Mike Mudge presents the Logical Analysis of Chinese Checkers — a problem from Italy!

The game of Chinese Checkers has numerous definitions; this article is based upon a suggestion made by Michael Meierruth of Milan, Italy and will use the definition given by him.

It is a game played by oneself on a 7 by 7 board with the four 2 by 2 corners removed (see Diagram A). Initially, all the squares are occupied (indicated diagrammatically by 'X') and the game begins with the choosing of a square whose piece is to be removed. After that, each legal move consists of picking up a piece and jumping (vertically or horizontally, *not* diagonally) over another piece onto an empty square. The piece that has been jumped over is then removed. The object of the game is to finish with exactly one piece on the board, that piece being located at a position which was predicted before the initial move was made.

Michael reliased how amazingly difficult it was to find a sequence of moves which would result in exactly one piece being left; because the board is relatively small he attempted to use a computer with a simple backtracking algorithm to generate the complete game tree. Discovering this to be unmanageably large, ways were sought to prune some of its branches; this pruning was 'a fairly creative exercise...apply some deep and sophisticated reasoning...'. The closer to the root of the tree that pruning can be accomplished, the better the chances of exhaustively examining what was originally an unmanageably large search tree.

This 'simple' game seems to have of the order of 10^{20} possible leaves and so drastic pruning is required at some stage!

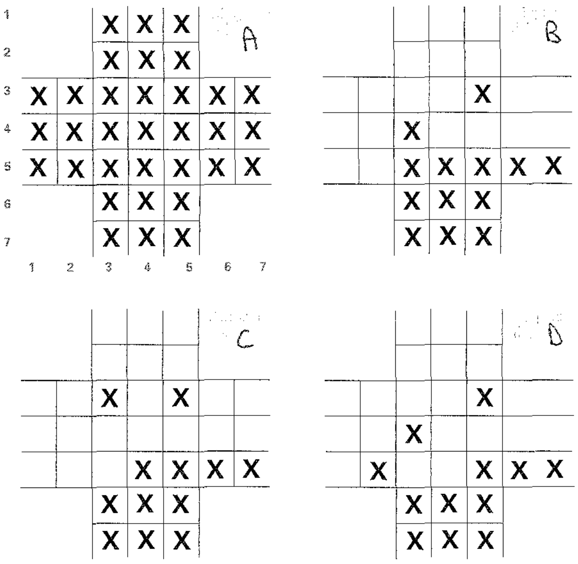
Question (1) Are there starting squares for which there is no solution to the game... regardless of which square is nominated to contain the final residual piece?

In trying to answer this apparently simple question Michael quickly discovered that pruning this game's search tree anywhere at all is extremely difficult, because there appears to be an apparently random sequence of 19 moves where only at the 20th move is it necessary to choose a move different from the 'first available move' in order to arrive at a final position. What makes this unusual is that at the 20th move we are already many billions of nodes into the search tree with no apparent 'qualitative' difference between the first available move (leading to a dead end) and the second available move leading to a final position (see Fig 1).

Notation: Solution sequences will be displayed in six columns — move number, move chosen, number of moves available, piece being moved, piece being jumped over and removed, and destination of piece being moved. (a,b) solution sequence means first piece removed from square (a,b). Thus 4,4 solution sequence leading to Diagram (B) after 19 moves, Diagram (C) after 'first available' 20th move leading to a dead end, and to Diagram (D) after 'second available' 20th move leading to a winning position.

It is conjectured (MM & MM!) that all starting positions will eventually lead to a final ending position given a free choice of square for the residual piece; although these are only known to the author when starting with the central square (4,4) solution and with the square (1,4) and its symmetrical equivalents.

Question (2) How can the qualitative difference between one move and another be defined? Can this 'value' of a move be used to extensively prune a search tree?



Question (3) How can solution sequences be obtained for a specified initial square, i.e. (a,b) solution sequences? For what set of residual square co-ordinates do these exist, is this set related algebraically to (a,b) and if so, how?

Answers to some, or all, of the above questions may be sent to Mike Mudge, 22 Gors Fach, Pwll-Trap, St Clears, Carmarthen, Dyfed SA33 4AQ, tel (0994) 231121, to arrive by 1 April 1993. Any communications received will be judged, using suitable subjective criteria, and a prize will be awarded by *PCW* to the 'best' contribution arriving by the closing date. It would be appreciated if all submissions contained a brief description of the hardware used, details of programs, run times and a summary of the results obtained, all in a form suitable for publication in *PCW.*

Review, Numbers Count -114- September 1992: The Kaprekar Process

This topic generated a very encouraging response, including a solution to Problem 5 in Finnish from Ilari K Luhtavaara of Lohja, Finland. Nigel Backhouse spent about three weeks in Fortran on an Opus V 386 finding all the powerful numbers up to and including degree 9. However, the very worthy prizewinner this month is Paul Rayner of Remingtons Farmhouse, Lamberhurst Road, Horsmonden, Tonbridge, Kent TN12 8LP, using a Viglen III/20 with co-processor. A search for 138hrs 15mins 51.6secs revealed that 28116440335967 is the only powerful number of degree 14 and a further 150hrs found no powerful number of degree 15 less than 302614100000000 (apart from 0 & 1!)

1	1	4	2 4	3 4	4 4	
2	1	3	3 2	3 3	3 4	
3	1	5	1 3	2 3	3 3	
4	1	5	1 5	1 4	1 3	
5	1	5	3 4	3 3	3 2	
10	1	10	1 3	2 3	3 3	
15	1	3	4 5	3 5	2 5	
19	1	5	5 1	5 2	5 3	B
20	2	4	5 4	5 3	5 2	D
31	1	2	5 4	6 4	7 4	winning

Fig 1

Additionally, Paul searched exhaustively for the terminal sequences in the Kaprekar Process (Radix 10) for values of s from 2 to 15, the longest loop occurring when s = 9 and being of length 14. Most of the terminal sequences for s = 16, 17 & 18 are also quoted, a very well documented piece of research culminating in complete sets of powerful numbers for radices from 3 to 10 and degree 1 to 10. All programming done in Turbo Pascal and optimised using Turbo Profiler.

Mike Mudge welcomes correspondence on any subject within the areas of number theory and computational mathematics. Particularly welcome are suggestions for future Numbers Count articles.