Nimx Code Description

Basic 10-Liner for the NOMAM 2014 Competition

by Cliff Hatch

Bouton's Algorithm

Details can be found in Wikipedia, but briefly the algorithm works as follows:

- The numbers of counters on the rows are listed as binary numbers. (The program uses 6 bit numbers, permitting a maximum of 63 counters per row.)
- The bits in each column of the list are summed.
- If the sums for all columns are even, the position is safe (i.e. it provides a path to victory), otherwise it is unsafe.

Variables

Variable	Description
A	Row number from which to remove counters.
B(r,p)	Array of numbers of counters expressed in binary bits. • r denotes the row. • p denotes the binary bit, 1-6, high to low order (reverse ordering economises on code slightly).
C(r)	Array containing the number of counters in each row. • r denotes the row.
D	Bit counter.
E	Number of counters on a particular row, for setting the start position.
F	Move OK flag. • =0, User has not input a valid move yet. • =1, The user has input a valid move.
I	Game over flag • =0, game over • >0, game in progress
J	General purpose counter.
N	Number of counters to remove.
P	Power of 2, used for translating numbers of counters into binary bits.
Q$(1)	String to receive input when the user presses return to replay.
R	Number of rows.

Variable	Description
S	Safe position flag. • =0, Safe • >0, Not safe
T	Number of odd sums.
V	Row counter
W	Used to convert numbers to binary bits by subtracting successive powers of 2.
X(r)	Working copy of C(r), used to manipulate the position to find the machine's move.

Code Comments

0 READ R:DIM C(R):DIM B(R,6):DIM X(R):FOR J=1 TO R:READ E:C(J)=E:NEXT J:DATA 3,3,4,5

- Read the number of rows in the start position.
- Dimension arrays.
- Read the number of counters on each row.

1 DIM Q$(1):CLS :SETCOLOR 1,8,14:SETCOLOR 2,8,0:SETCOLOR 4,1,8:? "NIMX":GOSUB 9:IF RND(0)<0.5 THEN 3

- Initialise graphics.
- Print start position.
- Randomly choose who is to start. Go to line 3 if it is the machine.

2 F=0:WHILE F=0:? "Your Move: ";:INPUT A,N:IF A>=1 AND A<=R:IF N>=1 AND N<=C(A):C(A)=C(A)-N:GOSUB 9:F=1:ENDIF :ENDIF :WEND

- Input, validate and print the user's move.

3 GOSUB 7:IF I=0:? "You Win! Return to Replay";:INPUT Q$:RUN :ELSE :? "My Move: ";:FOR J=1 TO R:X(J)=C(J):NEXT J:A=0:S=1:ENDIF

- Check for user win, and replay when the return key is pressed (redundant if this is the first move of the game, but this action has to share a line with other code to stay below the 10 line limit).
- Initialise variables for calculating the machine's move. Includes making a copy of the current position, C(r).

4 WHILE A<R AND S<>0:C(A)=X(A):A=A+1:N=0:WHILE C(A)>0 AND S<>0:C(A)=C(A)-1:N=N+1:GOSUB 7:WEND :WEND

- Decrease the number of counters in each row, one by one, until a safe position is found or the last counter has been removed from the last row.
- On completion, if a safe move was found and executed S=0, Otherwise S<> 0.

5 IF S<>0 THEN C(A)=X(A):A=RAND(R)+1:WHILE C(A)=0:A=RAND(R)+1:WEND :N=RAND(C(A))+1:C(A)=C(A)-N

- If S<>0 no safe move was found, make a random one instead.
- Choose a row at random, ensure it is not empty and remove a random number of counters from it.

6 ? A;",";N:GOSUB 9:GOSUB 7:IF I=0:? "I Win! Return to Replay";:INPUT Q$:RUN :ELSE :GOTO 2:ENDIF

- Check for machine win. Replay when the return key is pressed.
- Otherwise go to line 2 to process the user's next move.

7 FOR V=1 TO R:P=32:W=C(V):FOR D=1 TO 6:IF W>=P:B(V,D)=1:W=W-P:ELSE :B(V,D)=0:ENDIF :P=P/2:NEXT D:NEXT V:S=0:T=0:I=0

- Start of Subroutine to apply Bouton's algorithm and set flags I and S.
- For each row, translate the number of counters into binary bits, starting with the high order bit (denoting $2^5 = 32$)
- Initialise S, T and I.

8 FOR D=1 TO 6:FOR J=1 TO R:T=T+B(J,D):NEXT J:S=S+T-(2*INT(T/2)):NEXT D:FOR J=1 TO R:I=I+C(J):NEXT J:RETURN

- For each column in the list of binary numbers, set T to the sum of the binary digits.
- Set S to the number of odd sums found over all columns (if this is zero the position is safe).
- Set I to the total number of counters (if this is zero the game is over).
- End of subroutine.

9 ? :FOR V=1 TO R:IF V<10:? " ";:ENDIF :? ;V;:? ;" ";:J=C(V):WHILE J>0:? ;"•";:J=J-1:WEND :? :NEXT V:? :RETURN

- Subroutine to print the current position.