-h4*
7.	*R-K4ch*	*N-B5*	*e1-e4ch*	*h5-f4*
8.	*R*×*Nch*	*K-R4*	*e4-f4ch*	*h4-h5*
9.	*P-N3*	any	*g2-g3*	any
10.	*R-R4* mate		*f4-h4* mate	

Figure 8: Forced mate found by Wilkins' program

Like many recent AI programs, Wilkins' program uses production rules to form plans. It performs a detailed static analysis of the current position, noting features such as pins, forks, overloaded pieces, etc. These features then suggest plans. Each plan has a tree structure which specifies the principal variation, relevant features from possible positions, expectations and goals. A small search of the game tree is performed to determine which of several plans is best, or to determine that a single plan will work. Unsuccessful searches return detailed information about why the search failed.

Although this program has not yet been included in a complete chess program, its results are already very promising. Such work is certainly of greater scientific interest than brute force tree searching programs.

Conclusions.

Chess programs are very complex, as is chess. All the simple things have already been done, and it would take two years' work to get to the state of the art. In the 1979 North America Computer Chess Championship one program was in a position to promote a pawn but, because it already had a queen and was written to handle only one queen at a time, was forced to promote the pawn to a bishop. At this stage the opposing program complained that it could not handle promotion to anything but a queen. The situation was resolved and the game continued, but it did illustrate the pitfalls that await the chess programmer.

Predictions regarding the future of chess programs are dangerous. In 1958, the Nobel prize winner Herbert Simon predicted a program would be world chess champion within 10 years. More recently, following the improved performance of programs on super-fast computers, Berliner and Newborn (organizer of many computer chess tournaments, and a continual contestant) each predict that a program will probably be world champion by 1990. If a brute force tree searching program does become world chess champion, such an approach cannot possibly succeed for *Go*, and this will have to become the task par excellence for game playing programs.

References

There are now many books on computer chess. The best of these is *Chess Skill in Man and Machine*, P. Frey (ed.), Springer-Verlag, 1977. This contains an introduction to computer chess, an account of psychological studies on how humans play chess, a detailed description of Chess 4.5, and several other interesting articles.

∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞

THE WORK OF THE ACTUARY

[*Richard Greenfield, partner in Campbell and Cook, Consulting Actuaries, supplies the following description of the actuary's work. He also writes that "an actuary is a person who wanted to be an accountant but didn't have the personality". This, he tells us, is the standard joke against the profession. Eds.*]

"A person who specializes in mathematical techniques relevant to financial transactions, particularly those affected by contingencies such as mortality and morbidity (sickness)" is perhaps a suitable general description of an actuary. The basic training of an actuary involves a study of compound interest (the assessment of the relative values of payments made at different times), the development, analysis and application of statistics (of which the (p.4-p.6) article by Mr Ward is an example) and combining the two (that is, the evaluation of payments which are contingent upon survivorship, death, etc.). Having grasped these techniques, the trainee actuary then proceeds to acquire a detailed knowledge of the areas in which they are applied. These are life insurance, superannuation, investment and, more recently but still to only a minor extent, general insurance. An actuary is also required to gain an understanding of basic economics so that the financial assumptions on which his calculations are based are reasonable.

LETTERS TO THE EDITOR

LUMP-SUM IN LIEU OF INCOME

The other day a colleague telephoned from the Family Court. Acting for a wife, he was negotiating with the husband's representative for a lump-sum payment in lieu of instalments of maintenance for a child of the marriage. As a matter of urgency he needed to know, "What sum of money invested at 10% p.a. with quarterly rests will produce an income of $25 a week and cut out exactly in $5\frac{3}{4}$ years?"

I recalled an old formula for the calculation of the balance of purchase money outstanding at the expiration of a terms contract of sale, *viz.*:

$$w = \frac{100stu - (100stu - qr)(1 + \frac{r}{100s})^{sv}}{r},$$

where
- q = the original balance of purchase money in $'s,
- r = the rate of interest per cent per annum,
- s = the number of interest periods (= the number of rests) per annum,
- t = the number of instalments payable during one interest period,
- u = the amount of each instalment in $'s,
- v = the duration of the contract in years, and
- w = the balance owing at the expiration of the contract in $'s.

To answer my colleague's question I took that formula and, making $w = 0$, transposed it to obtain the equation,

$$q = \frac{100stu}{r}\left[1 - \frac{1}{(1 + \frac{r}{100s})^{sv}}\right]$$

A substitution of the given particulars produced:

$$q = \frac{100 \times 4 \times 13 \times 25}{10}\left[1 - \frac{1}{1{\cdot}025^{23}}\right] = \$5{,}632{\cdot}94,$$

which I confidently provided as the answer.

Upon reflection it occurred to me afterwards that that was not strictly correct; first, because a year contains more than $4 \times 13 = 52$ weeks (the true average is 52·1775 weeks) and, secondly, because interest would not be paid upon the instalments withdrawn from the lump-sum during the first quarter. Clearly, the equation needed adjustment. Treating the weekly instalments as $1.00 each (i.e. $u = 1$, for any multiple could be simply calculated) and assuming that similar enquiries would always involve weekly income, I rephrased the question: "What sum of money ($\$q$) invested at r% p.a. with $1/s$ of a year rests will produce an income of $1.00 a week and cut out exactly in v years?" The revised equation became

$$q = 52{\cdot}1775\left[\frac{1}{s} + \frac{100}{r}\left\{1 - \frac{1}{(1 + \frac{r}{100s})^{[sv-1]}}\right\}\right]$$

A substitution here of the particulars given by my colleague produced

$$q = 52{\cdot}1775\left[\frac{1}{4} + \frac{100}{10}\left\{1 - \frac{1}{1{\cdot}025^{22}}\right\}\right] = 231{\cdot}7387,$$

which, when multiplied by 25, gave the true answer of \$5,793·47.

G.J. Strugnell, Solicitor,
106 Bell Street, Coburg.

∞ ∞ ∞ ∞

TATTSLOTTO SYSTEMS

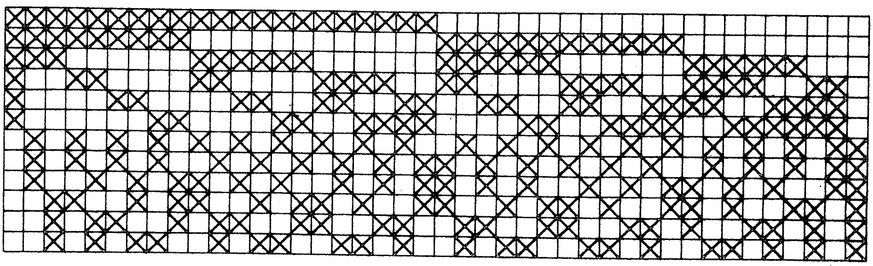
Most readers would be aware that if you play Tattslotto you must have a minimum of four winning numbers in one combination of six numbers in order to qualify for a prize.

If you wanted to play a full 'system 12' entry it would cost you \$231 as there are 924 combinations of six from 12.

Obviously this would be far too expensive for the average punter. How could we reduce the outlay and still retain some sort of guarantee assuming that our 12 selections included at least four of the winning numbers?

In other words, if we select 12 numbers to play Tattslotto, what is the *minimum* number of combinations required to guarantee that we will win at least one prize providing that our 12 numbers include at least four of the winning numbers?

The plan below sets out how 42 entries can cover the twelve numbers.



This plan guarantees that if your 12 selections included four or more of the winning numbers - you would be certain of winning at least one prize.

1. To enter this plan you would place your 12 numbers at the left of the diagram.

2. Transfer your numbers on to the entry forms by matching the corresponding ×'s as set out in the diagram.

With this system the chance of winning at least one minor prize is one in 9·57, as can be revealed by counting up all the possibilities. The total outlay involved is $10.50. In commercially available plans the outlay is much higher: $18.50 in one recently advertised.

Michael Morley,
8 Hocknell Street, Canterbury.

[*We have had to condense a much longer letter from Mr Morley. We agree that if the object is to maximize one's chances of winning a* prize, *then his plan is best. If, however, one wishes to maximize the* expected return, *this is not so. Eds.*]

∞ ∞ ∞ ∞

COHEN'S FIRST THEOREM

The square root of an integer of the form $n^2 + 1$ generates (in a way I will make clear) an infinite set of Pythagorean triads such that for every triad in a given set there is the same mathematical relationship between a corresponding pair of numbers in that triad, and such that that same relationship is unique to that particular set of triads and hence to the square root of that integer.

To explain what I mean, take first $n = 1$, so that we want the triads generated by $\sqrt{2}$. If $x^2 = 2$, then $x - 1 = 1/(x + 1) = 1/[2 + (x - 1)]$ and by proceeding in this way we can generate a continued fraction for $x - 1$, that is to say $\sqrt{2} - 1$. [*Continued fractions were discussed in* Function, *Vol.4, Part 4. Eds.*] The continued fraction is

$$\sqrt{2} - 1 = 1/(2 + 1/(2 + 1/(\ldots))),$$

and if we approximate this rationally by n/d, the next slightly better approximation will be $1/(2 + n/d)$ or $d/(2d + n)$.

Using this formula, we deduce a sequence of approximations. The first approximation is $\frac{1}{2}$, the second is $\frac{2}{5}$, the third $\frac{5}{12}$ and so on. The successive numerators (or denominators) form a sequence of positive integers 1,2,5,12,29,70,169,408,... where each term is obtained by doubling the preceding term and adding the one before that.

Take any two successive terms of this sequence and use the well-known formula for generating Pythagorean triads. Say the terms are m,p. Then the triad is $p^2 - m^2$, $2pm$, $p^2 + m^2$. The simplest case is $m = 1$, $p = 2$. This gives the (3,4,5) triangle. The next case is $m = 2$, $p = 5$ which gives the (20,21,29) triangle. Next is the (119,120,169) triangle.

What do all these triangles have in common? Suppose we had a triangle in which the base and the height were of equal length, then the hypotenuse would be $\sqrt{2}$ times that length and could not possibly be an integer. For the best possible approximation the base and the height should differ by just one unit, which is exactly what we find! In every case, the two sides forming the right angle differ by just one unit - no more and no less.

Notice also that the hypotenuse and the perimeter of each of these triangles is also a member of the sequence.

The next value of n is 2 and this allows us to generate triads from $\sqrt{5}$. The Cohen sequence for $\sqrt{5}$ is 1,4,17,72,..., where each term is four times the preceding term plus the one before that. The triads are (8,15,17),(136,273,305), (2448,4895,5473),etc. and the second longest side here is twice the shortest side alternately plus or minus one - just what we need for the best possible approximation to a triangle whose hypotenuse is $\sqrt{5}$ times its shortest side. (Here however, the perimeter may not be a member of the sequence, although the hypotenuse is.)

This same pattern applies to $n = 3$, and gives triads generated by $\sqrt{10}$. For these the second longest side is three times the shortest alternately plus or minus one, exactly as we would expect.

A similar process can be applied to the square root of numbers $n^2 - 1$. The simplest case is $n = 2$, which generates triads from $\sqrt{3}$. After that, there are triads from $\sqrt{8}$ and so on. These are more difficult than the ones generated by the square roots of $n^2 + 1$.

S.J. Cohen,
348A Bourke Street,
Darlinghurst, N.S.W.

[Again, we have had to condense a much longer letter - or rather succession of letters. This is a fascinating topic and there is much more to it. The case of $n^2 - 1$ *is rather more complicated and, as Mr Cohen notes, it is easy to miss triads - the method requires some extension here, which Mr Cohen leaves as a challenge to the reader. His first two triads from* $\sqrt{3}$ *are (3,4,5) and (8,15,17), so that* $\sqrt{3}$ *is approximated first by 4/3 and then by 15/8. The (3,4,5) triad does not appear in direct analogy to the* $n^2 + 1$ *case, nor does the next triad (33,56,65). Mr Cohen writes that he is currently unemployed. We wish him well. Eds]*

∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞

UNIQUENESS

"[President Reagan's assailant] was identified as a 22-year old white from Colorado."

ABC News, 7.15 a.m., 31.3.81.

PROBLEM SECTION

SOLUTION TO PROBLEM 4.3.1

The problem, submitted by Colin Wright, a student at Monash, read as follows:

Given a cube, it is possible to cut a hole in it through which a larger cube can be passed?

Colin's answer, the only one received, reads as follows.

Yes, given a cube of side length one unit, it is possible to cut a hole in it such that a cube of side length $\sqrt{9/8}$ can be passed through it.

To do this, start with a cube of unit size and picture inside it a square of unit size; horizontal, and half way between the top and bottom faces of the cube; as per Figure 1.

Rotate this square 45° about the axis shown in Figure 1 to produce the situation depicted in Figure 2. At this point the square is still of unit size because the closest and farthest edges are still in contact with the appropriate faces of the cube.

Now rotate the square about the axis shown in Figure 2, increasing the size of the square so that the edges remain in contact with their respective faces of the cube. Continue rotating it until prevented from doing so by the edges of the cube. The situation is now that of Figure 3.

The measurements shown in Figure 3 can be easily calculated from the diagram using the knowledge that the figure inside the cube is still a square. From these measurements the size of the square can easily be determined.

Now, if one were to cut a hole in the cube such that the hole is the projection of the square perpendicular to itself, one would then have a hole that is square in cross section and large enough to allow a larger cube than the original to pass through.

Figure 1

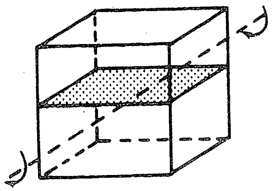


Figure 2

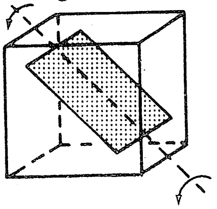
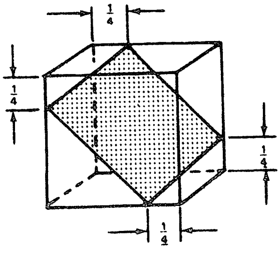


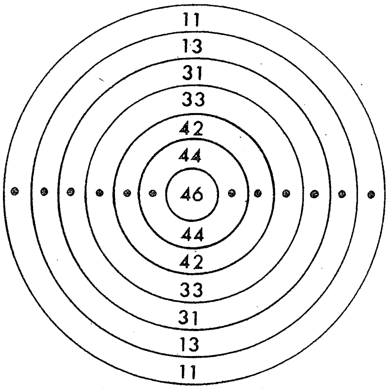
Figure 3



SOLUTION TO PROBLEM 4.5.1

This read:

What is the minimum number of hits necessary to score exactly 100 on this rather unusual rifle target?



A number of readers gave the answer 8 ($6 \times 13 + 2 \times 11 = 100$), but none provided a complete proof. For conscience' sake we put the matter on a computer which looked at all combinations of up to 8 shots (beyond which the matter is clearly impossible). This confirmed our belief that the above answer gives in fact the only way to score 100.

SOLUTION TO PROBLEM 4.5.2

In how many ways can LUCKY DIPS be spelt in the figure at the right?

L	U	C	K	Y
U	C	K	Y	D
C	K	Y	D	I
K	Y	D	I	P
Y	D	I	P	S

Since to go from the *L* to the *S* means going down 4 spaces and across 4 spaces, 8 moves are involved. We may choose the "down" moves to occur on any 4 of these 8, so there are $\binom{8}{4}$ or 70 different ways.

SOLUTION TO PROBLEM 4.5.3

The problem was:

Show that if a set *S* of 10 different numbers is chosen from 1,2,3,...,98,99, there will always be two completely disjoint subsets of *S* whose sum is the same. For instance, if $S = \{1,18,20,22,33,49,57,58,83,87\}$, then $22 + 49 = 18 + 20 + 33$. This time, it happens that $1 + 57 = 58$, too.

This surprising result may be proved by use of a very powerful method, known as the "pigeonhole principle", often attributed to the 19th century mathematician *Dirichlet*.

First note that the ten largest numbers we could choose (90,91,...,99) add up to 945. Hence, no matter what *S* we choose, the sums available for the subsets are precisely the numbers 1,2,...,945. But each member of *S* either belongs to a particular subset or it doesn't, so that there are $2^{10} - 1$ subsets of *S* (omitting the empty set). I.e. there are 1023 subsets and there are only 945 sums available. Thus at least one of these sums must be shared by two or more subsets.

For another problem whose solution involved the pigeonhole principle, see Problem 3.2.3.

This issue's new problems all come from the one source. They were those issued with the menu at the annual dinner of the Applied Mathematics Division of the Australian Mathematical Society. The wording is slightly altered.

PROBLEM 5.2.1

If x,y are digits and $x028y$ is divisible by 23, what are x,y?

PROBLEM 5.2.2

A farmer has 10 sheep, all of which are identical in their feeding characteristics, and 3 paddocks which are all equally good pasture in all respects. He puts 6 sheep in the first paddock, 3 in the second and one in the third. After 3 days, the 6 sheep in paddock 1 have eaten it out and he sells the 6 sheep; after 4 more days, the 3 sheep in paddock 2 have eaten it out and he sells them. When will the last sheep exhaust the pasture of paddock 3?

PROBLEM 5.2.3

One circuit of a running track is 1300 metres. The track is to be marked at the least number of points which can be used as starting and/or finishing lines for races of any multiple of 100m. Where should the points be chosen?

PROBLEM 5.2.4

Arrange the 52 cards of a pack into 13 tricks of 4 cards each so that:

(1) in each trick all cards belong to different suits;

(2) in each trick all cards are of different ranks;

(3) each pair of tricks has just one rank in common;

(4) given any two ranks, they occur together in just one trick;

(5) no trick contains more than one pair of cards with adjacent ranks.

[Hint: See Problem 5.2.3.]

∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞

AN ALGORITHM FOR $\sqrt[3]{N}$

Let a_n be an approximation to $\sqrt[3]{N}$. Then $a_{n+1} = \sqrt[4]{(a_nN)}$ is a better approximation. (Can you prove this?) The sequence $\{a_n\}$ thus converges to $\sqrt[3]{N}$. Although the convergence is not rapid, the algorithm works well on even quite primitive calculators. A square root button (hit twice) is needed, and a memory (to store N) helps.

A convenient value for a_0 is $\sqrt{N}$, assuming $N > 0$.

THE HILTON SUMMATION

The *r*th *triangular number* t_r is defined as the sum of the first r positive integers:

$$t_r = 1 + 2 + 3 + \ldots + r.$$

Clearly the triangular numbers satisfy the equation

$$t_r - t_{r-1} = r. \qquad (1)$$

They also satisfy another equation:

$$t_r + t_{r-1} = r^2. \qquad (2)$$

The proof of this statement is geometric (see the diagram, which also makes clear the reason for calling t_r 'triangular').

Multiply Equation (1) by Equation (2) to find

$$t_r^2 - t_{r-1}^2 = r^3. \qquad (3)$$

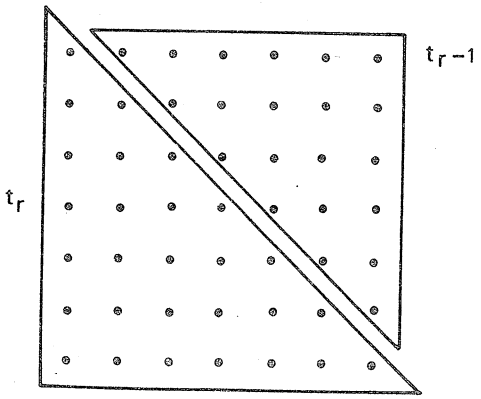
If now we add up the first n cubes $1^3, 2^3, 3^3, \ldots, n^3$, we find

$$\begin{aligned}1^3 + 2^3 + 3^3 + \ldots + n^3 &= (t_1^2 - t_0^2) + (t_2^2 - t_1^2) \\ &\quad + (t_3^2 - t_2^2) + \ldots + (t_n^2 - t_{n-1}^2) \\ &= t_n^2, \text{ as } t_0 \text{ is zero.}\end{aligned}$$

It follows that

$$1^3 + 2^3 + 3^3 + \ldots + n^3 = (1 + 2 + 3 + \ldots + n)^2.$$

This proof was first produced by Jeanette Hilton, a 13-year old schoolgirl. It was published in *The Mathematical Gazette* in 1974.



TRUE OR FALSE TESTS

Let us imagine a test (you may have been subjected to something like it) consisting of 100 statements. Opposite each statement are two boxes, one marked T (for "true"), the other F (for "false"). Let us further suppose that the examiners are fair-minded. The correct response, indicated by the placement of a cross in the appropriate box, is as likely to lie with a T as it is with an F. The examiners, we will assume, award a mark of +1 for every correct response and 0 for every incorrect response.

This is the situation we now analyse. It falls under the rubric of the binomial distribution, and a simple version of it at that. The chance of arriving, by guesswork or other random means, at exactly k correct answers is precisely $\binom{100}{k}(\frac{1}{2})^k(\frac{1}{2})^{100-k}$. This formula applies to all strategies: picking all the T's, picking all the F's, tossing a coin, or simply gazing into space and having a punt.

For an examinee with no knowledge, the expected score is (clearly enough) 50. The standard deviation, for a sample of such candidates, is given by the formula

$$\sigma = \sqrt{(npq)}.$$

Here $n = 100$ and $p = q = \frac{1}{2}$, so that σ, the standard deviation, is 5.

For some of the questions that follow, we use the normal distribution as an approximation to the binomial. Now, the binomial distribution is discrete - the relevant histogram goes up and down in steps. The normal distribution is continuous - that is to say, its graph is a smooth curve. Important points of principle are involved here.

In the binomial case, we can give sense to the question: "What is the probability that a student, knowing nothing of the subject, will score a mark of exactly 45?" The answer is, by the formula given above, $\binom{100}{45}/2^{100}$ or 0·04847.

If the normal distribution is used, we cannot, unless we accept the answer "zero", ask this question. We instead require a convention (called the *continuity correction*) according to which, we ask the question: "What is the probability that the score lies between 44·5 and 45·5?"

This we can look up in standard tables - or, in some cases, ask our pocket calculators. The answer from this viewpoint is 0·04839, an excellent approximation to the exact result derived above.

So let us use these distributions to ask (and resolve) some basic questions.

Q1. What should the pass-mark be on this test?

A1. Here the answer, assuming the pass-mark to correspond to 50% assured knowledge, is 75.

Q2. What is the probability that a student who knows nothing about the subject scores 55 or more?

A2. The student has to score at least one standard deviation above the mean. The probability of this, by pure chance, is approximately 1/6 from normal tables (more accurately, ·1841 using the binomial distribution).

Q3. A student on a pre-test scores 45, and on a post-test scores 55. How certain can we be that this represents genuine progress?

A3. Clearly we assume the student knew nothing the first time. Thus the two tests are independent and the probability is that of Q2. We have somewhat shaky grounds for confidence in the student's improvement. There is a moderate probability he could score as well by pure chance.

Q4. What do we make of scores significantly below 50?

A4. Suppose someone scores 0. Systematically, the student has got *every answer wrong*! Is this abysmal ignorance - or possibly a bright student who misread the instructions? If, on a subsequent test, he raises his mark to 50, has he improved?

This last problem reminds us of a story - it is said to relate to an actual test administered at the University of Melbourne some 25 years ago, but we cannot vouch for its accuracy.

Apparently, this test involved 30 questions and was scored rather differently from the one discussed above. Here a correct answer gave the examinee +1 mark and an incorrect answer -1. On this scheme, the possible marks are the 31 even numbers -30, -28, ..., 0, ..., 28, 30.

One student is supposed to have scored -30, a score with a probability of 2^{-30}, or about 1/1 000 000 000, of being achieved by chance. It was decided that he really knew the work, but had attempted to cross out wrong answers, rather than indicate the correct answer by a cross.

The solution was to give him 30 marks, less 1 for misreading the instructions. His total score thus became 29. Similarly a mark of -28 could be converted into +27 and so on, all negative scores being eliminated!

∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞