Cannibals and Missionaries

Ruby Lim
Department of Mathematics
The University of Melbourne
Parkville, VICTORIA 3052
AUSTRALIA

Abstract

This paper describes the game of Cannibals and Missionaries modelled using APL68000. It constitutes the original version of a student project report for a fourth-year mathematics course titled "Operations Research Modelling".

1. Introduction

A group consisting of an equal number of cannibals and missionaries seeks to cross a river. A boat, which will hold a limited number of people, is available and can be navigated by any combination of cannibals and missionaries. The object of this game is to transport the entire group across the river. If the cannibals outnumber the missionaries on either side of the river, or in the boat, then the outnumbered missionary or missionaries die(s). Obviously, we do not want this to happen. Say, a general case has N cannibals, N missionaries and M people in the boat, where $N > M > 1$. I will be examining different combinations of N and M.

This mathematical model is often mistaken for a simple game, where the solutions are simple and straight-forward. However, this is not the case. Complex steps are involved in the calculation of which states are feasible, which states have not been encountered before and how to generate the feasible solutions, if they exist at all. Using the tree diagrams, state diagrams and/or further mathematical analysis, solutions can be calculated by hand mathematically. Complications can arise unless the position of the boat is taken into consideration, ie which bank of the river it is on. Because of the many iterations, the computer is used to find the solutions.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission
©1992 ACM 0-89791-478-3 92/0007-0135 $1.50

2. State of the System

This game is mathematically modelled with a state system that is represented by two variables, (c, m), where

c = number of cannibals on the **first** bank of the river

m = number of missionaries on the **first** bank of the river.

It is not necessary to give the number of cannibals and missionaries on the second bank, since the total number of each must always equal N.

From this, transitions can be made from one state to another. The set of allowable states consists of all pairs (c, m) satisfying:

1. The number of people in the boat must be less than or equal to M.

2. The number of cannibals and missionaries on **both** banks and in the boat must satisfy:

   a. $0 \le c \le N$

   b. $0 \le m \le N$, and

   c. $c < m$ or $m = 0$.

Therefore, the number of cannibals and missionaries on either bank and in the boat must satisfy:

   a. $0 \le c \le N$

   b. $0 \le m \le N$

   c. $c = m$ or $m = 0$ or $m = N$.

3. Do not return to a state that has already been encountered (ie to prevent cycles or the problem may never be solved).

To demonstrate the state transitions, let us first consider a specific case where there are 3 cannibals, 3 missionaries and a boat capacity of 2 at most.

Possible transitions are:

(1) (3, 3) --> (1, 3) meaning 'send two cannibals across the river'

(2) (1, 3) --> (2, 3) return one cannibal

(3) (2, 3) --> (0, 3) send two cannibals across the river

(4) (0, 3) --> (1, 3) return one cannibal

(5) (1, 3) --> (1, 1) send two missionaries across the river, and so on.

The state of the system can be illustrated using different geometrical representations.

3. Tree Diagrams

A tree diagram constructs this mathematical model by branching from one state to produce the next state(s). One must be careful to use only allowable states. The transition from (3, 3) to (1, 3) is possible by sending two cannibals across in the boat, but the transition from (3, 3) to (3, 1) is not allowable, because it would result in disaster for the one missionary remaining on the first bank.

In Fig. 1, the first part of a tree diagram shows all possible transitions. Those shown by dotted lines should be ignored since they lead back to a state already encountered.

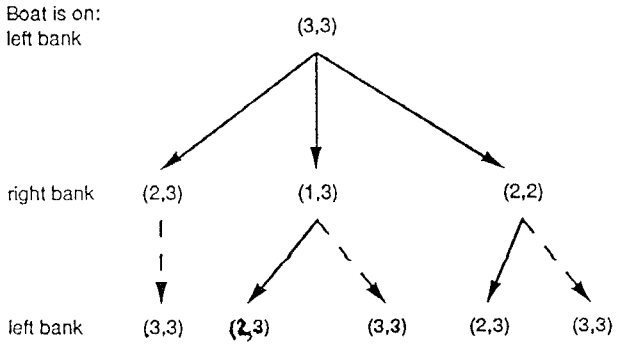


Fig. 1

Evaluation by tree diagrams can be done width-wise or depth-wise. Fig. 1 is constructed width-wise, where a parent state produces all the possible children states, which in turn produce the next generation of all possible states.

4. State Diagrams

This model can also be constructed using state diagrams. The states are represented on a rectangular coordinate system, where every allowable pair (c, m) is plotted. For the problem with three cannibals and three missionaries, c and m can be 0, 1, 2 or 3. There are therefore 16 possible pairs (c, m), but some of these are not allowable. The set of allowable states consists of all pairs (c, m) satisfying

a. $0 \le c \le N$

b. $0 \le m \le N$, and

c. $c = m$ or $m = 0$ or $m = N$.

This set of states is shown in Fig. 2. There are ten states, lying in the shape of the letter Z. This is called the *state diagram.*

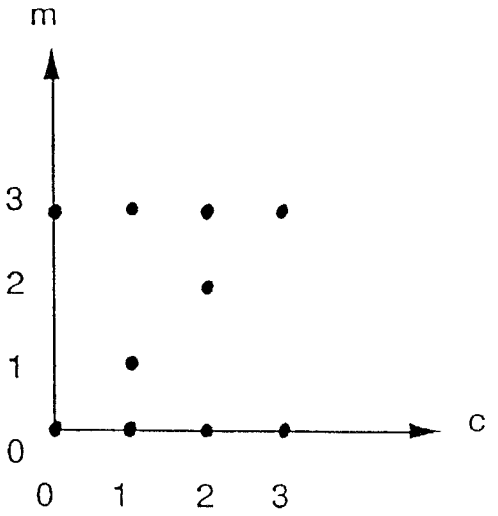


Fig. 2

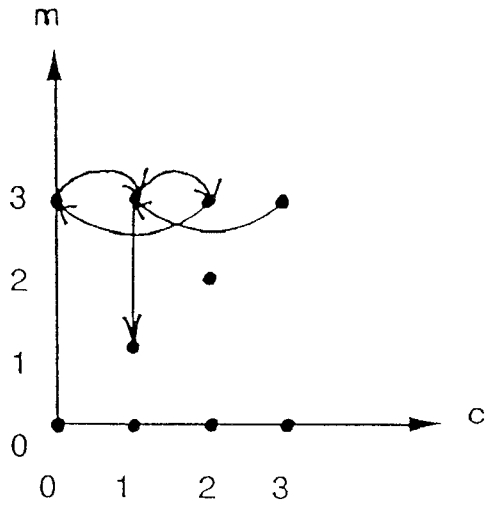
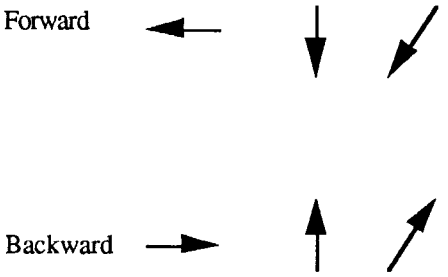


Fig. 3

Transitions between states can also be shown geometrically on the state diagram. Fig. 3 shows the five transitions. The puzzle of the cannibals and missionaries can now be interpreted as calling for a sequence starting at the upper right corner of the state diagram and terminating at the lower left corner.

Use forward/backward/forward movements to represent the boat moving from the first bank to the second, and then back to the first. The first move is a forward one, where the boat leaves the first bank to travel to the second. The last move is also a forward one since all the cannibals and missionaries have been transported to the second bank, with no one to return the boat to its original position. This signifies the end of the game, with a specific solution.



Some states cannot be reached directly, for example, (3, 3) to (0, 3) must first go through from (3, 3) to (1, 3) to (2, 3) to (0, 3), since the boat can only carry a maximum of two, and in some states, the boat may be on the other bank. One must keep in mind the position of the boat. For example, (2, 3) forwards and (2, 3) backwards are different states, and are not to be mistaken for a similar state already encountered!

5. Existence and Uniqueness

Certain combinations of N and M will not be solvable. For example, four cannibals and four missionaries cannot be taken safely across a river with a boat holding only two people. On a state diagram, classify the allowable states into three classes:

T those along the top (m = N)

B those along the bottom (m = 0)

D those along the diagonal (0 < m = c < 3).

Since the boat only holds two people, it is impossible to go directly from a state of type T to one of type B. Thus, any solution of the game must include a state of type D. To leave the T states without returning, a transition from (2, 4) will go forwards to (2, 2). The next transition is backwards to (3, 3) (see Fig. 4). Thereafter no forward move exists to reach the bottom states. At most four of the eight people can be transported across the river.

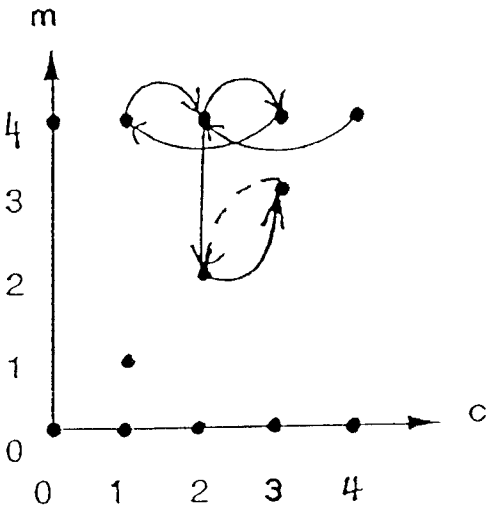


Fig. 4

Fig. 5 represents the combination of five cannibals and five missionaries with a boat capacity of three. At least eleven crossings are required to transport all ten people safely across the river.

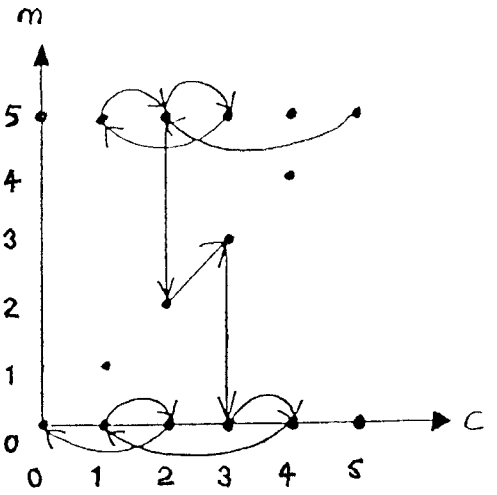


Fig. 5

In cases where the boat capacity is four or mire, any number of cannibals and missionaries can be transported across the river, by traversing along the diagonal of the state diagram. Fig. 6 shows the nine crossings needed for the combination of six cannibals and six missionaries with a boat capacity of four.

Fig. 6 is also the solution to six cannibals and six missionaries with a boat capacity of five, using the diagonal states. However there is a solution involving only seven crossings as shown in Fig. 7.

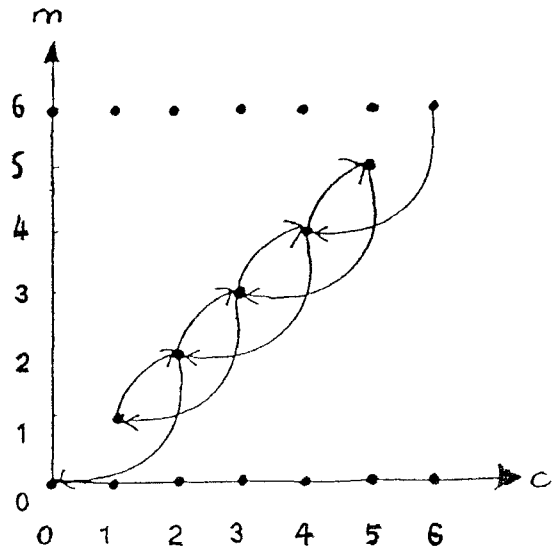


Fig. 6

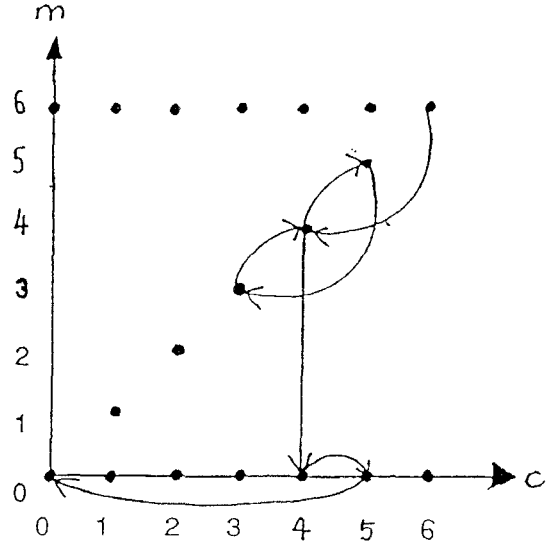


Fig. 7

Feasible solutions always exist for any combination with a boat that holds at least four people, by traversing along the diagonal of the state diagram. However, the number of crossings may not be minimal. Optimal solutions, in terms of shortest path, may be obtained using less straight-forward combinations of top and bottom states.

6. Computer Program

The computer program is written in APL68000, using the Apple Macintosh computer. The code involves complex functions which manipulate the input data to calculate how to obtain all the possible states, how to generate the new allowable states, how to display the output neatly on a page, so as to make this game **look** simple and straight-forward.

The size of the output, when feasible solutions exist, is large because of the number of them. The computer model considers three variables, (B, c, m), where B is the position of the boat, ie forward (1) signifies the boat is on the first bank and about to go to the second bank, and backward (0) signifies the boat is on the second bank and about to go to the first; c and m are as defines earlier.

Note that the game always starts with an equal number of cannibals and missionaries, say N. Obviously, the boat capacity M is less than N.

6.1 Input

When the mainline is run, instructions tell the user to input the number of cannibals and missionaries, and the boat capacity, and whether he wants a graphical representation of the output or not. To make the game interesting, the user is asked to input N and M values that satisfy $1 < M < N$. Thus, the smallest value of N is recommended to be 3 and the smallest value of M is recommended to be 4.

6.2 Solution

During execution, two tables are constructed. One table, A, stores the current solutions, and the other, B, stores the partial solutions. Each table has three columns; the first column indicates the bank that the boat is on; the second column indicates the number of cannibals on the **current** bank; the last column indicates the number of missionaries on the **current** bank; unlike the tree diagram and state diagram representations above, which indicated the number of cannibals and missionaries on the **first** bank. It is more convenient to consider current bank's states in the program, since only one algorithm is needed to travel between banks.

Say we start with (1, 3, 3). Put this in state into B and A. (see Fig. 8).

The program will use various functions to generate the new allowable states. Put them into B (see Fig. 8). These are stacked bottom up. Put the last entry of B into A, also stacked bottom up . From here, iterate by generating new allowable states, and stacking them into B and A. When a state in A and B is one that has already been encountered, remove it from A and B. Put the last entry from B into A. A feasible solution is found when the state (0, 3, 3) is reached. There is no feasible solution when the last entry in B cannot generate states that are allowable.

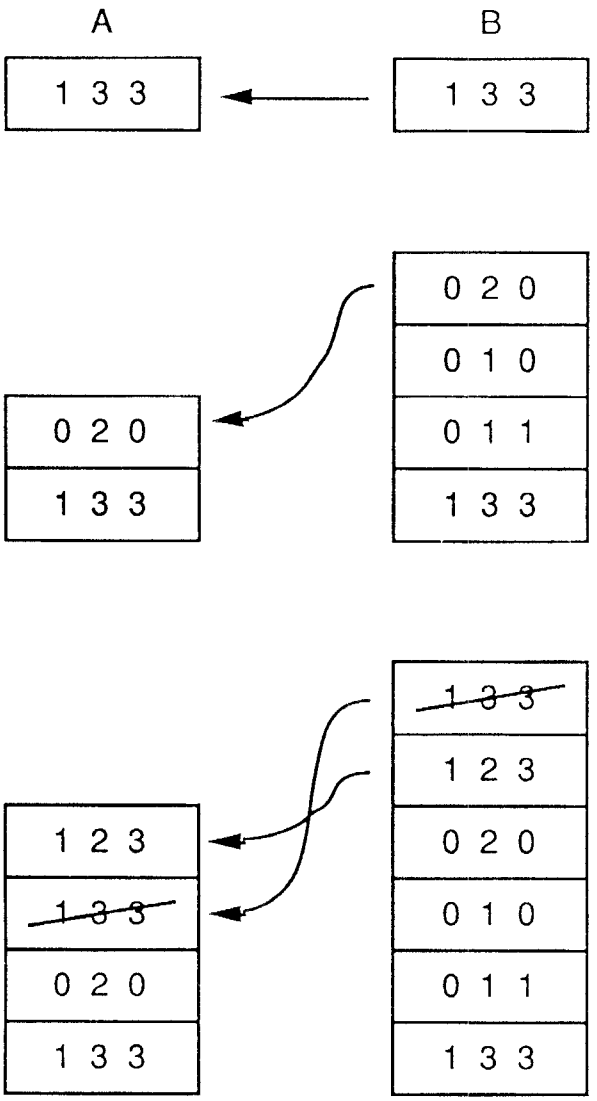


Fig. 8

In terms of a tree diagram, evaluation is implemented depth-wise, where all new states in a branch are generated before going on to the next branch; and not width-wise, where all new states on the same level of the tree are generated before going to the next level. Depth-wise evaluation is used in this program because width-wise evaluation involved more complex data storage algorithms.

6.3 Output

The output prints out all the shortest path solutions to the game, with instructions to send a certain number of cannibals and/or missionaries across the river, and then return a certain number of them. The solution is printed out from top down even though the data was stored bottom up as it was produced.

Here is a sample output from the case with four cannibals, four missionaries and a boat capacity of two at most:

```
THERE IS NO FEASIBLE SOLUTION
```

With three cannibals, three missionaries and a boat capacity of two:

```
THE FEASIBLE SOLUTIONS ARE:

      CAN MIS        CAN MIS
SEND    2 0    SEND    2 0
RETURN  1 0    RETURN  1 0
SEND    2 0    SEND    2 0
RETURN  1 0    RETURN  1 0
SEND    0 2    SEND    0 2
RETURN  1 1    RETURN  1 1
SEND    0 2    SEND    0 2
RETURN  1 0    RETURN  1 0
SEND    2 0    SEND    2 0
RETURN  1 0    RETURN  0 1
SEND    2 0    SEND    1 1


      CAN MIS        CAN MIS
SEND    1 1    SEND    1 1
RETURN  0 1    RETURN  0 1
SEND    2 0    SEND    2 0
RETURN  1 0    RETURN  1 0
SEND    0 2    SEND    0 2
RETURN  1 1    RETURN  1 1
SEND    0 2    SEND    0 2
RETURN  1 0    RETURN  1 0
SEND    2 0    SEND    2 0
RETURN  1 0    RETURN  0 1
SEND    2 0    SEND    1 1
```

All the shortest path feasible solutions are printed.

During the input of data, the user will be asked whether he wants graphical output or not. If he so desires, this program is designed to return a simple graphical representation, provided the user is satisfied with only one shortest path feasible solution. Otherwise, all the shortest path feasible solutions are printed without a graphical representation. Here are the first four steps of the first shortest path feasible solution, printed with a graphical representation:

```
CCC
MMM


SEND    2 CANNIBALS 0 MISSIONARIES
C                     CC
MMM

RETURN 1 CANNIBALS 0 MISSIONARIES
CC                    C
MMM
```

```
SEND    2 CANNIBALS 0 MISSIONARIES
                      CCC
MMM

RETURN 1 CANNIBALS 0 MISSIONARIES
C                     CC
MMM
```

The graphical representation is prefered visually as the user is able to see each crossing clearly on the screen. improved user-interaction can be achieved by actually playing a game with him.

6.4 Problems

Execution of the program is slow for large M, and not so much for large N. this is because there are numerous solutions to the game, and every solution is found. So, this is programmed to print out only the solutions with the shortest path. However, there are still many of them! For example, four cannibals, four missionaries with a boat capacity of three has 332 shortest path solutions!

This program is designed to print out all the solutions with the fewest number of crossings. Options are available as to the type of output that the user wants. The code can be modified to print out only the first K solutions, where K is a specific number defined by the programmer or the user. This will reduce execution time considerably. However, the first K solutions may not necessarily be optimal in terms of shortest path. So the code can also be modified to print out only the first K shortest path solutions, by ranking in order of preference for the fewest number of crossings. The first shortest path solution is ranked one, the next shortest path solution is ranked two, and so on. All the feasible solutions still have to be found and compared to the first K shortest path solutions. This means that execution time could still take a while, especially when there are many feasible solutions.

Other options include printing just one feasible solution, or just one shortest path feasible solution. Otherwise, print out the first K solutions that have x or less number of crossings, which may not necessarily be the shortest path. (x is a specific number defined by the programmer or user).

Therefore, this program will output all the optimal solutions in terms of shortest path. However, for boat capacity of three or greater, execution time of the program takes a few minutes, and no output is printed in some cases because the APL workspace is full!

7. Conclusion

There is a feasible solution whenever a boat can hold at least four people, or the number of cannibals/missionaries is less than twice the boat capacity, ie $M \geq 4$ and $1 < N < 2M$.

To traverse from the top of the state diagram to the bottom when $M < 4$, two vertical jumps are required (see Fig. 5). When $M \geq 4$, traversal along the diagonal gives a feasible solution (see Fig. 6).

If the boat can hold an even number of people, the minimum number of crossings is obtained by traversing the diagonal of the state diagram. If the boat capacity is odd, this solution may not be optimal in terms of the minimum number of crossings.

Other variants to make the game more interesting, is to modify the rules. For instance, the cannibals are stronger than the missionaries to the extent that the missionaries are safe only when they exceed the cannibals in number (on both banks and in the boat). Or, the boat can only be navigated by a particular combination of cannibals and missionaries.

The next stage in the computer program would be to replace the 'C' and 'M' with faces or some sort of human figure. Also, a pictorial representation can be designed for the screen, where the output does not scroll up every time a new move is made. The picture may consist of a river, a boat and some human figures. For the more ambitious, colour and sound effects may even be included! but most importantly, execution time needs to be improved to cater for larger problems when the boat capacity is increased.

Unfortunately, at this stage, the program has relatively little "user friendliness". Interaction occurs only when the user inputs the necessary data, which is pretty basic. An interactive approach can be developed by actually playing a game with the user. For example, by asking what his next move is. If his move is illegal, a message can be returned saying, "The boat can carry only two people" or "There are more cannibals than missionaries on the first bank, one missionary dies".

8. Reference

R. Bellman, K. L. Cooke and J. A. Lockett, **Algorithms, Graphs and Computers**, p. 196-212, Academic Press (1970).

9. Acknowledgements

Mr Craig Brown, Canon Information Systems Research Australia, 1 Thomas Holt Drive, North Ryde, NSW 2113, Australia.

Dr Moshe Sniedovich, Mathematics Department, The University of Melbourne, Parkville, VIC 3150, Australia.

Gwynne Scotford Management Consultants, P O Box 196, Neutral Bay, NSW 2089, Australia. I can be contacted at this address.

Appendix

Computer Code and Comments

The computer code is short and precise and was developed with elegance in mind. Unfortunately, efficiency for large scale problems has not been achieved. The program consists of eleven functions, namely CANMIS, INPUT, CM, POSSCOMB, NEWSTATES, INBOAT, SHAPE, JOIN, GRAPHIC, GRINBOAT and OUTPUT.

```
CANMIS;⎕IO;A;B;M;N;Y;Z;GO
[1]   ⎕IO←0
[2]   INPUT
[3]   CM
[4]   →(GO=1)/0
[5]   OUTPUT
```

CANMIS is the mainline of this program, consisting of three functions called INPUT, CM and OUTPUT.

```
  INPUT
 [1]  NINPUT:'PLEASE TYPE IN THE
      NUMBER OF CANNIBALS AND
      MISSIONARIES:' ⋄ N←⎕
 [2]   →(N>2)/MINPUT
 [3]   'SHOULD BE 3 OR GREATER'
 [4]   →NINPUT
 [5]  MINPUT:'PLEASE TYPE IN THE
   NUMBER OF PEOPLE IN THE BOAT:'
       ⋄ M←⎕
 [6]   →((M>1)^M<N)/GOINPUT
 [7]   'MUST BE GREATER THAN 1 BUT
LESS THAN CANNIBALS AND MISSIONARIES'
 [8]   →MINPUT
 [9]  GOINPUT:'DO YOU WANT GRAPHICAL
     OUTPUT? (1=YES, 0=NO)' ⋄ GO←⎕
[10]  →((GO≥0)^GO≤1)/INITIALISE
[11]  'ERROR: PRESS 1 OR 0 ONLY'
[12]  →GOINPUT
[13] INITIALISE:A←B←1 3⍴(1,N,N)
[14]  Z←0 0⍴0
[15]  Y←0 0⍴0
```

INPUT is also an initialization function.

```
      CM;TEMP;NUMBER
[1]   NUMBER←1000
[2]   POSSCOMB
[3]  LOOP:B←(NEWSTATES(B[0;])),[0]B
[4]   A←B[0;],[0]A
[5]  LOOP2:→(^/(A[0;]=(0,N,N)))/FEAS
[6]   →(~^/1↓(B^.=A[0;]))/LOOP
[7]   →(^/((⍴A)=⍴B))/0
[8]  LOOP3:→((B[0;0]≠B[1;0]))/DIFF
[9]   B←1 0↓B
[10]  A[0;]←B[0;]
[11]  →LOOP2
[12] DIFF:B←1 0↓B
[13]  A←1 0↓A
[14]  →LOOP3
[15] FEAS:→(GO=1)/GOUTPUT
[16]  →((⍴A[;0])>NUMBER)/LOOP3
[17]  →((⍴A[;0])<NUMBER)/CUT
[18]  Z←Z SHAPE TEMP←INBOAT⍟A
[19]  →((¯1↑⍴Z)<60)/LOOP3
[20]  Y←Y JOIN Z
[21]  Z←0 0⍴0
[22]  →LOOP3
[23] CUT:NUMBER←⍴A[;0]
[24]  Z←0 0⍴0
[25]  Y←0 0⍴0
[26]  Z←Z SHAPE TEMP←INBOAT⍟A
[27]  →LOOP3
[28] GOUTPUT:GRAPHIC⍟A
```

```
POSSCOMB
[1] PC←((3×N+1),2)⍴0
[2] PC[;0]←(3×N+1)⍴⍳N+1
[3] PC[;1]←((N+1)⍴0),((N+1)⍴N),⍳N+1
[4] PC←¯1 0↓1 0↓PC
```

POSSCOMB is all the possible combinations of the states that satisfy the rules.

```
NST←NEWSTATES CS;XY
[1] XY←((⍴PC)⍴(CS[1],CS[2]))-PC
[2] XY←((XY[;0]≥0)^(XY[;1]≥0))/[0]XY
[3] XY←((XY[;0]+XY[;1])≤M)/[0]XY
[4] XY←((XY[;1]=0)∨(XY[;0]≤XY[;1]))
/[0]XY
[5] XY←((XY[;0]≠0)∨(XY[;1]≠0))/[0]XY
[6] NST←XY+(⍴XY)⍴((N,N)-
(CS[1],CS[2]))
[7] NST←(((¯1↓⍴NST),1)⍴~CS[0]),NST
```

From the current state, NEWSTATES uses POSSCOMB to generate the new allowable states, some of which are elements of POSSCOMB, and these new states satisfy:
a. $X \geq 0$ and $Y \geq 0$ and $X + Y \geq 0$
b. $X + Y \leq M$
c. $X \leq Y$ or $Y = 0$, where X = number of cannibals **in the boat** and Y = number of missionaries **in the boat.**
CS is the current situation.

```
      ∇IN←INBOAT A;AB;AX;C;P
[1]   AB←¯1 1↓A
[2]   AX←1 1↓A
[3]   C←AX+AB-(⍴AB)⍴(N,N)
[4]   P←2 6⍴'SEND  RETURN'
[5]   P←((⍴C[;0]),⍴P[0;])⍴P
[6]   IN←P,⍉C
```

INBOAT calculates the number of people in the boat.

```
      ∇R←OLD SHAPE NEW
[1]   →(0=1↑⍴OLD)/OLDEMPTY
[2]   R←OLD,' ',' ',' ',' ',' ',⍉NEW
[3]   →END
[4]  OLDEMPTY:R←⍉NEW
[5]  END:

R←Y JOIN Z
[1]   →(0=1↑⍴Y)/YEMPTY
[2]   R←Y,[⎕IO]' ',[⎕IO]' ',[⎕IO]Z
[3]   →END
[4]  YEMPTY:R←Z
[5]  END:
```

JOIN is called when the screen is full with feasible solutions.

```
       OUTPUT;A
[1]   →((0=1↑⍴Z)^0=1↑⍴Y)/NOFEAS
[2]   'THE FEASIBLE SOLUTIONS ARE:'
[3]   ''
[4]   A←(⍴Y)⌈⍴Z
[5]   (A[1]+2)⍴'      CAN MIS    '
[6]   →(∨/(0,0)=⍴Y)/PRINTZ
[7]   Y
[8]   ''
[9]   ''
[10] PRINTZ:Z
[11]  →END
[12] NOFEAS:'THERE IS NO FEASIBLE
     SOLUTION'
[13] END:
```

GRINBOAT (Graphical INBOAT) calculates the number of people in the boat. This function is an extension to INBOAT, which is used in producing standard output. GRINBOAT is used in producing a graphical output.

```
∇IN←GRINBOAT A;AB;AX;C;P;Q;R;S;T;I
[1]   AB←¯1 1↓A
[2]   AX←1 1↓A
[3]   C←AX+AB-(⍴AB)⍴(N,N)
[4]   P←2 6⍴'SEND  RETURN'
[5]   P←((⍴C[;0]),⍴P[0;])⍴P
[6]   S←((⍴C[;0]),10)⍴' CANNIBALS'
[7]   T←((⍴C[;0]),13)⍴' MISSIONARIES'
[8]   Q←⍉((⍴C[;0]),1)⍴C[;0]
[9]   R←⍉((⍴C[;1]),1)⍴C[;1]
[10]  I←¯1
[11] LOOP:I←I+1
[12]  →(I≥⍴C[;0])/END
[13]  →(C[I;0]>1)/MISS
[14]  S[I;9]←' '
[15]  →(C[I;0]=1)/MISS
[16]  Q[I;]←(⍴Q[I;])⍴' '
[17]  S[I;]←(⍴S[I;])⍴' '
[18] MISS:→(C[I;1]>1)/LOOP
[19]  T[I;10]←'Y'
[20]  T[I;11]←' '
[21]  T[I;12]←' '
[22]  →(C[I;1]=1)/LOOP
[23]  R[I;]←(⍴R[I;])⍴' '
[24]  T[I;]←(⍴T[I;])⍴' '
[25]  →LOOP
[26] END:IN←P,Q,S,R,T
```

```
      ∇GRAPHIC[⎕]∇
      ∇GRAPHIC A;INB
 [1]   INB←GRINBOAT A
 [2]  LOOP:→A[0;0]/PRINT
 [3]   A[0;1]←N-A[0;1]
 [4]   A[0;2]←N-A[0;2]
 [5]  PRINT:
 [6]   (20⍴((A[0;1])⍴'C'),'
   20 blanks ),(N-A[0;1])⍴'C'
 [7]   (20⍴((A[0;2])⍴'M'),'
   20 blanks ),(N-A[0;2])⍴'M'
 [8]   IGNORED←⎕DL 2
 [9]   ''
[10]   ''
[11]   →(^/(0,0,0)=A[0;])/END
[12]   INB[0;]
[13]  IGNORED←⎕DL 1
[14]   A←1 0↓A
[15]   INB←1 0↓INB
[16]   →LOOP
[17] END:'ALL THE CANNIBALS AND
      MISSIONARIES HAVE CROSSED
      THE RIVER SAFELY'
```

GRAPHIC calls GRINBOAT and returns a graphical representation of the output.